# FINE-GRAINED RECOGNITION:
# DATA, RECOGNITION, AND APPLICATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jonathan Krause
October 2016

# Abstract

Fine-grained recognition refers to the task in computer vision of automatically differentiating similar object categories from one another, *e.g.* species of birds, types of cars, breeds of dogs, or varieties of aircraft. Since this is a task that the majority of humans are untrained in, any progress has the promise of augmenting human vision. Applications are numerous: beyond simply being able to describe the world in more detail, fine-grained recognition can be used for improved scene understanding, studying society, and even analyzing biodiversity.

Unfortunately, fine-grained recognition is extremely challenging. Even acquiring data is difficult due to the expertise required in annotation, and differences between categories can still be extremely subtle and formidable to learn. Despite this, enormous progress has been made on fine-grained recognition in recent years, with the works described herein contributing to these advances.

In this work, I will describe the efforts I have made toward tackling fine-grained recognition. These include four projects aimed at improving automatic recognition of fine-grained categories, which look at both the data and recognition algorithms used. I also describe an application of fine-grained recognition in understanding society from images. Finally, I present a work looking at finer-grained natural language descriptions of images.

# Acknowledgments

Thank you to my friends and colleagues in the Stanford Vision Lab, who have been the source of countless interesting discussions and, more generally, made the-day to-day of my PhD downright enjoyable: Alexandre Alahi, Chris Baldassano, Lamberto Ballan, Jia Deng, Alireza Fathi, Timnit Gebru, Michelle Greene, Oliver Groth, Albert Haque, Kenji Hata, De-An Huang, Marius Cătălin Iordan, Justin Johnson, Armand Joulin, Andrej Karpathy, Ranjay Krishna, Jia Li, Joseph Lim, Sean Ma, Juan Carlos Niebles, Guido Pusiol, Vignesh Ramanthan, Olga Russakovsky, Hao Su, Kevin Tang, Bangpeng Yao, Serena Yeung, and Yuke Zhu.

Thank you to my co-authors, who have contributed to or otherwise helped to shape all of my work: Alex Berg, Michael Bernstein, Duyun Chen, Jia Deng, Tom Duerig, Timnit Gebru, Andrew Howard, Zhiheng Huang, Hailin Jin Andrej Karpathy, Aditya Khosla, Daphne Koller, Jia Li, James Little, Sean Ma, David Meger, Bojan Pepik, James Philbin, Olga Russakovsky, Benjamin Sapp, Sanjeev Satheesh, Bernt Schiele, Michael Stark, Hao Su, Alexander Toshev, Yilun Wang, Jianchao Yang, and Howard Zhou.

A special thanks to Jia Deng, Michael Stark, Howard Zhou, and Timnit Gebru, for shaping how I think about research and being able to put up with me for long stretches of time on-end.

Thank you to my reading and oral committee members, who have provided useful feedback in the preparation of this document and otherwise throughout the years: Jia Deng, John Duchi, Stefano Ermon, and Silvio Savarese. An additional thanks to Silvio Savarese and Percy Liang for being on my Qualifying Examination committee.

Thank you to other friends, who have kept me company when I have been exhausted after paper deadlines (and moments in between). Names are too numerous to list, but include Gabor Angeli, Justin Aschenbener, Jason Aschenbener, Christine Cheng, David Held, Irene Kaplow, Manuel Lagang, Brian Mok, Calvin Murdock, Saurabh Pandey, Richard Wang, Tony Wu, Samuel Yang, and Stephan Zheng.

Thank you to my family, who have offered me their support as I experienced the ups and downs that is the PhD roller coaster – I would not have made it without their love and support.

Thank you to those who I've forgotten to include in this section.

Finally, thank you to my advisor Fei-Fei Li, who has supported me and helped me grow for five long years that have gone by altogether too quickly. Thank you for taking a chance on me when I was a new PhD student, thank you for putting up with me when I didn't listen, and thank you for all of the food.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

One of the defining tasks of modern computer vision is *object classification* – differentiating between object categories. This task, sometimes also referred to as *object recognition*, has long been a fundamental problem in the quest to turn pixels into semantics, and progress has been slow but steady over many years. As a result of the combined efforts of the field, today object recognition stands mostly solved [151, 97, 167], and progress in object recognition has additionally translated to improvements in other key problems [65, 148].

A more recent, but harder, problem, is *fine-grained recognition*, the task of differentiating between very similar object categories, *e.g.* species of birds, types of cars, breeds of dogs, or varieties of aircraft. In contrast to traditional object recognition, fine-grained recognition typically requires reasoning about very subtle differences between images (Fig. 1.1), and entails a level of human expertise that standard object recognition does not. Due to the challenge of the task, any progress in fine-grained recognition is extremely promising, since human vision can be effectively augmented by even an imperfect computer vision algorithm. Applications of fine-grained recognition include biodiversity analysis [20], improved scene understanding [164], analyzing the world [63], and teaching people about unfamiliar categories [17].

There are three main questions to consider about fine-grained recognition: First, what type of data do you use to train with (Data)? Since fine-grained recognition is a task inherently difficult for humans, acquiring data can be extremely challenging and

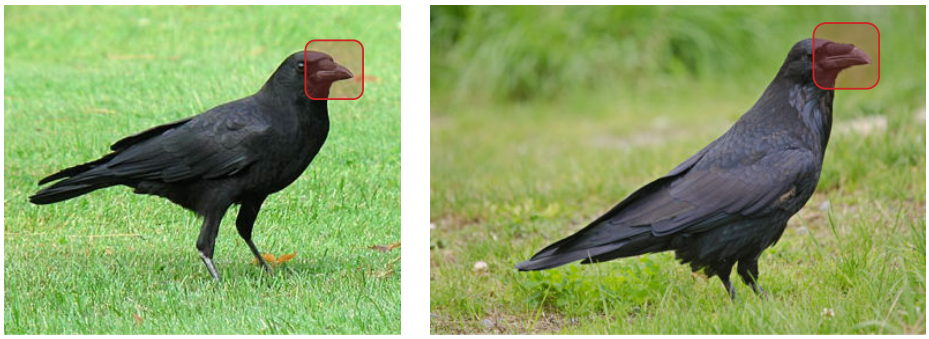



Figure 1.1: Differences between fine-grained categories can be extremely subtle and challenging to identify. Here, these two bird species are identical in color and overall shape, but differ ever so slightly in the size and shape of their beaks. Effective fine-grained recognition thus typically requires making correspondences across images reasoning about local differences.

laborious, so reducing constraints on data collection as much as possible is critical. Once data is collected, what type of algorithm should you use to effectively recognize fine-grained categories (Recognition)? Besides the strong interplay between recognition algorithms and the type and quantity of data collected, additional questions include the choice to reason about object pose, the use of part-based models, effective feature representations, and model complexity. Last, how might you even want to use fine-grained recognition in the wild (Application)? Making fine-grained recognition not only work, but actually be useful, is of key consideration. In this work, I will describe the efforts I have made toward solving fine-grained recognition, which touch upon these three areas: *Data, Recognition, and Application*.

This dissertation begins with works that address the core algorithmic problem of designing scalable methods to recognize fine-grained categories. In Chapter 2, I introduce a method and dataset for fine-grained recognition of cars, with the key contribution being an algorithm for bringing state-of-the-art 2D methods for recognition into 3D. This approach, which is designed to work on rigid objects like cars, uses freely available CAD models to predict 3D geometry, which is then used to normalize for viewpoint and thereby draw correspondences across images.

In Chapters 3 and 4 I detail two approaches for automatically learning parts for

fine-grained recognition. By detecting the same object parts in different images, one can achieve a form of pose normalization, undoing a key axis of variation across images. Of note, the method in Chapter 3 was among the first to investigate learning features for fine-grained recognition in a neural network framework, and the approach in Chapter 4 introduces a novel method for co-segmentation and segmentation refinement using bounding boxes, which has applications beyond fine-grained recognition.

In Chapter 5 I describe an application of fine-grained recognition in understanding society. The key insight is that fine-grained categories, beyond being 1 of $K$ labels, carry valuable semantic information – for example, a 2008 Toyota Prius was made around 2008, has a Japanese manufacturer (Toyota), is a Hatchback, has high fuel efficiency (45 MPG), is a Hybrid, and costs $9,500 as of 2012. This information, when detected across different geographical locations, can be used to analyze neighborhoods and cities across the United States.

In Chapter 6 I return to the core problem of recognizing fine-grained categories, but approach it from a different angle. Instead of keeping the data fixed and improving the recognition algorithm, I investigate ways of using freely and publicly available information from the web for fine-grained recognition, inverting the problem into one where the algorithm is fixed and the data varies. This leads to large improvements in recognition performance (state of the art on four fine-grained datasets as of this writing), does not require manually collecting any expert annotations, uses simple and generic methods of recognition, and allows scaling fine-grained recognition up to more than 10,000 categories.

Finally, in Chapter 7 I present a work that makes a recent problem in computer vision, *image captioning*, and makes it finer-grained, producing descriptions at the level of paragraphs instead of single sentences. Our approach involves decomposing the problem in both the input and output space, representing images by collections of regions of interest and representing paragraphs hierarchically in terms of their constituent sentences. Chapter 8 concludes with final thoughts.

# Chapter 2

# 3D Object Representations for Fine-Grained Recognition

While 3D object representations are being revived in the context of multi-view object class detection and scene understanding, they have not yet attained wide-spread use in fine-grained categorization. State-of-the-art approaches achieve remarkable performance when training data is plentiful, but they are typically tied to flat, 2D representations that model objects as a collection of unconnected views, limiting their ability to generalize across viewpoints. In this paper, we therefore lift two state-of-the-art 2D object representations to 3D, on the level of both local feature appearance and location. In extensive experiments on existing and newly proposed datasets, we show our 3D object representations outperform their state-of-the-art 2D counterparts for fine-grained categorization and demonstrate their efficacy for estimating 3D geometry from images via ultra-wide baseline matching and 3D reconstruction.

This work was done with Michael Stark, Jia Deng, and Fei-Fei Li, and appeared in [95].

## 2.1 Introduction

Three-dimensional representations of objects and scenes have been deemed the holy grail since the early days of computer vision due to their potential to provide more

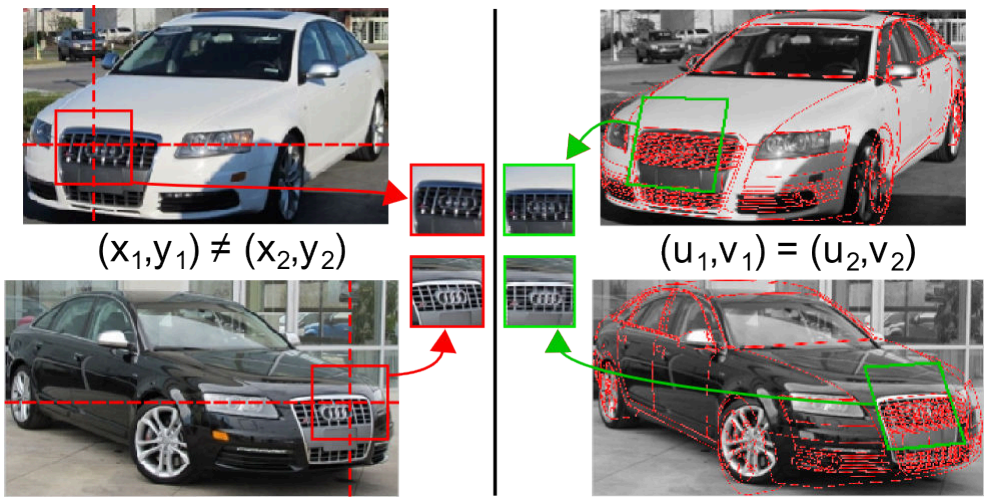



Figure 2.1: Corresponding patches may have vastly different image coordinates and appearances. Our 3D representations make their positions and appearances directly comparable.

faithful and compact depictions of the visual world than view-based representations.

Recently, 3D methods have been revived in the context of multi-view object class detection [152, 110, 142] and scene-understanding [77, 69]. For these applications, 3D representations exhibit favorable performance due to their ability to link object parts across multiple views.

Surprisingly, these strengths have hardly been exploited in fine-grained recognition, one of the most active areas in computer vision today.

There, most state-of-the-art approaches still rely entirely on "flat" image representations that model both the appearance of individual features and their location in the 2D image plane. The former typically takes the form of densely sampled local appearance features such as SIFT [116], often followed by a non-linear coding step [185]. For the latter, various spatial pooling strategies have proven successful, ranging from global histograms [134] over spatial pyramids [102] to local search regions [46]. While these approaches have delivered remarkable performance in fine-grained categorization tasks that are difficult even for humans [56, 180, 204], they are still limited by the need to observe a relatively dense viewpoint sampling for each category in order to learn reliable models.

In this paper, we therefore take a different route, and follow the intuition that the distinctive features of a fine-grained category, such as the characteristic grille of the car in Fig. 2.1, are most naturally represented in 3D object space, not in the 2D image plane – this comprises both the appearance of the features (appearance varies with viewpoint, so a viewpoint-independent appearance representation is desired) and their location with respect to an object (the grille appears in a specific region of the 3D car surface, not necessarily in a particular image position). We establish the notion of 3D object space by first obtaining an estimate of the 3D geometry of an object, then representing features relative to this geometry. Specifically, the geometry estimate allows rectification of an image patch with respect to the estimated surface normal at its center point and characterizes its location as coordinates on the 3D object surface.

The basis of our implementation is given by two state-of-the-art 2D object representations. The first, spatial pyramid matching [102], specifically using locality constrained linear coding (LLC [185]), has attained wide-spread use due to its consistently high performance on various image categorization benchmarks, combining local feature coding with a spatial pyramid representation. The second representation, BubbleBank (BB [46]), has recently been shown to outperform prior work in fine-grained categorization. It relies on extracting discriminative patches from training images, convolving them in local regions within a test image, and using the responses as features.

Our paper makes the following contributions: First, we lift two state-of-the-art 2D object representations to 3D w.r.t. both the appearance and location of local features. We demonstrate the resulting 3D object representations outperform both their respective 2D counterparts and state-of-the-art baselines in fine-grained categorization. Second, we introduce a new dataset of 207 fine-grained categories, separated into two subsets: a small-scale, but ultra-fine-grained set of 10 BMW models, and a large-scale set of 197 car types. Third, we demonstrate the usefulness of our 3D object representation for estimating 3D geometry from test images in the form of ultra-wide baseline matching [208]. Fourth, we provide first experimental results on the challenging task of 3D reconstruction of fine-grained categories, which, to our knowledge, has not been attempted in the literature before.

## 2.2 Related Work

Fine-grained recognition is a growing subfield of computer vision. Driven by real-life applications, the focus has so far mostly been on distinguishing animate categories, such as flowers [134], leaves [100], dog breeds [115, 140], and birds [56, 180, 204]. For these categories, the challenge consists in capturing subtle appearance differences (such as a differently colored beak of a bird) despite variations in articulated pose, which are most reliably handled by collecting and memorizing discriminative, local appearance features from each available object view [198, 46].

**The role of object parts.** It has been realized that spatial information typically aids categorization, either for providing a frame of reference in which appearance features are computed (spatial pooling [102, 185, 56, 180, 204, 46]), or as a feature in itself [73, 164]. For both, object part detectors have proven to provide reliable spatial information, as constellations [73], deformable parts [180, 164], and poselets [22, 204]. While these models successfully leverage spatial information, they are still "flat", *i.e.* , built on independent views.

A remarkable exception to this trend is the recent work by Farrell *et al.* [56], which implements local appearance features and spatial pooling relative to a volumetric 3D bird model. While our work is similar in spirit, it goes beyond [56] in several important directions. First, in contrast to [56], our work does not rely on extensive annotation of training data. Instead, we leverage existing 3D CAD models for the basic-level object class of interest, without the need for any manual intervention. Second, we explicitly design our methods to be robust against errors in the estimation of rough 3D geometry by pooling over multiple predictions of coarse category and viewpoint instead of relying on a single prediction. Third, instead of focusing on a single representation for spatial pooling (two 3D ellipsoids for a bird's head and body for [56]), we compare and extend into 3D two different, state-of-the-art 2D methods for spatial pooling (SPM and BB), demonstrating improved performance.

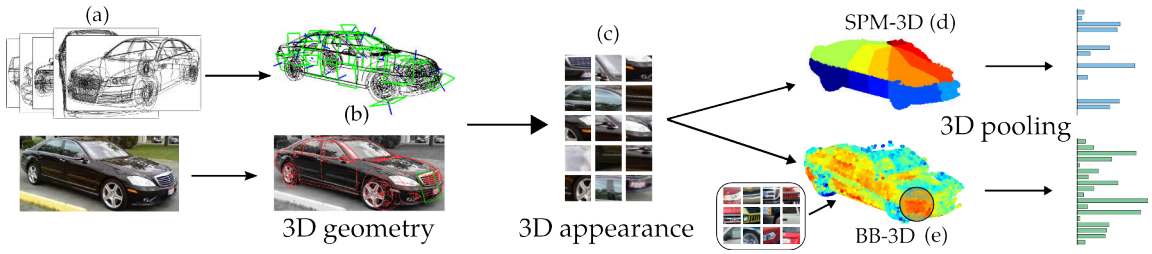



Figure 2.2: An overview of how we estimate 3D geometry and lift SPM and BB to 3D. See text for details.

**3D representations for recognition.** We draw inspiration from approaches in multi-view object class recognition that leverage 3D representations to establish correspondences across different viewpoints [208, 142, 70, 146]. While these prior approaches establish correspondences on a fixed, small set of object parts, our 3D variant of BB yields hundreds of correspondences on the level of individual local features, making them applicable to challenging tasks such as 3D reconstruction from a fine-grained category (Sec. 2.5.3). Similar to prior works in multi-view detection, we leverage 3D CAD models to generate synthetic training data [110, 208, 189, 142].

## 2.3 3D Object Representations

In the following, we describe the design of our 3D object representations for fine-grained categorization. Both are based on estimating the 3D geometry of an object under consideration (Sec. 2.3.2), and represent both the appearance of individual local features (Sec. 2.3.3) and their locations (Sec. 2.3.4) in 3D object space. While our representations are based on state-of-the-art 2D representations that have proven to be effective for recognition (SPM [185], BB [46]), we effectively lift them to 3D by exchanging the underlying parameterization of feature extraction and spatial pooling, leading to improved performance (Sec. 2.5).

### 2.3.1 2D Base Representations

We start with a brief review of our 2D base representations, spatial pyramids (SPM) and bubble bank (BB). The **spatial pyramid** (SPM) representation [102, 185] is an extension of the bag-of-visual-words paradigm [41], enriching local appearance features with spatial information by pooling them with respect to spatial grids of varying resolution, concatenating the result into a large feature vector. We consider SPM in combination with locality-constrained linear coding (LLC [185]), which is considered state-of-the-art for a wide variety of image classification benchmarks.

The **BubbleBank** (BB) [46] representation is based on the notion of "bubbles": feature templates that are convolved with an image in local search regions, where the regions are determined by the template's image location during training. These responses are max-pooled over the region to produce a single feature. The final representation consists of pooling responses of a large bank of bubbles over their respective search regions. In [46], bubbles were defined manually via crowdsourcing, but also deliver state-of-the-art performance even when randomly sampled from training data, in a manner similar to [197].

### 2.3.2 3D Geometry Estimation

The basis of our 3D object representations is given by estimating the 3D geometry of an object, providing a frame of reference for both our 3D appearance representation (Sec. 2.3.3) and our 3D spatial pooling (Sec. 2.3.4). Since we are focusing on rigid objects (cars), we can model rough 3D object geometry by a discrete set of exemplar 3D CAD models, which are readily available. The first stage in our fine-grained categorization pipeline (Fig. 2.2) consists of identifying one (or multiple) CAD model(s) that best fit the image. The matching between a 3D CAD model and a 2D image is implemented by a set of classifiers that have been trained to distinguish among CAD models and viewpoints.

**Synthetic training data.** In line with recent work in multi-view recognition [110, 142], we leverage 3D CAD models as a cheap and reliable source of synthetic training data. Crucially, this gives precise 3D coordinates that we can use to anchor our appearance and location representation and is entirely free of human intervention (in contrast to approaches relying on part annotations [56, 204, 115]). In our experiments (Sec. 2.5), our 3D geometry classifiers are trained from 41 CAD models of cars (Fig. 2.2(a)), rendered at 36 azimuths, 4 elevations, and against 10 random background variations, for a total of 59,040 synthetic images.

**3D Geometry classifiers.** In order to match 3D CAD models to 2D images, we train a massive bank of classifiers for nearly the entire cross product of CAD models and viewpoints. In practice, we found it sufficient to group CAD models belonging to the same coarse category together (for cars, we define sedan, SUV, coupe, convertible, pickup, hatchback, and station wagon as coarse categories), and train a bank of viewpoint-dependent classifiers for all of them, resulting in 1,008 possible combinations of viewpoints and coarse categories. All classifiers are based on HOG [42] features in connection with a one-vs-all linear SVM. Empirically, we found that exemplar SVMs [118] did not result in a significant improvement but where computationally much more demanding.

**Multiple hypotheses.** An incorrect estimation of 3D geometry leads to errors in later stages of the fine-grained categorization pipeline that are hard to recover from. Thus, rather than commit to a single viewpoint and coarse category, we maintain a list of the top $N$ estimates, and max-pool features across all of them. Fig. 2.5(a) verifies the positive effect on performance.

### 2.3.3 3D Appearance Representation

The goal of our 3D appearance representation is to ensure that a discriminative local feature is represented only once, as opposed to requiring multiple representations in different viewpoints. Making these connections across viewpoints is important in order to generalize to test cases that have been observed from viewpoints not present

in the training data. We achieve this through an appearance representation that is (to an extent) viewpoint invariant, by transforming local image patches into a unified frame of reference prior to feature computation.

**Patch sampling.** The basis of our 3D appearance representation is given by a dense sampling of image patches that we extract from a given training or test image. In contrast to existing 2D representations, we sample patches directly from the 3D surface of the object of interest relative to its estimated 3D geometry (Sec. 2.3.2). In particular, we precompute thousands of uniformly spaced patch locations on the surface of our CAD models by dart throwing [39]. Each patch location comes with an associated surface normal and upward direction, determining its 3D orientation, and a flat, planar rectangle, determining its support region (Fig. 2.2(b)). For feature extraction, we project all patches visible from the estimated viewpoint into the image, resulting in a set of perspectively distorted quadrilaterals.

**Patch rectification and descriptors.** Prior to feature computation, we rectify the projected quadrilaterals to a common reference rectangle, effectively compensating for perspective projection. While this applies only to locally planar surfaces in theory, it typically results in correctly rectified patches also for curved surfaces, such as the car grille in Fig. 2.1 and the patches in Fig. 2.2(c). We densely sample RootSIFT [12, 116] descriptors on each rectified patch.

### 2.3.4 3D Spatial Pooling

The goal of our 3D spatial pooling is to characterize the position of local features with respect to the 3D geometry of an object. Like 3D appearance (Sec. 2.3.3), we utilize our 3D geometry estimate (Sec. 2.3.2) as the basis.

**3D Spatial Pyramid (SPM-3D).** After patch extraction we have a set of rectified patches with corresponding 3D locations. As in the case of 2D SPM, we can extract descriptors from each of these patches and quantize them using a trained codebook. However, unlike a standard 2D SPM, which only considers the 2D location of each

patch, our representation includes the location of each patch in 3D object coordinates, allowing us to pool over a more relevant space. Specifically, we partition the surface of the object based on azimuth and elevation relative to the center of the CAD model, *i.e.* we partition the space $\mathcal{S} = [0, 2\pi] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$, as visualized in Fig. 2.2(d), and pool quantized descriptors accordingly. As in 2D SPM, we use multiple scales, *i.e.* we pool over $1 \times 1$, $2 \times 2$, and $4 \times 4$ partitions of $\mathcal{S}$.

**3D BubbleBank (BB-3D).** Similar to our lifting of 2D SPM to 3D, we lift 2D BubbleBank (BB) to 3D by converting pooling regions from 2D to 3D. After extracting descriptors for each of the rectified patches, we convolve the descriptors with a set of bubbles obtained randomly over the training set. Crucially, each of the patches and bubbles has an associated 3D location, so we can pool over all patches within a 3D region of the bubble, illustrated in Fig. 2.2(e). By pooling over regions of sufficient size, additional robustness w.r.t. 3D geometry estimation is obtained. Our approach contrasts with the 2D equivalent [46] in that we do not rely on a feature appearing in the same 2D location within an image during both training and test, but rather at the same location with respect to 3D object geometry.

### 2.3.5 Classification with 3D Representations

We combine our 3D object representations (SPM-3D, BB-3D) with linear SVM classifiers that we train in a one-vs-all fashion for fine-grained categorization, in analogy to their 2D counterparts (SPM, BB), allowing us to pinpoint performance differences to the respective representations.

## 2.4 Novel Fine-Grained Dataset

In order to provide a suitable test bed for our 3D representations, we have collected a challenging, large-scale dataset of car models, released at `http://ai.stanford.edu/~jkrause/cars/car_dataset.html`. It consists of BMW-10, a small, ultra-fine-grained set of 10 BMW sedans (512 images) hand-collected by the authors, plus

Figure 2.3: One image each of 196 of the 197 classes in car-197 and each of the 10 classes in BMW-10.

car-197, a large set of 197 car models (16,185 images) covering sedans, SUVs, coupes, convertibles, pickups, hatchbacks, and station wagons. Since dataset collection proved non-trivial, we give the most important challenges and insights.

**Identifying visually distinct classes.** Since cars are manmade objects whose class list changes on a yearly basis, and models of cars do not have a different appearance from year to year, no simple list of visually distinct cars exists which we can use as a base. We thus first crawl a popular car website for a list of all types of cars made since 1990. We then apply an aggressive deduplication procedure, based on perceptual hashing [202], to a limited number of provided example images for these classes, determining a subset of visually distinct classes, from which we sample 197.

**Finding candidate images.** Candidate images for each class were collected from Flickr, Google, and Bing. To reduce annotation cost and ensure diversity in the data, the candidate images for each class were deduplicated using the same perceptual hash algorithm [202], leaving a set of several thousand candidate images for each of the 197 target classes. These images were then put on Amazon Mechanical Turk (AMT) in order to determine whether they belong to their respective target classes.

**Training annotators.** The main challenge in crowdsourcing the collection of a fine-grained dataset is that workers are typically non-experts. To compensate, we implemented a qualification task (a set of particularly hard examples of the actual annotation task) and provide a set of positive and negative example images for the car class a worker is annotating, drawing the negative examples from classes known

a priori to be similar to the target class.

**Modeling annotator reliability.** Even after training, workers differ in quality by large margins. To tackle this problem, we use the Get Another Label (GAL) system [82], which simultaneously estimates the probability a candidate image belongs to its target class and determines a quality level for each worker. Candidate images whose probability of belonging to the target class exceeds a specified threshold are then added to the set of images for that category. After obtaining images for each of the 197 target classes, we collect a bounding box for each image via AMT, using a quality-controlled system provided to us by the authors of [165]. Finally, an additional stage of deduplication is performed on the images when cropped to their bounding boxes. Fig. 2.3 shows example dataset images.

**Class Lists** In Tab. 2.1 we give the classes and number of images in each class for BMW-10. In Tab. 2.2 we do the same for car-197. A coarse category distribution for car-197 is given in Fig. 2.4. The training and test splits for each dataset are fixed and have equal size.

| Class | Num. Images |
|---|---|
| BMW 3 Series Sedan 2007 | 53 |
| BMW 3 Series Sedan 2009 | 53 |
| BMW 3 Series Sedan 2012 | 50 |
| BMW 5 Series Sedan 2007 | 50 |
| BMW 5 Series Sedan 2008 | 52 |
| BMW 5 Series Sedan 2011 | 50 |
| BMW M3 Sedan 2008 | 51 |
| BMW M5 Sedan 2007 | 52 |
| BMW ActiveHybrid 5 Sedan 2011 | 50 |
| BMW Alpina B7 Sedan 2011 | 51 |

Table 2.1: Class list and image count for BMW-10

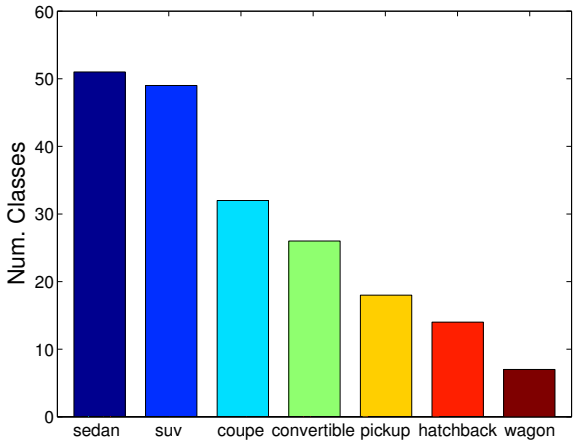



Figure 2.4: The distribution of coarse types in car-197.

## 2.5 Experiments

In the following, we carefully analyze the performance of our 3D object representations for a variety of different tasks, highlighting both their discriminative power for categorization and their unique ability to provide precise, point-wise correspondences between largely different views of the same object or even different instances of the same fine-grained category. First, we consider the task of fine-grained categorization of cars on various datasets (Sec. 2.5.1), reporting superior performance of our 3D object representations (SPM-3D, BB-3D) in comparison to both their respective 2D counterparts (SPM, BB) and state-of-the-art baselines [42, 21, 164, 185, 46]. Second, we successfully estimate the change in camera pose between multiple views of the same object (ultra-wide baseline matching, Sec. 2.5.2). And third, we give first promising results on reconstructing partial 3D models from fine-grained category instances (Sec. 2.5.3).

**Implementation Details.** Cropped images are used, as is standard in fine-grained classification, and are scaled such that the maximum dimension is 300 pixels. KDES [21] uses only grayscale kernel descriptors SPM uses codebook size 4096 and 3 layers of SPM (1x1, 2x2, and 4x4). BB uses 10k and 20k bubbles for small-scale and large-scale experiments, respectively, and a pooling region of 50% of the image height and

| Class | Num. Images | Class | Num. Images | Class | Num. Images |
|---|---|---|---|---|---|
| AM General Hummer SUV | 89 | Chevrolet Malibu Hybrid Sedan 2010 | 77 | Hyundai Tucson SUV 2012 | 87 |
| Acura RL Sedan 2012 | 64 | Chevrolet TrailBlazer SS 2009 | 80 | Hyundai Veracruz SUV 2012 | 84 |
| Acura TL Sedan 2012 | 86 | Chevrolet Silverado 2500HD Regular Cab 2012 | 76 | Hyundai Sonata Hybrid Sedan 2012 | 67 |
| Acura TL Type-S 2008 | 84 | Chevrolet Silverado 1500 Classic Extended Cab 2007 | 85 | Hyundai Elantra Sedan 2007 | 84 |
| Acura TSX Sedan 2012 | 81 | Chevrolet Express Van 2007 | 70 | Hyundai Accent Sedan 2012 | 48 |
| Acura Integra Type R 2001 | 89 | Chevrolet Monte Carlo Coupe 2007 | 90 | Hyundai Genesis Sedan 2012 | 87 |
| Acura ZDX Hatchback 2012 | 78 | Chevrolet Malibu Sedan 2007 | 89 | Hyundai Sonata Sedan 2012 | 79 |
| Aston Martin V8 Vantage Convertible 2012 | 90 | Chevrolet Silverado 1500 Extended Cab 2012 | 87 | Hyundai Elantra Touring Hatchback 2012 | 85 |
| Aston Martin V8 Vantage Coupe 2012 | 82 | Chevrolet Silverado 1500 Regular Cab 2012 | 88 | Hyundai Azera Sedan 2012 | 84 |
| Aston Martin Virage Convertible 2012 | 66 | Chrysler Aspen SUV 2009 | 87 | Infiniti G Coupe IPL 2012 | 68 |
| Aston Martin Virage Coupe 2012 | 76 | Chrysler Sebring Convertible 2010 | 81 | Infiniti QX56 SUV 2011 | 65 |
| Audi RS 4 Convertible 2008 | 73 | Chrysler Town and Country Minivan 2012 | 75 | Isuzu Ascender SUV 2008 | 80 |
| Audi A5 Coupe 2012 | 82 | Chrysler 300 SRT-8 2010 | 97 | Jaguar XK XKR 2012 | 93 |
| Audi TTS Coupe 2012 | 85 | Chrysler Crossfire Convertible 2008 | 86 | Jeep Patriot SUV 2012 | 88 |
| Audi R8 Coupe 2012 | 86 | Chrysler PT Cruiser Convertible 2008 | 90 | Jeep Wrangler SUV 2012 | 86 |
| Audi V8 Sedan 1994 | 87 | Daewoo Nubira Wagon 2002 | 90 | Jeep Liberty SUV 2012 | 89 |
| Audi 100 Sedan 1994 | 81 | Dodge Caliber Wagon 2012 | 81 | Jeep Grand Cherokee SUV 2012 | 90 |
| Audi 100 Wagon 1994 | 85 | Dodge Caliber Wagon 2007 | 84 | Jeep Compass SUV 2012 | 85 |
| Audi TT Hatchback 2011 | 81 | Dodge Caravan Minivan 1997 | 87 | Lamborghini Reventon Coupe 2008 | 72 |
| Audi S6 Sedan 2011 | 92 | Dodge Ram Pickup 3500 Crew Cab 2010 | 85 | Lamborghini Aventador Coupe 2012 | 87 |
| Audi S5 Convertible 2012 | 84 | Dodge Ram Pickup 3500 Quad Cab 2009 | 88 | Lamborghini Gallardo LP 570-4 Superleggera 2012 | 71 |
| Audi S5 Coupe 2012 | 85 | Dodge Sprinter Cargo Van 2009 | 79 | Lamborghini Diablo Coupe 2001 | 89 |
| Audi S4 Sedan 2012 | 79 | Dodge Journey SUV 2012 | 88 | Land Rover Range Rover SUV 2012 | 85 |
| Audi S4 Sedan 2007 | 90 | Dodge Dakota Crew Cab 2010 | 82 | Land Rover LR2 SUV 2012 | 85 |
| Audi TT RS Coupe 2012 | 79 | Dodge Dakota Club Cab 2007 | 77 | Lincoln Town Car Sedan 2011 | 78 |
| BMW ActiveHybrid 5 Sedan 2012 | 68 | Dodge Magnum Wagon 2008 | 80 | MINI Cooper Roadster Convertible 2012 | 73 |
| BMW 1 Series Convertible 2012 | 71 | Dodge Challenger SRT8 2011 | 78 | Maybach Landaulet Convertible 2012 | 58 |
| BMW 1 Series Coupe 2012 | 82 | Dodge Durango SUV 2012 | 87 | Mazda Tribute SUV 2011 | 72 |
| BMW 3 Series Sedan 2012 | 85 | Dodge Durango SUV 2007 | 91 | McLaren MP4-12C Coupe 2012 | 88 |
| BMW 3 Series Wagon 2012 | 83 | Dodge Charger Sedan 2012 | 82 | Mercedes-Benz 300-Class Convertible 1993 | 96 |
| BMW 6 Series Convertible 2007 | 88 | Dodge Charger SRT-8 2009 | 84 | Mercedes-Benz C-Class Sedan 2012 | 91 |
| BMW X5 SUV 2007 | 83 | Eagle Talon Hatchback 1998 | 92 | Mercedes-Benz SL-Class Coupe 2009 | 73 |
| BMW X6 SUV 2012 | 84 | FIAT 500 Abarth 2012 | 55 | Mercedes-Benz E-Class Sedan 2012 | 87 |
| BMW M3 Coupe 2012 | 89 | FIAT 500 Convertible 2012 | 67 | Mercedes-Benz S-Class Sedan 2012 | 89 |
| BMW M5 Sedan 2010 | 82 | Ferrari FF Coupe 2012 | 84 | Mercedes-Benz Sprinter Van 2012 | 82 |
| BMW M6 Convertible 2010 | 82 | Ferrari California Convertible 2012 | 78 | Mitsubishi Lancer Sedan 2012 | 95 |
| BMW X3 SUV 2012 | 77 | Ferrari 458 Italia Convertible 2012 | 79 | Nissan Leaf Hatchback 2012 | 84 |
| BMW Z4 Convertible 2012 | 81 | Ferrari 458 Italia Coupe 2012 | 85 | Nissan NV Passenger Van 2012 | 77 |
| Bentley Continental Supersports Convertible 2012 | 73 | Fisker Karma Sedan 2012 | 87 | Nissan Juke Hatchback 2012 | 88 |
| Bentley Arnage Sedan 2009 | 78 | Ford F-450 Super Duty Crew Cab 2012 | 83 | Nissan 240SX Coupe 1998 | 92 |
| Bentley Mulsanne Sedan 2011 | 71 | Ford Mustang Convertible 2007 | 89 | Plymouth Neon Coupe 1999 | 88 |
| Bentley Continental GT Coupe 2012 | 69 | Ford Freestar Minivan 2007 | 88 | Porsche Panamera Sedan 2012 | 87 |
| Bentley Continental GT Coupe 2007 | 92 | Ford Expedition EL SUV 2009 | 89 | Ram C/V Cargo Van Minivan 2012 | 82 |
| Bentley Continental Flying Spur Sedan 2007 | 89 | Ford Edge SUV 2012 | 86 | Rolls-Royce Phantom Drophead Coupe Convertible 2012 | 61 |
| Bugatti Veyron 16.4 Convertible 2009 | 65 | Ford Ranger SuperCab 2011 | 84 | Rolls-Royce Ghost Sedan 2012 | 77 |
| Bugatti Veyron 16.4 Coupe 2009 | 87 | Ford GT Coupe 2006 | 91 | Rolls-Royce Phantom Sedan 2012 | 88 |
| Buick Regal GS 2012 | 70 | Ford F-150 Regular Cab 2012 | 85 | Scion xD Hatchback 2012 | 83 |
| Buick Rainier SUV 2007 | 85 | Ford F-150 Regular Cab 2007 | 90 | Smart fortwo Convertible 2012 | 80 |
| Buick Verano Sedan 2012 | 75 | Ford Focus Sedan 2007 | 90 | Spyker C8 Convertible 2009 | 90 |
| Buick Enclave SUV 2012 | 84 | Ford E-Series Wagon Van 2012 | 75 | Spyker C8 Coupe 2009 | 85 |
| Cadillac CTS-V Sedan 2012 | 86 | Ford Fiesta Sedan 2012 | 85 | Suzuki Aerio Sedan 2007 | 76 |
| Cadillac SRX SUV 2012 | 82 | GMC Terrain SUV 2012 | 83 | Suzuki Kizashi Sedan 2012 | 92 |
| Cadillac Escalade EXT Crew Cab 2007 | 89 | GMC Savana Van 2012 | 66 | Suzuki SX4 Hatchback 2012 | 84 |
| Chevrolet Silverado 1500 Hybrid Crew Cab 2012 | 80 | GMC Yukon Hybrid SUV 2012 | 85 | Suzuki SX4 Sedan 2012 | 81 |
| Chevrolet Corvette Convertible 2012 | 79 | GMC Acadia SUV 2012 | 89 | Tesla Model S Sedan 2012 | 77 |
| Chevrolet Corvette ZR1 2012 | 93 | GMC Canyon Extended Cab 2012 | 80 | Toyota Sequoia SUV 2012 | 77 |
| Chevrolet Corvette Ron Fellows Edition Z06 2007 | 75 | GMC Savana Cargo Van 2012 | 70 | Toyota Camry Sedan 2012 | 87 |
| Chevrolet Traverse SUV 2012 | 88 | Geo Metro Convertible 1993 | 89 | Toyota Corolla Sedan 2012 | 87 |
| Chevrolet Camaro Convertible 2012 | 89 | HUMMER H3T Crew Cab 2010 | 78 | Toyota 4Runner SUV 2012 | 81 |
| Chevrolet HHR SS 2010 | 73 | HUMMER H2 SUT Crew Cab 2009 | 87 | Volkswagen Golf Hatchback 2012 | 86 |
| Chevrolet Impala Sedan 2007 | 86 | Honda Odyssey Minivan 2012 | 84 | Volkswagen Golf Hatchback 1991 | 92 |
| Chevrolet Tahoe Hybrid SUV 2012 | 74 | Honda Odyssey Minivan 2007 | 82 | Volkswagen Beetle Hatchback 2012 | 85 |
| Chevrolet Sonic Sedan 2012 | 88 | Honda Accord Coupe 2012 | 78 | Volvo C30 Hatchback 2012 | 83 |
| Chevrolet Express Cargo Van 2007 | 59 | Honda Accord Sedan 2012 | 77 | Volvo 240 Sedan 1993 | 91 |
| Chevrolet Avalanche Crew Cab 2012 | 90 | Hyundai Veloster Hatchback 2012 | 82 | Volvo XC90 SUV 2007 | 86 |
| Chevrolet Cobalt SS 2010 | 83 | Hyundai Santa Fe SUV 2012 | 84 | | |

Table 2.2: Class list and image count for car-197

the entire image width. For our 3D object representations, we use 3 viewpoint/coarse category hypotheses and extract patches corresponding to 3 CAD models for each such hypothesis. On BMW-10 and BMW-10 (flipped), only sedan CAD models are used. For SPM-3D, we use a codebook of size 4096. For BB-3D, we use 10k bubbles at small-scale and 20k bubbles at large scale. Features for BB and BB-3D use a power-scaling [46] parameter of 16. BB-3D-G refers to pooling bubble responses globally, and BB-3D-L pools across the width and length of the car, but only 25% of its height. SPM-3D-L uses 1x1, 2x2, and 4x4 partitions of azimuth-elevation space, with SPM-3D-G using only a 1x1 partition. In all cases we train one-vs-all $L_2$-regularized $L_2$-loss SVMs, selecting a regularization parameter $C$ from the range $10^{-6}, 10^{-5}, \ldots, 10^{10}$ by 25-fold cross-validation for small-scale experiments and use a constant $C$ value $10^{10}$ for large-scale experiments.

As input into our 3D geometry classifier, we extracted HOG [42] features on a fixed grid of size $340 \times 192$ with a spacing of 16 pixels. We use the implementation of HOG given by [57]. The 3D geometry classifier was trained using an $L_2$-regularized, $L_2$-loss SVM from the LibLinear library [54]. The values of $C$ used were $10^{-6}, 10^{-5}, \ldots, 10^{3}$. The $C$ value used was selected based on minimizing 0-1 error with respect to getting the azimuth prediction within 30 degrees of the ground truth azimuth on BMW-10, where azimuths were hand-annotated. For each geometry (CAD model) hypothesis, 4k patches were extracted on car-types, 2k patches were extracted on BMW-10, and 1k patches were extracted on car-197. All extracted patches were selected to be visible based on the predicted viewpoint. Patches for which the dot product of the patch normal vector and the vector from the patch center to the camera exceeds a fixed threshold of 0.15 were not selected. The projection of each 3D patch was rectified to the size $36 \times 36$, though patches which were not entirely visible (*e.g.* due to being near the edge of the image and therefore potentially cut off) were only rectified to fill as much of the $36 \times 36$ patch as is visible. If enough (50%) of the 3D patch's projection were not visible, however, the patch was similarly not selected. Thus each of the extracted patches is visible from a clean viewing angle.

We use the VLFeat[175] implementation of SIFT [116], using a piecewise-flat windowing function, which dramatically speeds up descriptor computation. For BMW-10 and car-197, descriptors are computed on the rectified patches every 4 pixels with a patch size of 16. For car-types, descriptors are extracted every 2 pixels with a patch size of 16.

Pooled features for SPM and the SPM-3D methods are $L_2$-normalized for all datasets, as is standard. On BMW-10 and car-types, features for BB and BB-3D are normalized to zero mean and unit variance. For car-197, features for all methods are normalized to fit in the range $[-1, 1]$, which we have found can substantially improve SVM training time.

(a)

| | HOG [42] | PB(mvDPM) [164] | structDPM [164] | SPM [185] | SPM-3D-L **(ours)** | BB [46] | BB-3D-G **(ours)** |
|---|---|---|---|---|---|---|---|
| car-types | 77.5 | 85.3 | 93.5 | 84.5 | 85.7 | 92.6 | **94.5** |

(b)

| | HOG [42] | PB(mvDPM) [164] | KDES [21] | SPM [185] | SPM-3D-G **(ours)** | SPM-3D-L **(ours)** | BB [46] | BB-3D-G **(ours)** | BB-3D-L **(ours)** |
|---|---|---|---|---|---|---|---|---|---|
| BMW-10 | 28.3 | 29.1 | 46.5 | 52.8 | 58.3 | 58.7 | 58.7 | **66.1** | 64.7 |
| BMW-10 (flipped) | - | - | - | 66.1 | - | 67.3 | 69.3 | **76.0** | - |

Table 2.3: (a) Comparison to state-of-the-art on *car-types* [164]. (b) In-depth analysis on our *BMW-10* dataset.

### 2.5.1 Fine-Grained Categorization

We commence our evaluation by comparing the performance of our 3D object representations, their respective 2D counterparts, and state-of-the-art baselines for the task of fine-grained categorization. To that end, we consider three different datasets of varying granularity. i) We use the existing car-types dataset [164], consisting of 14 car classes from a variety of coarse categories (sedans, hatchbacks, sedans, SUVs, convertibles), to establish that our methods outperform the previous state-of-the-art in car classification. ii) We provide an in-depth analysis of our methods in comparison to their respective 2D counterparts on BMW-10, our ultra-fine-grained set of 10 BMW sedans. iii) We demonstrate the ability of our methods to scale up to hundreds of fine-grained classes on our large-scale dataset (see Sec. 2.4).

**i) Car-types.** Tab. 2.3(a) gives classification accuracy for our methods SPM-3D, BB-3D-G, and prior work. Curiously, the performance on this dataset seems to have almost saturated, with the weakest prior method (a simple HOG template) achieving 77.5% and the strongest (structDPM, a multi-class DPM [57]) achieving 93.5% accuracy. We believe this high level of performance to indicate a rather coarse granularity of this dataset, which is reinforced by the two strongest prior methods (PB(mvDPM), 85.3%, structDPM, 93.5%) being based on part-layout information rather than discriminative local features (such as SPM, 84.5%). In comparison, our method BB-3D-G (94.5%) outperforms the best reported prior result of structDPM (93.5%) by 1%. In addition, unlike structDPM, it scales effortlessly to large-scale

datasets such as car-197, since it does not rely on joint regularization across all classes. Comparing 3D to 2D, BB-3D-G outperforms BB (92.6%) by 1.9%, and SPM-3D (85.7%) beats SPM by 1.2%.

**ii) BMW-10.** Tab. 2.3(b) gives the results for our ultra-fine-grained dataset of 10 BMW sedans, focusing on different variants of our methods SPM-3D and BB-3D and their respective 2D counterparts, and adding KDES [21] to the state-of-the-art baselines. We make the following observations: first, the general level of performance is significantly lower than for car-types (HOG achieves an accuracy of 28.3%, PB(mvDPM) 29.1%), which indicates that our dataset is significantly more fine-grained. Second, as a result, methods relying on discriminative local features rather than a global feature template (HOG) or part layout (PB(mvDPM)) perform much better (KDES 46.5%, SPM 52.8%, BB 58.7%). Third, our 3D object representations improve significantly over their respective 2D counterparts: SPM-3D-L (58.7%) improves over SPM by 5.9% and BB-3D-G (66.1%) improves over BB by 7.4%.

In Tab. 2.3(b), we also investigate the impact of enriching the original set of training images by flipped versions of each image, effectively doubling the amount of training data and increasing the density with which different object viewpoints are represented. Performance improves significantly for all methods, by 8.6% (SPM-3D-L), 9.9% (BB-3D-G), 10.6% (BB), and 13.3% (SPM). Notice that while the 2D methods benefit more from adding training data, the relative ordering of results between different methods is consistent with BMW-10 without flipping: SPM-3D-L (67.3%) outperforms SPM by 1.2%, and BB-3D-G (76.0%) outperforms BB by 6.7%. Fig. 2.6(a) visualizes the discriminative power of each of the 10k bubbles of BB-3D-G on BMW-10 (flipped). Each point is a 3D bubble location, with its size and hue proportional to $\sum_{j=1}^{10} |w_{i,j}|^5$, where $w_{i,j}$ is the SVM weight on feature $i$ for class $j$. This indicates that discriminative features are primarily at the front and back of cars, which is both correct and human-interpretable.

**Global vs. local pooling.** In Tab. 2.3(b), we further examine the impact of the size of the 3D pooling region on performance, distinguishing global pooling (SPM-3D-G, BB-3D-G) and local pooling (SPM-3D-L, BB-3D-L). For SPM-3D, local pooling improves performance slightly by 0.4%. For BB-3D, global pooling is 1.4% better, beating the next best result by 6%. We believe this counterintuitive superiority of BB-3D-G over BB-3D-L to be due to 1) mispredictions in viewpoint, and 2) the strong left-right symmetry of cars: it can help to look on the right side of a car for a bubble originally found on the left side. On the basis of these results all other experiments have used SPM-3D-L and BB-3D-G.

**Amount of training data.** Fig. 2.5(b) plots classification accuracy (y-axis) of the two best performing 2D and 3D methods of Tab. 2.3(b) versus the number of training images, which we vary from 1 to 16 in powers of two (we also add the results on all of BMW-10 and BMW-10 (flipped) as the two right-most points for reference). For each experiment, we randomly sample training images and fix them for both methods. We observe the 3D representation to outperform its 2D counterpart consistently for all numbers of training images. The difference is most pronounced for 16 training images (9.8%) and decreases for larger numbers (to 7.4% and 6.7%, respectively), indicating that the 3D representation can utilize training data more effectively.

**iii) Car-197.** Tab. 2.4(top) gives the results for our large-scale dataset of 197 fine-grained car categories, again comparing SPM-3D-L and BB-3D-G to SPM and BB. Surprisingly, SPM performs remarkably well with the increased data (69.5%), beating our best 3D representation BB-3D-G (67.6%) by 1.9%. Similarly to on the car-types, BMW-10, and BMW-10 (flipped) datasets, BB-3D-G outperforms the 2D version BB (63.6%) by 4%, and SPM-3D-L (65.6%) by 2%. Analyzing these results, we believe our 3D representations to suffer from certain coarse categories (*e.g.* large pickup trucks) being underrepresented in our 3D CAD models, which is a shortcoming of this particular experimental setup, not a limitation of our methods. Stacking the four methods together naturally results in the best performance, 75.5%, indicating that our 3D representations encode information that is complementary to their 2D

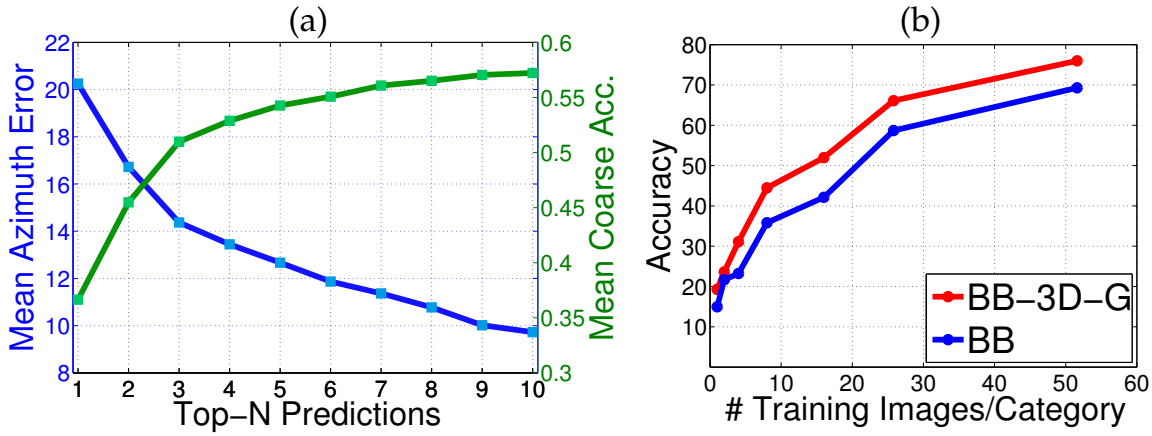



Figure 2.5: (a) Viewpoint/coarse category acc. over top-$N$ predictions. (b) Accuracy vs. number of training images.

counterparts, despite using the same base descriptors.

**Summary.** We conclude that lifting 2D object representations to 3D is in fact beneficial for all except one case for both SPM and BB, leading to significant improvements in classification performance over state-of-the-art on existing (car-types) and ultra-fine-grained (BMW-10) datasets.

### 2.5.2 Ultra-Wide Baseline Matching

Characterizing the relation between different views of the same visual scene is one of the most important tasks in computer vision, providing the basis for 3D reconstruction using structure-from-motion techniques [71]. For known intrinsic camera parameters, it can be phrased as estimating the fundamental matrix, based on putative point-to-point correspondences between the views. Ultra-wide baseline matching has been suggested [208] as a way to quantify the ability of a method to localize corresponding 3D points across viewpoints, specifically for wide baselines between 45° and 180° for which pure local feature-based methods such as SIFT [116] typically fail.

**Methodology.** We follow the protocol of [208] and perform ultra-wide baseline matching on 134 image pairs of the *3D Object Classes* dataset [152]. We modify BB-3D-L to find putative correspondences for pairs of images as follows: for a given pair, each patch in the first image defines a bubble in the second image, and is convolved with all patches of the second image that fall into the bubble's 3D pooling region (0.1 of the object surface). For each bubble, the maximally responding patch is memorized, and possibly kept as a putative correspondence after thresholding. A fundamental matrix relating both images is then computed using standard multi-view geometry [71] and RANSAC. Fig. 2.6(b) visualizes the results for 3 image pairs using BB-3D-L by depicting a random selection of epipolar lines and corresponding feature matches. Please note, although this method uses rough 3D geometry information to bootstrap patch rectification and local pooling (Sec. 2.3.2), it still mainly relies on the local evidence of the test image pair to establish feature-level correspondences (like SIFT). It can not be expected to deliver results for baselines larger than 135° (denoted "-" in Tab. 2.4), since no local evidence is shared between the views.

**Results.** Tab. 2.4 gives ultra-wide baseline matching results, where performance is measured as the fraction of fundamental matrices for which the Sampson error [71] w.r.t. ground truth correspondences falls under a threshold (20 pixels as in [208]). Rows of Tab. 2.4 correspond to different baselines (difference in azimuth angle between two views of the same object). The last two rows give averages over all baselines (Av.1, imputing zero accuracy for "-"), and over only the first two baselines that can be delivered by all methods (Av.2). Columns of Tab. 2.4 compare two variants of our BB-3D-L, BB-3D-S (using only a single coarse category and viewpoint prediction) and BB-3D-M (using multiple), with comparable local feature-based methods, SIFT [116] and the multi-view local part detectors of [208]. Although we can not expect to compete with the full-blown 3D shape model of [208] and the multi-view incarnation of DPM [57], 3D$^2$PM [142], that leverage global shape and have been explicitly designed and trained for viewpoint prediction on that dataset, we include their results as reference.

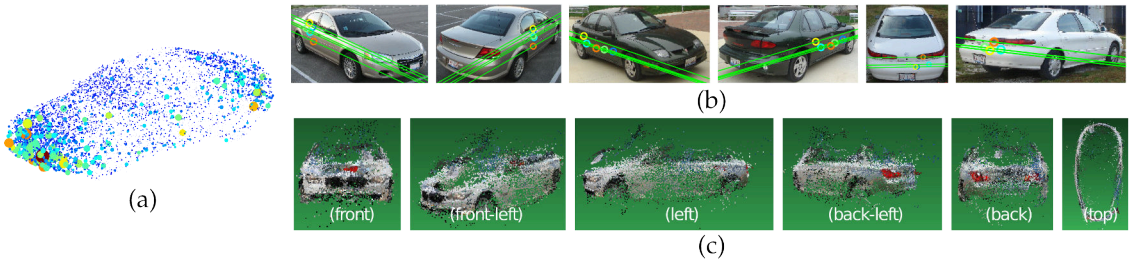



Figure 2.6: (a) Discriminative bubbles for BB-3D-G on BMW-10 (flipped). (b) Ultra-wide baseline matching on *3D Object Classes* [152] (green: epipolar lines, colored circles: corresponding inlier features). (c) Fully automatic 3D reconstruction from 46 images of a fine-grained category from BMW-10.

From the four left-most columns of Tab. 2.4, we observe that our 3D object representations outperform all local feature-based methods by a significant margin even for Av.1: SIFT fails catastrophically (0.5%), and local part detectors [208] (22%) are outperformed by both BB-3D-S (24.6%) and BB-3D-M (25.8%). This difference is even more pronounced when considering individual baselines (for 45°, BB-3D-M (71.1%) outperforms parts by 44.7%, for 90°, BB-3D-S (40%) outperforms parts by 13%) or Av.2 (BB-3D-M (51.6%) outperforms parts by 24.6%). While being not quite on par with the state-of-the-art results obtained by 3D$^2$PM (67.8%) overall, for 45° baseline, BB-3D-M in fact beats 3D$^2$PM by 13.2%, and typically provides hundreds of densely spaced inlier features as opposed to a sparse set of 20 object parts, which we will exploit for 3D reconstruction in Sec. 2.5.3. Using multiple coarse category models (BB-3D-M) improves over the single case (BB-3D-S) by 1.2% (Av.1) and 2.3% (Av.2), respectively, in step with the increased robustness shown in Fig. 2.5(a).

### 2.5.3 3D Fine-Grained Category Reconstruction

Having verified the ability of our BB-3D representation to establish accurate feature correspondences across different views of the same object (Sec. 2.5.2), we now move on to the even more challenging task of finding correspondences between examples of the same fine-grained category (but not the same instance). Note that this is feasible for cars: their 3D geometry is almost uniquely determined by fine-grained category

| | SPM [185] | SPM-3D-L **(ours)** | BB [46] | BB-3D-G **(ours)** | Stacked |
|---|---|---|---|---|---|
| car-197 | **69.5** | 65.6 | 63.6 | 67.6 | 75.5 |

| Bl. | SIFT [116] | Parts [208] | **BB-3D-S** | **BB-3D-M** | 3D Shape [208] | 3D$^2$PM [142] |
|---|---|---|---|---|---|---|
| 45° | 2% | 27% | 58.5% | **71.7%** | 55% | **58.5%** |
| 90° | 0% | 27% | **40%** | 31.4% | 60% | **77.1%** |
| 135° | - | 10% | - | - | 52% | **58.6%** |
| 180° | - | 24% | - | - | 41% | **70.6%** |
| Av.1 | 0.5% | 22% | 24.6% | **25.8%** | 52% | **66.4%** |
| Av.2 | 1% | 27% | 49.3% | **51.6%** | 57.5% | **67.8%** |

Table 2.4: Top: Results on car-197. Bottom: Ultra-wide baseline matching results on *3D Object Classes* [152].

affiliation, and our features are invariant to the remaining variation (such as color). At the same time, this task is very challenging, since the background varies drastically between different views, and can hence not provide any evidence for correspondence. To our knowledge, reconstruction of a fine-grained category has not been reported in the literature before.

**Methodology.** We run BB-3D-L for all pairs of images of a category to obtain putative correspondences, and feed feature locations and correspondences into VisualSFM, a front-end for multi-core bundle adjustment [188]. We run their SFM pipeline as a black box, using standard parameter settings except for increasing the number of iterations.

**Results.** Fig. 2.6(c) depicts qualitative results for the reconstruction of a fine-grained class of our BMW-10 dataset (2012 BMW ActiveHybrid 7 Sedan), rendered from different viewpoints. Please note that we have not applied any dense, pixelwise refinement of the sparse reconstruction. The point density is due to the high number of inlier correspondences generated by BB-3D-L. Clearly, the reconstruction is incomplete and contains spurious points, in particular for textureless regions (*e.g.* the hood) – however, the local 3D pooling of BB-3D-L successfully bounds the degree

to which correspondences can "drift away" on the 3D geometry. The resulting reconstruction has a sedan shape, and shows the characteristic grille, headlight, and rear light features of a BMW. We believe this result to be highly promising, opening a vast array of future research directions, such as high-fidelity reconstruction from fine-grained categories, or using the reconstructed model for recognition.

## 2.6 Conclusions

We have demonstrated that 3D object representations can be beneficial for fine-grained categorization of rigid classes, specifically cars. By lifting two state-of-the-art 2D object representations to 3D (SPM and BB), we obtained state-of-the-art performance on both existing (car-types) and our new datasets (BMW-10, car-197). In addition, we leveraged our BB-3D-L representation for ultra-wide baseline matching, and showed first promising results for the automatic reconstruction of 3D models from fine-grained category instances.

# Chapter 3

# Learning Features and Parts for Fine-Grained Recognition

This work addresses the problem of fine-grained recognition: recognizing subordinate categories such as bird species, car models, or dog breeds. We focus on two major challenges: learning expressive appearance descriptors and localizing discriminative parts. To this end, we propose an object representation that detects important parts and describes fine-grained appearances. The part detectors are learned in a fully unsupervised manner, based on the insight that images with similar poses can be automatically discovered for fine-grained classes in the same domain. The appearance descriptors are learned using a convolutional neural network. Our approach requires only image level class labels, without any use of part annotations or segmentation masks, which may be costly to obtain.

This project was joint work with Timnit Gebru, Jia Deng, Li-Jia Li, and Fei-Fei Li, and appeared in [92].

## 3.1 Introduction

Fine-grained recognition [73, 56, 141, 18, 196, 197, 51] refers to the task of distinguishing sub-ordinate categories such as bird species [186, 182], dog breeds [88], aircraft [117], or car models [164, 95]. It is one of the cornerstones of object recognition

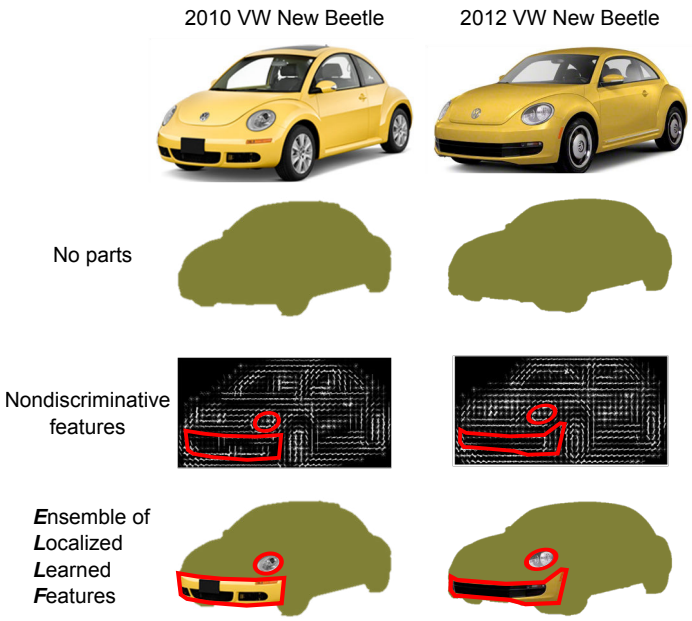



Figure 3.1: The key to fine-grained recognition is localizing important parts and representing part appearance discriminatively, as global cues describing the overall shape or color often cannot capture the subtle differences. Without any information at the level of parts it is difficult to differentiate between two very similar classes. Similarly, features such as HOG [42], which are not discriminative for fine-grained classes, do not contain enough information.

due to the potential to make computers rival human experts in visual understanding.

However, two major challenges need to be solved before fine-grained recognition can achieve this goal. First, recognizing fine-grained classes typically requires differentiating fine details in appearance. It calls for an appearance representation that retains details critical for discrimination and discards unnecessary information. The retained discriminative details can be very subtle and highly domain-specific. For example, the two cars in Fig. 3.1 differ at the front bumper and turning signal. Descriptors such as SIFT [116] or HOG [42], while successful in more coarse-grained recognition tasks, may not be discriminative enough to differentiate at this level of detail.

Another challenge is in discovering and locating the parts that contain discriminative details. When humans describe differences between fine-grained classes, we almost always point out the location ("vertical bars on the car's grille", "black patch

on the bird's beak"). That is, we locate the relevant object parts and then check the appearance. The Beetles in Fig. 3.1 are much easier to differentiate when told to explicitly look at the front bumper. This brings forward the issue of part discovery – which parts are discriminative and where are they? One possibility is to annotate the location of various parts by hand. However, this approach is costly and it may be difficult to scale up to handle many different types of fine-grained classes. We hypothesize that the ultimate solution to fine-grained recognition should entail both localizing important parts with minimal supervision and effectively describing their appearances in a way that does not discard information useful for classification.

In this paper we simultaneously tackle both feature learning and part discovery. Our central idea is *learning* both features and parts to form a unified object representation. Specifically, we use convolutional neural networks (CNNs) to learn appearance descriptors, and perform unsupervised part discovery to obtain a collection of part detectors. By learning the features appropriate to describe the object categories in question, we let the data determine which features are effective for discrimination, which helps avoid losing information useful for categorization. By keeping part discovery completely unsupervised with respect to part annotations, we aim to make our algorithm scalable to a variety of fine-grained domains, including ones for which it is not known a priori which parts are discriminative. In recognition time, we detect parts and represent their appearances using the learned features, leading to an "Ensemble of Localized Learned Features" (ELLF), a novel representation for fine-grained recognition. To our knowledge our approach is the first to integrate feature learning and unsupervised part discovery in fine-grained recognition.

## 3.2 Related Work

**Part-Based Representations.** Many fine-grained recognition techniques involve part-based representations inspired by work on generic object recognition [22, 57]. Some explicitly model pose [56, 204] whereas others use less structured approaches [18, 115, 62, 31]. Typically part detectors are learned using hand-annotated keypoints. Our approach departs from most prior work in that the part detectors are learned

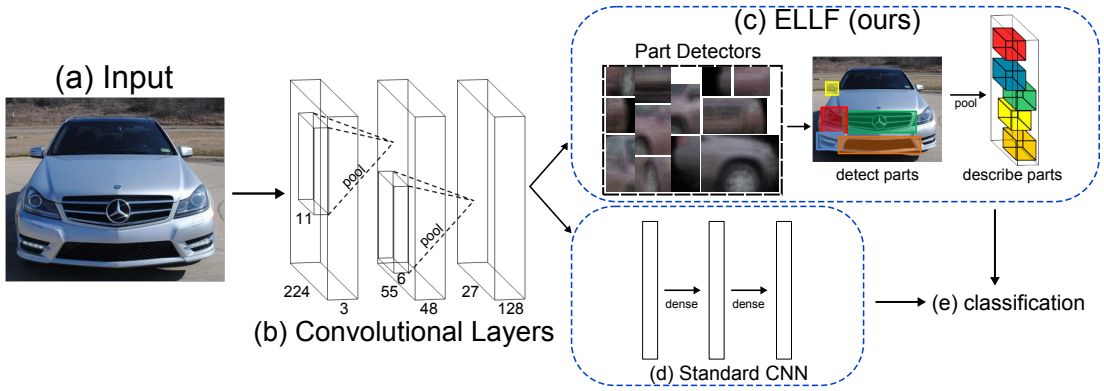



Figure 3.2: Overview of our Ensemble of Localized Learned Features (ELLF) representation. Given an input image (a), we detect parts using a collection of unsupervised part detectors (Sec. 3.3.3). We also feed the image into a convolutional neural network (CNN) (b) that outputs a grid of discriminative features (Sec. 3.3.2). The CNN is learned with class labels and then truncated, retaining the first two convolutional layers which retain spatial information. We describe the appearance of each detected part using the learned CNN features by pooling in the detected region of each part (c). Appearance of of any undetected parts is set to zero. This results in our ELLF representation that is then used to predict fine-grained object categories (e). In comparison, a standard CNN (d) passes the output of the convolutional layers through several fully connected layers in order to make a prediction.

with zero supervision, which means that we can tackle multiple domains, including ones for which nothing more than class labels and bounding boxes are available.

**Feature Learning.** Feature learning is a promising approach that can generate powerful appearance representations. Much work has focused on encoding low-level features such as SIFT or HOG (*e.g.* [185, 143, 196]) or mining discriminative templates [196, 197]. The recent success of convolutional neural networks [103, 97] on large-scale classification and face recognition [78, 79] demonstrates that powerful features can be learned directly from pixels. This inspires us to adopt convolutional neural networks (CNNs) for fine-grained recognition. Note that unlike the DeCAF system [49] that trains features on ImageNet [45], we do not perform any pre-training using additional data. This requires care when choosing the network architecture and necessitates using a larger variety of data deformations in order to cope with and

increase the size of the training set. To our knowledge this is the first time deep neural networks have been used for fine-grained recognition without any form of domain adaptation.

**Other Approaches.** In addition to the approaches outlined above, segmentation has also been found to be particularly useful [133, 9, 31, 30] in fine-grained recognition tasks. Another line of research focuses on putting humans in the loop [181, 46, 181, 139, 27]. These are complementary approaches that can be jointly used with our method, and we do not attempt to incorporate these additional cues in our work.

## 3.3 Approach

### 3.3.1 Overview

Our representation builds on the intuition that we need to localize parts and then compare their appearances. Fig. 3.2 provides an overview of the algorithm. The main idea is to have a representation that enables easy comparison of appearance features on corresponding parts. This leads to ELLF: Ensemble of Localized Learned Features. Suppose we have a collection of $n$ object parts with associated part detectors, which we assume for now have already been trained. Given an input image (Fig. 3.2(a)), let $a_i$ be the appearance of part $i$, as described by a convolutional neural network (Fig. 3.2(b)). The ELLF representation is then simply $(a_1, a_2, \ldots, a_n)$, the concatenation of part appearances (Fig. 3.2(c)). Note that due to view point change and occlusion, not all parts are necessarily detected. When part $i$ is not detected, the appearance $a_i$ is set to zero, preventing a classifier (Fig. 3.2(e)) from using any information at that part. With images represented by ELLF, we can then train classifiers such as linear SVMs to perform fine-grained classification. For us, the collection of parts is determined in an unsupervised framework and they are described using features from a convolutional neural network.

One desirable property of using ELLF is that it compares the appearances of each part and aggregates the similarities together. This is different from traditional

approaches for generic object recognition such as spatial pyramid matching (SPM) [102] where a linear kernel compares the appearances at the same *spatial location* instead of the same *part*. SPM is thus sub-optimal for objects of different poses, because all the parts are not necessarily visible or at the same location across images.

We would like to highlight one other difference between ELLF and traditional bag-of-words representations. A linear kernel defined on bag-of-words histograms or its softly quantized generalizations such as LLC [185] roughly corresponds to comparing the *presence* of each visual word. In this case we have already quantized the appearances into visual words and are only checking whether specific visual words occur. The subtle appearance differences for fine-grained classes might still get lost in quantization. In contrast, since we describe object parts using features trained from a CNN, our representation keeps rich appearance descriptors in the final representation.

Now that we have defined ELLF, we proceed to describe the process of generating ELLF. There are two key components: learning discriminative appearance features and discovering parts.

### 3.3.2 Feature Learning

A hallmark of fine-grained recognition is that it demands rich and expressive appearance descriptors, as traditional descriptors like SIFT [116] or HOG [42] may not capture the right balance between discriminativeness and invariance for fine-grained classes. To this end we adopt the philosophy of end-to-end training of feature descriptors using neural networks, allowing the descriptors to adapt to the idiosyncrasies of individual categories. To our knowledge this is the first time deep feature learning is applied on fine-grained recognition involving no pre-training with additional data. We demonstrate that even on relatively small datasets, feature learning can be effective for fine-grained recognition.

In particular, we use a convolutional neural network (CNN) [103] that accepts pixels as its input and outputs probabilities of classes. We modify the architecture of Krizhevsky *et al.* [97] to account for our smaller-scale data, which we have found to be very important for preventing overfitting. The network consists of two convolutional

layers followed by three fully connected layers with a softmax loss. Each convolutional layer performs convolutions with a bank of filters on the 3D input matrix and outputs filter responses in the form of a 3D matrix. Since filter parameters are learned from the data, the network has the potential to generate feature descriptors tailored to specific domains. More details are given in Sec. 3.4.1.

After training, we remove the fully connected layers and use the two convolutional layers as a generator of pixel-level appearance descriptors. Note that it is necessary to cut off the features at this point in order to maintain spatial information – features in the fully connected layers are completely unordered. To obtain a descriptor for a region (such as a bounding box given by a part detector), we perform max-pooling of the descriptors located inside the region. Thus, one way to interpret our parts is as movable pooling regions in a CNN architecture.

### 3.3.3 Part Discovery

The goal of part discovery is to obtain a collection of reliable part detectors. Our key contribution is a part discovery algorithm that is fully unsupervised. Previous work has relied on hand-annotated keypoints to train part detectors [18]. Here we bypass human annotations completely, which has the advantage of scaling to very large-scale datasets.

How can we train part detectors without any annotations? The key observation is that objects with the same pose can often be automatically discovered by local low-level cues. Aligning poses between images is, in general, a difficult problem, because appearance may vary wildly even within the same category. However, localizing parts primarily depends on an understanding of the overall object shape without the need to scrutinize the local details—a blurred image of a dog may prevent you from recognizing the breed but will likely hold enough information for you to localize the parts.

This intuition motivates our part discovery procedure. We first discover sets of aligned images with similar poses. Under the assumption that images within a set are well aligned, the same parts have similar locations across images. We can thus

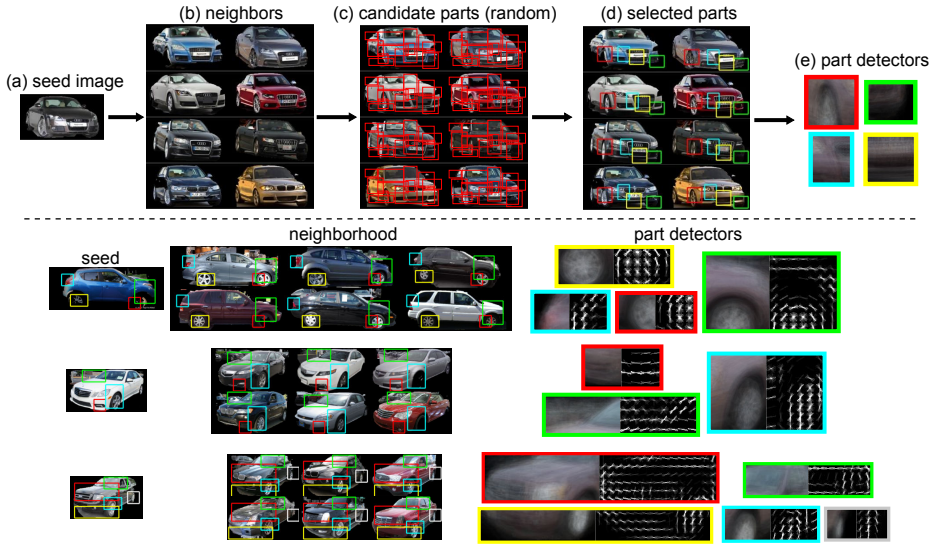



Figure 3.3: Top: Our fully unsupervised part discovery pipeline. We randomly sample a seed training image (a) and retrieve the nearest neighbors (b) in terms of global HOG appearance. This allows us to identify well aligned images with similar poses. From a large sample of random parts (c) we pick the sub-windows with high energy (d) as candidate parts, which are then used to train our final part detectors (e), visualized as the average of the patches used as positive examples in training. Bottom: Examples of parts discovered by our method. Leftmost is the seed image used to generate the set of aligned images, a subset of which is shown in the middle. Shown at the right are the average of the image patches used as positives to train each part detector and the learned weights. Our method is able to discover a variety of parts from each neighborhood.

train a part detector using the patches from the same spatial location as positive examples and patches from elsewhere as negative examples. Fig. 3.3 (top) illustrates this intuition. We now elaborate on the individual steps.

### Discovering Aligned Images

The first step is discovering sets of aligned images. We use a randomized algorithm. We pick a seed image at random (Fig. 3.3(a)) and then retrieve nearest neighbors in terms of HOG features, extracted at multiple scales. To help reduce the influence of the background, we perform GrabCut [149] before extracting HOG features, initializing the foreground model with the object's bounding box, which is typically

given in fine-grained recognition. These foreground segmentations are centered for the purpose of comparing across images. We repeat this process, randomly sampling multiple sets and using each set to generate multiple part detectors. With a reasonable number of training images to choose from, this method typically results in a set of images with nearly the same pose (Fig. 3.3(b)).

### Part Selection

Next we select the parts to detect, as every location within the segmented foreground can be a potential part. To address this issue, we randomly sample a large number of regions with various sizes as candidates (Fig. 3.3(c)). We then select the parts with the highest energy, as measured by the variance of HOG across images (Fig. 3.3(d)). This helps prevent selecting parts which lack discriminative information – a part which does not vary at all across images cannot be useful for discrimination. Each time we select a part, we remove from the candidate list any parts that overlap more than a fixed threshold $\rho$ with an already selected part, set to 15% in our implementation. This helps prevent learning redundant parts for a given set of aligned images.

### Detector Learning

We then learn a detector for each selected part (Fig. 3.3(e)). Specifically, let $I_j$ be the aligned images and $z^+$ be location of the selected part. Under the assumption that the images are well aligned, our learning objective for the part detector is finding a template $w$ that minimizes the hinge loss

$$\begin{aligned} \min_w \quad & \textstyle\sum_j \max\{0, 1 - w^T h(I_j, z^+) \\ & + \textstyle\sum_j \sum_{z_j^-} \max\{0, 1 + w^T h(I_j, z_j^-)\}, \end{aligned} \tag{3.1}$$

where $h(I_j, z^+)$ extracts features (HOG) on image $I_j$ at positive patch location $z^+$. The variable $z_j^-$ are the locations of the negative patches on image $I_j$, chosen randomly such that they do not overlap with the positive patch at location $z^+$.

We now relax the assumption that the images are well aligned to be robust to misalignment. Instead of having a fixed location $z^+$, we introduce a latent variable

$z_j^+$ to represent the true location of the part on image $z_j^+$. Our learning objective is thus

$$\begin{aligned} \min_w \quad & \textstyle\sum_j \max\{0, 1 - \max_{z_j^+} w^T h(I_j, z_j^+) \\ & + \textstyle\sum_j \sum_{z_j^-} \max\{0, 1 + w^T h(I_j, z_j^-)\}, \end{aligned} \tag{3.2}$$

where we search for the best match over all possible locations $z_j^+$. The objective can be optimized by alternating between optimizing $z_j^+$ with fixed $w$ and optimizing $w$ with fixed $z_j^+$, similar to the latent SVM optimization introduced in [57]. We initialize the latent variable $z_j^+$ with the original location $z^+$. Also similar to [57], we augment the HOG feature $h(I, z)$ with $(dx \cdot dy, dx^2, dy^2)$ to include a spatial prior that penalizes patches too far away from the original $z^+$. Here $dx = x_z - x_{z^+}$ and $dy = y_z - y_{z^+}$, where $(x_z, y_z)$ is the coordinate of the location $z$ and $(x_{z^+}, y_{z^+})$ is the coordinate of the original location. This effectively defines a Gaussian prior on the true location relative to the original location $z^+$, preventing part detectors from spuriously firing at regions that by chance appear similar to the part while still allowing the parts themselves to move around in order to best fit the actual part location in each image.

At detection time, we set a threshold $\tau$ on the detector response. If the response is below $\tau$, the part is considered not visible in the image and its appearance descriptor will be set to zero, preventing the classifier from receiving any information about a part that is not present.

### Ensemble of Parts

To obtain a collection of part detectors, we repeat our discovery procedure multiple times. It is worth noting that the randomization throughout our discovery procedure can help increase the robustness of the recognition algorithm. As we will demonstrate in our experiments, increasing the number of randomly sampled part detectors improves performance. See Fig. 3.3 (bottom) for more examples of our part discovery pipeline.

## 3.4 Experiments

### 3.4.1 Datasets and Implementation Details

We evaluate our algorithm on the Cars [95] fine-grained benchmark. We follow the standard evaluation procedure: all training and testing is performed on cropped images from the object bounding boxes. We report classification accuracy, *i.e.* the accuracy as averaged over the test examples. In our experiments no annotations except class labels and bounding boxes are used in training.

We use the `cuda-convnet` implementation [97] of CNNs. The network consists of 48 $11 \times 11$ filters with $3 \times 3$ pooling regions with stride 2 for the first convolutional layer, and 128 $6 \times 6$ filters with pooling regions of size $6 \times 6$ every 6 pixels for the second convolutional layer. The number of units for the three fully connected layers are all 2048. All units are rectified linear units except for the fully connected layers, where we found that linear units work best. We resize each image to $256 \times 256$ and use various techniques for CNNs to prevent overfitting, including dropout, color perturbation, random rotations, and sampling subwindows of $224 \times 224$, as described in [97]. This architecture was determined by extensive experiments using a validation set drawn from the training set, as the network used in [97] suffers from substantial overfitting on fine-grained datasets, which typically have over 100x less training data than in ILSVRC2012 [151], the dataset used to train [97]. At test time, we average predictions over the four corners, middle patch, and their horizontal flips.

For part discovery, each aligned set consists of 1 query image and 49 nearest neighbors. To generate the candidate parts from the aligned images, we randomly sample 5000 patches and pick the top 10 with the highest HOG energy, measured across images. The threshold $\tau$ used for part detection is set to $-1$. While this low threshold results in part detections in nearly every image, even ones in which the part is not present, we have found that this improves performance, with the intuition that the small amount of signal we get by increasing the number of part detections is worth the corresponding increase in noise.

To train our final classifier, a linear SVM, we use as data ELLF features extracted on the original images as well as their horizontal flips. Note that this means that

| Method | Accuracy |
|---|---|
| BB [46] | 63.6 |
| BB-3D-G [95] | 67.6 |
| LLC [185] | 69.5 |
| CNN-SPM (small) | 67.9 |
| CNN-SPM (large) | 69.3 |
| CNN | 70.5 |
| ELLF (ours) | 73.9 |

Table 3.1: Main results on classifying cars. BB, BB-3D-G, and LLC results as reported in [95].

the classifier itself does not have access to some of the data deformations – namely, the color perturbations, random rotations, or subwindow sampling. It also does not use dropout for regularization. Although in principle one could train the SVM with these deformations, part detection remains a relatively costly operation compared to extracting CNN features, which makes computing ELLF features on many deformations expensive. At test time we average predictions over each test image and its horizontal flip to produce the final classification.

### 3.4.2 Results and Analysis

Tab. 3.1 reports our results compared with prior work. ELLF beats the previous state of the art, LLC [185], as well as BB [46] and BB-3D-G [95], two works designed for fine-grained recognition. This validates the claim that learning discriminative features and using them with parts discovered automatically is an effective strategy for fine-grained recognition. In the following sections we present more analysis.

#### Feature Learning

A plain CNN (70.5%) already outperforms previous work using traditional features such as SIFT or HOG (BB, BB-3D-G, and LLC). This is without extra unlabeled data or any kind of pre-training. It suggests that feature learning is able to generate rich appearance descriptors that adapt to particular categories, even with our limited amount of data.

### Usefulness of Parts

We next perform a control experiment that showcase the key benefits our ELLF representation—the same segments of the representations from two images refer to appearances of the same part. To verify this intuition, we replace our detected parts with SPM grids, where the same segments of the representations refer to the same image location instead of part. CNN-SPM(small) and CNN-SPM(large) in table 3.1 report the results of this control experiments. CNN-SPM(small) uses 1x1, 2x2, and 4x4 spatial pooling regions, and CNN-SPM(large) in addition uses 8x8 regions, both building on top of our CNN features. The ELLF representation outperforms both standard and very high-dimensional SPM representations, demonstrating that it is indeed helpful to enable comparing appearances at corresponding parts and that the gains from using ELLF are not simply due to feature dimension. Note that the performance of both CNN-SPM(small) and CNN-SPM(large) is below that of a standard CNN, which is due to the lower number of deformations the SVM classifier is trained on and the fact that it is not trained using dropout (see Sec. 3.4.1 for details).

### Consistency of Parts

To validate that the discovered parts generalize beyond our training data, we show a sample of parts and their top ten detections on the test set in Fig. 3.7. The top two rows show that discovered parts tend to fire rather consistently, even under mild changes in viewpoint. Examples of failure cases are given in the last row, in which 180-degree rotations of cars cause the part detectors to misfire on patches which, although locally very similar to the target part in appearance and position, are nonetheless different parts.

### Number of Parts

We also investigate how much part discovery contributes to performance. Fig. 3.4 plots the classification accuracy versus number of part detectors discovered. It also plots the performance of directly using full CNN models without part discovery

(CNN). Performance increases with the number of parts, up to a point when it plateaus. Remarkably 100 part detectors are sufficient to significantly improve the standalone CNN model. This shows that part discovery is an essential component of our representation. Eventually performance saturates (at around 1000 parts discovered).

### ELLF vs. CNN predictions

In Fig. 3.5 we show a sampling of images where our method was correct and a standard CNN was incorrect, with the parts that contributed most toward the decision value of the correct class displayed. Our part detectors fire on a diverse range of parts, whereas a CNN is confined to a fixed pooling grid. In Fig. 3.6 we show example failure modes of ELLF, where the CNN was correct but ELLF was not. One disadvantage of ELLF's reliance on GrabCut for segmentation is that part detection suffers when the segmentation is flawed, hurting our recognition performance.

### Confusing Classes

In Fig. 3.8 we show the pairs of classes most confused with one another. The confusion score between a pair of classes $C, D$ is determined by

$$Conf(C, D) = \frac{P_{C,D}}{\sum_i P_{C,i}} + \frac{P_{D,C}}{\sum_i P_{D,i}} \tag{3.3}$$

where $P_{a,b}$ is the number of images with ground truth label $a$ predicted as class $b$. These pairs tend to consist of different models or years within the same make, except for the classes Chevrolet Express Cargo Van 2007 vs. GMC Savana Van 2012, in which case the difference in make is visually represented only by the logo small on the front of the van.

### Most Useful Parts

Which parts, discovered in an unsupervised fashion, are most useful for discrimination across all 196 categories? To measure this, we sum the absolute value of the learned

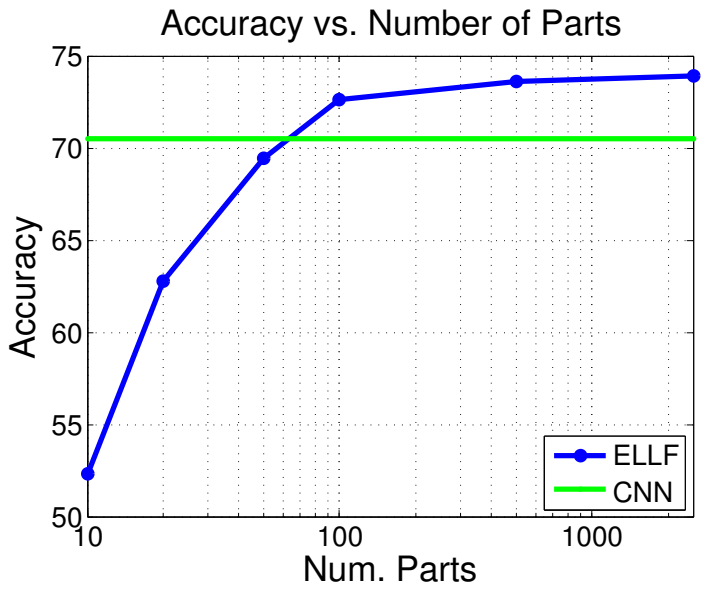



Figure 3.4: Car classification accuracy versus the number of parts used in our ELLF representation. Results achieved using CNN alone are also included (green line).

classifier weights across classes and dimensions for each part to produce an importance score for each part. The top 10 parts are shown in Fig. 3.9. These parts are relatively large and thus can give information about the overall shape or type of car, *e.g.* whether the car is a sedan, SUV, or coupe. We also observe that these parts tend to occur in the top half of the automobiles, almost never overlapping with the tires. This agrees with the intuition that tires are not useful regions to look at when discriminating cars.

### 3.4.3 Limitations

Although it is a step in the right direction, ELLF is far from perfect. Performance can be improved by pre-training the CNN on ImageNet [45, 97], especially for smaller datasets like the Cars dataset [95] used in this paper. However, such constraints will be alleviated as fine-grained datasets continue to grow in size. Second, in our implementation, part detection takes significantly more time than extracting CNN features. In order to increase the practicality of ELLF, part detection needs to be sped up.

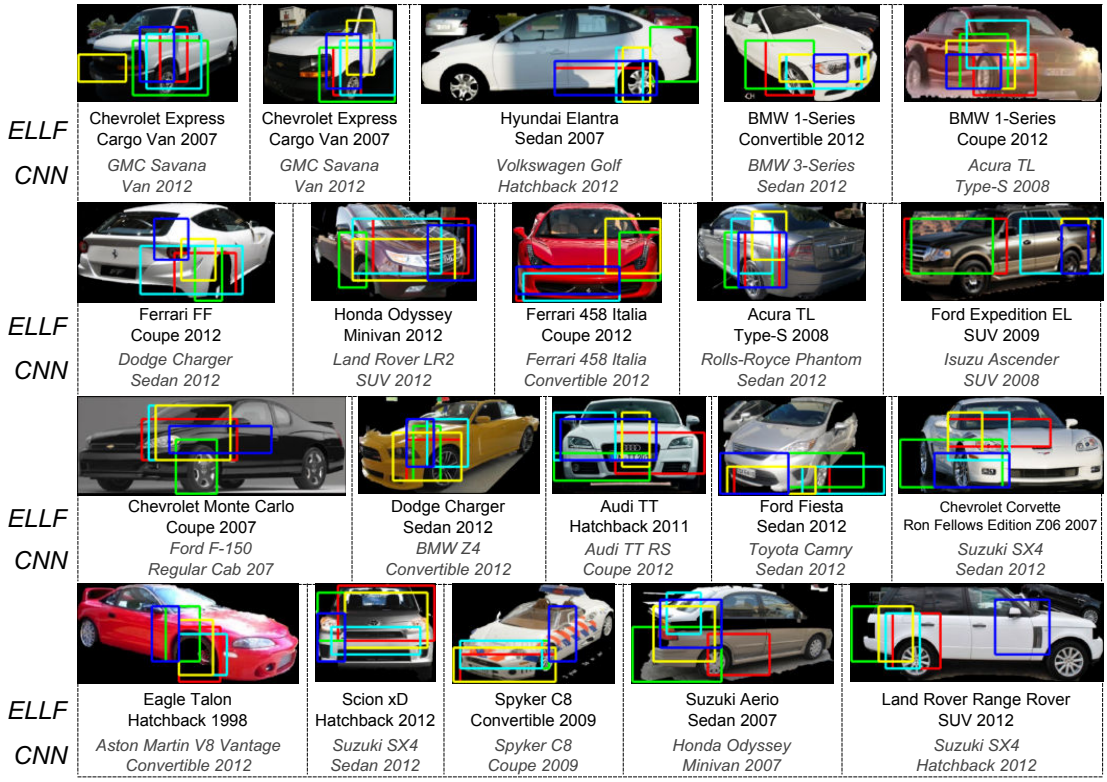



Figure 3.5: Example images where ELLF was correct and a standard CNN was incorrect. On each image the five parts for our method that contributed most to a correct classification are shown. Incorrect predictions are colored gray and in italics.


Figure 3.6: Example failure cases of ELLF. Incorrect predictions are colored gray and in italics. Part detection for ELLF suffers when GrabCut produces an incorrect segmentation, either by segmenting out too much of the target car or by keeping too much of the background.

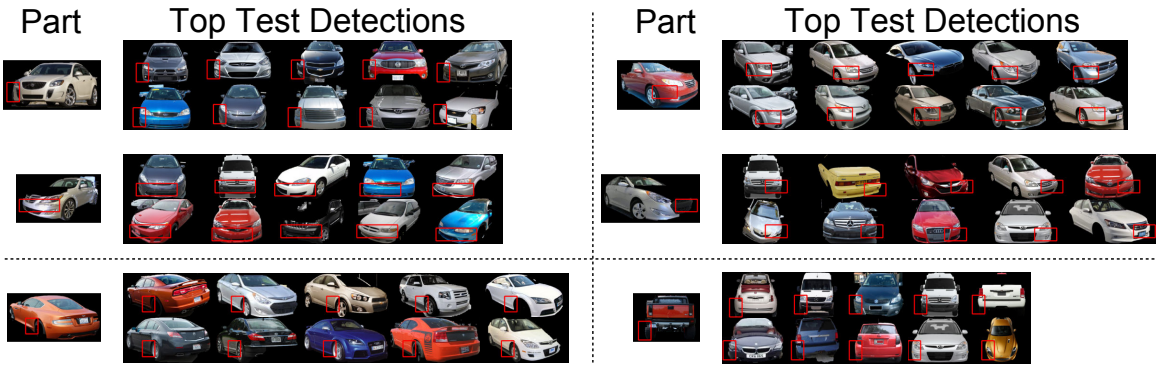



Figure 3.7: A sample of parts and the ten test detections with the highest response. Each part is visualized on top of the seed image of the neighborhood which produced the trained part. The first two rows are success cases: the parts detectors fire consistently on the same parts of the car, even under the presence of some viewpoint variation. The last row are failure cases: since each part detector is based on local evidence, when different parts have the same appearance and occur in the same position in the image plane, as can occur when a car undergoes a 180-degree rotation, the part detectors misfire.


Figure 3.8: The five most confusing pairs of classes for ELLF in the Car dataset, in descending order of confusion as determined by Eq. 3.3. We observe that these pairs of classes differ only in very small details.

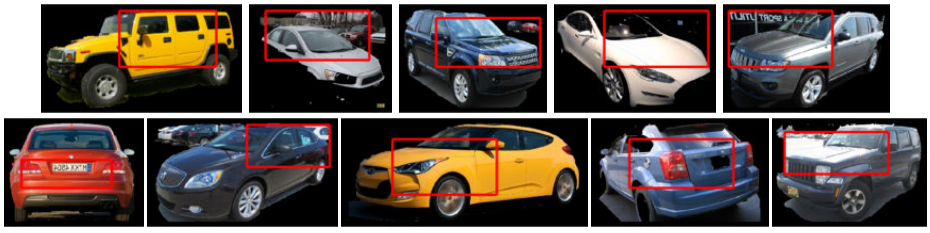


Figure 3.9: The most useful parts for overall car classification. These parts tend to be large, giving information about the general type of car (SUV, sedan, etc.).

Finally, instead of extracting CNN features, disjointly learning parts and then learning a classifier (an SVM), jointly learning parts with features would likely bring improvements to both part discovery and feature learning. This method would also enable us to further take advantage of data deformations, since the cost of part detection makes it impractical to train our classifier on a significant number of deformations (see Sec. 3.4.1). Although the advantages of jointly learning features and parts are clear, training such a non-rigid neural network efficiently (*i.e.* on a GPU) poses many implementation challenges – learning movable parts and pooling regions is an open problem in the design and implementation of neural networks.

## 3.5 Conclusions

In this paper we have proposed an approach for fine-grained recognition that tackles both feature learning and part discovery. Our main results are 1) that learning discriminative features in a supervised setting can be effective for fine-grained recognition, even at the small scales present in current fine-grained datasets, and 2) that one can learn parts useful for recognition without any part-level annotations. In the future we would like to combine part discovery and feature learning into a joint model and lift our part representation into 3D, which should yield more accurate correspondences.

## 3.6 Acknowledgments

This work was partially supported by an ONR MURI grant and the Yahoo! FREP program.

# Chapter 4

# Fine-Grained Recognition without Part Annotations

Scaling up fine-grained recognition to all domains of fine-grained objects is a challenge the computer vision community will need to face in order to realize its goal of recognizing all object categories. Current state-of-the-art techniques rely heavily upon the use of keypoint or part annotations, but scaling up to hundreds or thousands of domains renders this annotation cost-prohibitive for all but the most important categories. In this work we propose a method for fine-grained recognition that uses no part annotations. Our method is based on generating parts using co-segmentation and alignment, which we combine in a discriminative mixture. Experimental results show its efficacy, demonstrating state-of-the-art results even when compared to methods that use part annotations during training.

This work was done while interning with Adobe Research, is joint work with Hailin Jin, Jianchao Yang, and Fei-Fei Li, and appeared in [93].

## 4.1 Introduction

Models of fine-grained recognition have made great progress in recognizing an ever-increasing number of categories. Performance on one standard dataset [182] has increased from 10.3% [182] to 75.7% [25] in only three years. On the data side, there

has also been progress in expanding the set of fine-grained domains we have data for, which now includes *e.g.* birds [182, 186, 19], aircraft [176, 117], cars [164, 95, 114], flowers [132, 10], leaves [100], and dogs [88, 115].

Compared to generic object recognition, fine-grained recognition benefits more from learning critical parts of the objects that can help align objects of the same class and discriminate between neighboring classes [18, 56, 204, 30, 46]. Current state-of-the-art results are, therefore, from models that require part annotations as part of the supervised training process [203, 25]. This poses a problem for scaling up fine-grained recognition to an increasing number of domains.

Figure 4.1: In fine-grained recognition, categories share similar shapes, which allows for alignment to be done purely based on segmentation.

Towards the goal of training fine-grained classifiers without part annotations, we make an important observation. Fig 4.1 illustrates our idea. Objects in a fine-grained class share a high degree of shape similarity, allowing them to be aligned via segmentation alone. If we can align them early in the training process, we can learn the characteristic parts without the annotation effort.

In this work, we propose a method to generate parts which can be detected in novel images and learn which parts are useful for recognition. Our method for generating parts leverages recent progress in co-segmentation [68, 98] to segment the training images. We then densely align images which are similar in pose, performing alignment across all images as the composition of these more reliable local alignments. Despite

using fewer annotations, our method is state-of-the-art on the competitive CUB-2011 dataset [182] when using a VGGNet [162] for feature extraction, is on par with current state-of-the-art even when using a weaker CaffeNet [84] architecture, and is furthermore able to generalize to fine-grained domains which do not have part annotations, establishing a new state-of-the-art on the cars-196 [95] dataset by a large margin.

The remainder of the paper is organized as follows: We review related work in Sec. 4.2 and describe our method for generating parts in Sec. 4.3. Our use of these parts for recognition is covered in Sec. 4.4. We present experiments and analysis on the CUB-2011 and cars-196 datasets in Sec. 4.5 and conclude with future work in Sec. 4.6.

## 4.2 Related Work

**Fine-Grained Recognition.** A variety of methods have been developed for differentiating between fine-grained categories. Though many early approaches [51, 197, 198, 196] did not use part annotations, their performance has been eclipsed by methods developed to explicitly take advantage of the structure present in fine-grained classes [18, 25, 203, 205, 46]. A few works have even gone beyond the use of 2D part annotations, aiming to get a full correspondence across images via a 3D representation [56, 95]. Still other methods explore fine-grained classification with a human in the loop at test time, *e.g.* the visipedia project [27, 181, 183], which is complementary to our approach.

Of the methods developed that do not use part annotations, there have been a few works philosophically similar to ours in the goal of finding localized parts or regions in an unsupervised fashion [51, 62, 30], with [62] and [30] more relevant. Gavves *et al.* [62, 61] segment images via GrabCut [149], and then roughly align objects by parameterizing them as an ellipse. Chai *et al.* [30] do a joint segmentation and DPM [57] model fitting, extracting features around each DPM part. In contrast to these works, our alignment model is computed densely and is the composition of easier alignments between similar images. We also perform the task of detection at

test time and ultimately achieve better classification results than either.

Current state-of-the-art methods are Zhang *et al.*[203] and Branson *et al.*[25], which are both supervised at the level of parts during training. Both employ a part detection model, with [203] generalizing the R-CNN framework [65] to detect parts in addition to the whole object, and [25] training a strongly-supervised deformable part model in a structured learning framework [24]. Of these two, our method is more related to [203] in that we use an R-CNN model for detection, but unlike either our method is completely unsupervised at the level of parts.

**Co-segmentation.** Co-segmentation, the task of segmenting the object common to a set of images, has made great strides in recent years [68, 150, 86, 29, 31, 30, 98]. Co-segmentation has even seen some success in fine-grained recognition [29, 31, 30], exploiting the low intra-class variability of fine-grained classes. Unlike [29, 31, 30], our co-segmentation approach follows a graph-cut approach, inspired by Guillaumin, Kuettel, and Ferrari [68]. The main difference between our approach and [68] is that bounding boxes are available during training in our problem setting. We use this to significantly improve segmentation quality via a refinement step that finds a segmentation covering the bounding box well. This fixes many common failure modes typical of a graph-cut segmentation framework. Similar intuition with a more sophisticated approach can be found in [105]. In our setting no ground truth segmentations are available, unlike the full model of [68], so we are thus more related to their "image+class" model.

## 4.3 Generating Parts

At the core of our approach for generating parts is the concept of *alignment by segmentation*, the process of aligning images via aligning their figure-ground segmentation masks. The key insight is that, even for complicated and deformable objects such as birds, a figure-ground segmentation (Fig. 4.2(b)) is often sufficient in order to determine an object's pose and localize its parts, as demonstrated in Fig. 4.1. We

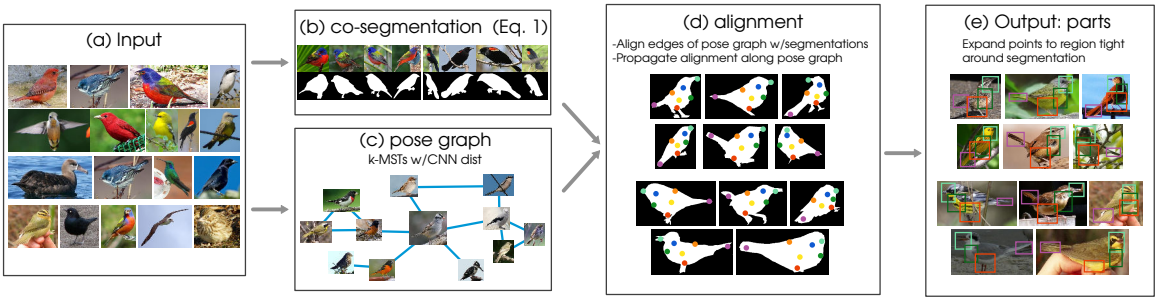



Figure 4.2: An overview of our method to generate parts used for recognition. We begin by segmenting all images in the training set with a co-segmentation approach (Sec. 4.3.1) and finding a graph used to determine which images to align (Sec. 4.3.2). With these, we sample points across all images, which form the basis for generating parts used in recognition (Sec. 4.4).

decompose the process of aligning all images as aligning pairs of images with similar poses, which we represent in a graph (Fig. 4.2 (c)), producing a global alignment (Fig. 4.2(d)) from these easier, local alignments. Based on these alignments we sample points across all images, which each determine a part (Fig. 4.2(e)).

### 4.3.1 Co-segmentation

In order to do alignment by segmentation, one must first establish a figure-ground segmentation of each image, which we do via co-segmentation (Fig. 4.2(b)). Co-segmentation is particularly attractive for fine-grained recognition because the appearance variation within each class is relatively small. Traditional co-segmentation approaches [150, 86] typically assume that no information besides object presence is available for each image, while in the setting of fine-grained recognition we also have bounding boxes available in these training images. Our approach effectively and efficiently uses this information.

**Formulation.** Our optimization for co-segmentation is inspired by Guillaumin *et al.* [68] and uses a GrabCut [149]-like approach at its base. Let $\theta_f^i$ be a foreground color model for image $i$, represented as a Gaussian mixture model, $\theta_b^i$ a similar background model, and $\theta_f^c$ a shared foreground color model for class $c$. The binary

assignment of pixel $p$ in image $i$ to either foreground or background is denoted $x_p^i$, its corresponding RGB value is $z_p^i$, the set of segmentation assignments across all images is $X$, and $p_f$ is a pixelwise foreground prior which we describe shortly. Our co-segmentation objective is:

$$\max_{X,\theta} \sum_i \left( \sum_p E(x_p^i, \theta^i, \theta_f^c; p_f^i) + \sum_{p,q} E(x_p^i, x_q^i) \right) \tag{4.1}$$

where

$$\begin{aligned} E(x_p^i, \theta^i, \theta_f^c; p_f^i) &= (1 - x_p^i) \log(p(z_p^i; \theta_b^i)) \\ + \frac{x_p^i}{2} \left( \log(p(z_p^i; \theta_f^i)) + \log(p(z_p^i; \theta_f^c)) \right) &+ E(x_p^i; p_f), \end{aligned} \tag{4.2}$$

$$E(x_p^i; p_f) = \begin{cases} \log(p_f) & x_p^i = 1 \\ \log(1 - p_f) & x_p^i = 0 \end{cases}, \tag{4.3}$$

and $E(x_p^i, x_q^i)$ is the standard pairwise term between pixels $p$ and $q$ for a GrabCut [149] segmentation model, enforcing consistency between neighboring pixels with respect to their assigned binary foreground/background values. If $p_f = .5$ and $\theta_i^f = \theta_i^c$ then this is equivalent to GrabCut, and if only $p_f = .5$ then it reduces to the "image + class" model of [68] without the learned per-term weights.

Optimization is performed separately for each fine-grained class $c$, and proceeds by iteratively updating the appearance models $\theta_f^i$, $\theta_b^i$, $\theta_f^c$ and optimizing the foreground/background masks $x^i$. As is standard in a GrabCut formulation, we initialize the appearance models using the provided bounding boxes, with the pixels inside each bounding box marked foreground and the rest as background. This initial background remains fixed as background throughout the optimization.

**Foreground Refinement.** Though we have already used the bounding boxes available in fine-grained training sets in a standard GrabCut[149] fashion, we have not fully exploited their usefulness. As noted by [105], objects in bounding boxes typically occupy a non-negligible portion of the bounding box, and we can use this

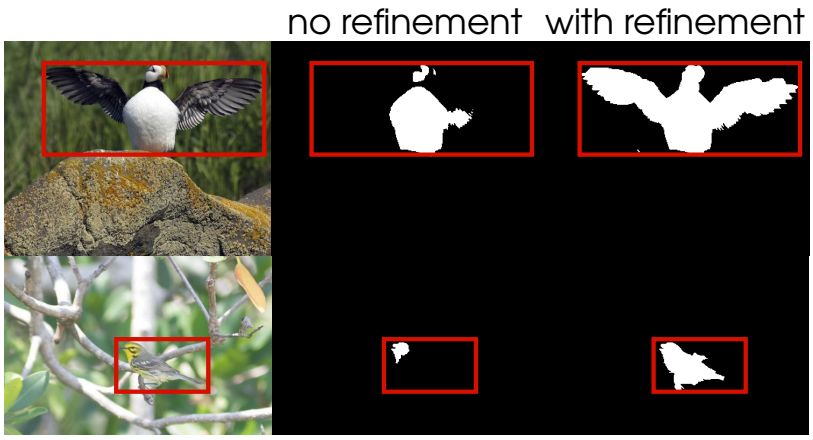



Figure 4.3: Refinement by searching for a foreground prior $p_f$ satisfying weak bounding box-level constraints can correct very large errors in segmentation.

knowledge to significantly improve segmentation quality. Formally we represent this as constraints that a foreground segmentation must occupy between $\omega_1$ and $\omega_2$ of the total area of its bounding box and span at least $\rho$ of its width and height. To satisfy these constraints, we perform a binary search over the pixelwise foreground prior $p_f$ (Eq. 4.3) on a per image basis after the initial segmentation until the constraints are satisfied, initializing with $p_f = .5$. Since $p_f = 0$ produces an entirely background segmentation and $p_f = 1$ corresponds to an entirely foreground segmentation, this search will produce a segmentation satisfying the constraints.

In our experiments we set $\omega_1 = 10\%$, $\omega_2 = 90\%$, and $\rho = 50\%$, representing a weak set of constraints that is satisfied in 99.97% of the images in CUB-2011 [182]. As we demonstrate in our experiments and visualize in Fig. 4.3, this weak prior dramatically improves segmentation results, and tends to fix a common failure mode of GrabCut segmentation methods of over- or under-segmenting images. More qualitative results are given in the experiments. We also note that most initial segmentations already satisfy these constraints, so this extra binary search does not significantly change the running time of co-segmentation, and is much cheaper than more sophisticated methods [105].

### 4.3.2 Choosing Images to Align

Aligning two objects with arbitrary poses remains a hard problem in computer vision. However, when given many instances of the same category, any single object is likely to have a similar pose with at least one other instance. This motivates our approach of decomposing the global task of aligning all training images into many smaller, simpler tasks of aligning images containing objects of similar poses. We formalize this requirement as building a connected graph $\mathcal{G}$ of images $\{I_i\}_{i=1}^n$ where each edge $(I_i, I_j)$ is between two images containing objects of similar poses to be aligned. To reduce the variance in alignment, we furthermore require that each image $I_i \in \mathcal{G}$ be connected to at least $k$ other images, aggregating all image to image alignments from the neighbors of $I_i$ to increase robustness. Because $\mathcal{G}$ represents a graph of pose similarity, we refer to it as a *pose graph* (Fig. 4.2 (c)).

How can we determine which images contain objects of similar poses without part annotations and without attempting the comparatively expensive process of aligning them in the first place? We do this with a simple, yet effective heuristic: as a proxy for difference in pose we measure the cosine distance between fourth-layer convolutional ($\text{conv}_4$) features around each bounding box, using a CNN pretrained on ILSVRC 2012 [84, 97, 151]. These intermediate features tend to be fairly robust to changes in *e.g.* background while maintaining information about pose; features in earlier layers are not as robust and features in later layers become too class-specific, eschewing pose information in favor of discriminative power. We have also experimented with using our segmentations from Sec. 4.3.1 and other feature representations to measure pose similarity, but have found very little difference in performance from simply using these CNN features. Fig. 4.4 shows examples of the nearest neighbors calculated with this metric, with more examples included in the experiments.

**Pose Graph Construction.** Using this distance metric we construct $\mathcal{G}$ satisfying the constraints by iteratively computing disjoint minimum spanning trees of the images, merging the trees into a single graph. Concretely, we decompose the pose graph as $\mathcal{G} = \bigcup_{i=1}^k M_i$, where $M_1$ is the minimum spanning tree of the dense graph $\mathcal{G}_D$ on all $n$ images with edge weights given by cosine distance, and $M_j$ is the minimum


Figure 4.4: Nearest neighbors with $\text{conv}_4$ features, which tend to preserve pose.

spanning tree of $\mathcal{G}_D \setminus \bigcup_{i=1}^{j-1} M_i$, which can be computed by setting the weights of all edges used in $M_1, \ldots, M_{j-1}$ to infinity. Since minimum spanning trees are connected, $\mathcal{G}$ is connected, and since $\mathcal{G}$ is composed of $k$ disjoint minimum spanning trees, each node in $\mathcal{G}$ is connected to at least $k$ other vertices, satisfying the constraints.

### 4.3.3 Aligning All Images

Given a pose graph connecting images with similar poses, what is the right way to use its structure to create an alignment between all images? In our approach, we first sample a large set of points in one image, representing the overall shape of an object, and then propagate these points to all images using the structure of the graph.

We start by sampling a set of points of size $k_1$ on the segmented foreground of a single image $I_r$, which we choose to be the image with the highest degree in $\mathcal{G}$. Then, while there is still at least one image that the points have not been propagated to, we propagate to the image $I_j$ adjacent to the largest number of images in $\mathcal{G}$ which have already been propagated to. Let $\tau_{i,j} : \mathbb{R}^2 \to \mathbb{R}^2$ be a dense alignment function mapping a point in image $i$ to its corresponding point in image $j$, which is learned based on the segmentations for $I_i$ and $I_j$. Then, to propagate each of the $k_1$ points, we use $\tau_{i,j}$ to propagate the corresponding point from each image $I_i$ adjacent to $I_j$ in $\mathcal{G}$ and aggregate these separate propagated points via an aggregation function $\alpha$, which

we take to be the median of points propagated from each adjacent image. In this work we learn each dense alignment function $\tau_{i,j}$ with shape context [16], which we make robust to horizontal flips by additionally optimizing over the choice of flipping one of the images.

### 4.3.4 From Alignment to Parts

After globally aligning all images, the problem remains of using the alignment to generate parts for use in recognition. Without any part annotations, at this point it is not possible to tell which parts are semantically meaningful or useful for classification, so we instead target our part generation at producing a *diverse* set of parts. Specifically, we select a subset of the propagated points of size $k_2$ to be expanded into parts for recognition. We do this by clustering the trajectories of the $k_1$ points across all images, *i.e.* we represent each point $i$ by its $2 \times n$-dimensional trajectory across all images, then cluster each of these trajectories via k-means into $k_2$ clusters, providing a good spread of points across the foreground of each image (Fig. 4.2 (d)).

We generate a single part from each one of these $k_2$ points by taking an area around each point with a fixed size with respect to the object's bounding box, then shrink the region until it is tight around the estimated segmentation (Fig. 4.2 (e)), yielding a tight bounding box around each generated part in each training image.

## 4.4 Using Parts for Recognition

Given a set of generated parts, what is the right way to use them for recognition, and how can they be found in novel images which do not even contain object-level bounding boxes? For classification, we propose an approach to discriminatively learn a mixture of parts, and for detection we compare two approaches which both can find our generated parts in novel images.


Figure 4.5: Not all parts of an object are equally useful for recognition. Some, such as the legs, are only rarely useful, while others, like the head, contain most information useful for discrimination.

### 4.4.1 Discriminative Combination of Parts

An effective fine-grained classifier must reason about information at all parts, combining multiple sources of visual information. The most straightforward approach, shared by [203], [25], and most prior approaches, is to concatenate features at each part into one large vector and train a single classifier. However, doing so ignores the observation that not all parts are equally useful for recognition (Fig. 4.5). Motivated by this, we propose a more nuanced approach. Our approach is inspired by the max-margin template selection method of [34], originally used for visual font recognition.

Let $f_p^i$ be features for image $i$ at part $p$, and let $w_{p,c}$ be classification weights for part $p$ and class $c$, learned for each part independently. Our goal is to learn a vector of $(k_2 + 1)$-dimensional weights $v$ satisfying:

$$\min_v \sum_{i=1}^{n} \sum_{c \neq c^i} \max\left(0, 1 - v^T u^i_{c^i,c}\right)^2 + \lambda \|v\|_1 \tag{4.4}$$

where the $p$-th element of $u^i_{c^i,c}$ is the difference in decision values between correct

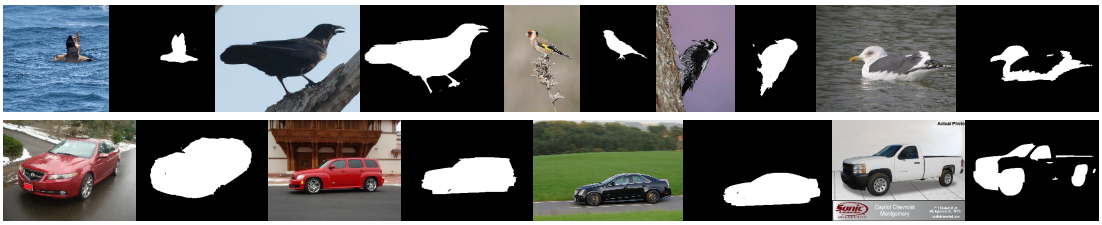


Figure 4.6: Example segmentations from our co-segmentation method on CUB-2011(top) and cars-196(bottom). The last image in each row is a failure case.

class $c^i$ and incorrect class $c$:

$$u^i_{c^i,c}(p) = (w_{p,c^i} - w_{p,c})^T f^i_p \tag{4.5}$$

This is equivalent to a one-class SVM (an SVM with only positive labels) with an $L_2$ loss and $L_1$ regularization, and can thus be solved efficiently by standard SVM solvers. Intuitively, this optimization tries to select a sparse weighing of classifiers such that, combined, the decision value for the correct class is always larger than the decision value for every other class by some margin, forming a discriminative combination of parts. Decision values for each $u^i$ can be calculated via cross-validation while training the independent classifiers at each part. In comparison to [34], the main difference is that our formulation operates directly on decision values rather than the probabilities output by the template system in [34]. The final classification is given by:

$$\arg\max_c \sum_{p=1}^{k_2} v_p w^T_{p,c} f_p \tag{4.6}$$

### 4.4.2 Finding Parts at Test Time

The other main challenge in using our automatically generated parts is finding them in novel, completely unannotated images. We experiment with two different methods, one based on [203] and the other a direct extension of our part generation method, applied to bounding box-level object detections.

**Part Detectors.** Our first method for locating parts at test time involves training dedicated part detectors, and is based on the $\Delta_{\text{box}}$ method of Zhang *et al.* [203], originally intended for use with ground truth part locations: an R-CNN [65] is trained for the entire object and every part by treating each as a separate category. At test time, all detectors are run on an image, each detection score is transformed into a probability via a sigmoid function, and the joint configuration of bounding box and parts is scored as the product of probabilities, with part detections that do not fall within 10 pixels of the bounding box set to probability 0. Since our set of generated parts is potentially much larger in size than the set of parts considered in [203], we change the joint configuration scoring method to normalize the log-probabilities from the part detectors by $\frac{1}{k_2}$. This gives the information at local parts and the global object equal weight, robustly combining the two. Our other difference from [203] is training each R-CNN with bounding box regression, which improves detection AP on CUB-2011 from 88.1 to 92.9.

**Test-Time Segmentation and Alignment.** Our second method is a direct extension of the segmentation and alignment done on the training images. We first do detection with an R-CNN trained on the whole bounding box, then use this predicted bounding box in our segmentation framework of Sec. 4.3.1, removing the foreground class appearance term since the class label is unknown at test time. The nearest neighbors of the test image in the training set are calculated using $\text{conv}_4$ features and alignment from those images is done exactly as described in Sec. 4.3.2. This method has an advantage over the part detector method in that an R-CNN only needs to be trained for the whole object, rather than $k_2 + 1$ categories, and it also produces a segmentation of test images, which may be useful for other applications, but it has the disadvantage of being somewhat slower at test time due to the alignment and segmentation steps. In all experiments with this method, we use 5 nearest neighbors.

## 4.5 Experiments

### 4.5.1 Datasets

We evaluate on the CUB-2011 [182] and cars-196 [95] datasets. CUB-2011 contains 11,788 images of 200 species of birds, and is generally considered the most competitive dataset within fine-grained recognition, while cars-196 has 16,185 images of 196 car types. Both datasets have a single bounding box annotation in each image, and CUB-2011 moreover contains rough segmentations and 15 keypoints annotated per image, which we do not use in our algorithm.

### 4.5.2 Implementation Details

For all R-CNN training we use $\text{fc}_6$ features from a network trained on ILSVRC2012 [151] with no fine-tuning and 16 pixels of padding around each bounding box or part. The R-CNN takes the majority of running time, at about 20 sec./image. Methods without fine-tuning also use $\text{fc}_6$ features, extracted on each part with 16 pixels of padding. All features are extracted using Caffe [84]. R-CNNs are trained with the default network (pre-trained on ILSVRC2012 and fine-tuned on PASCAL). Features for pose graph construction are from a pre-trained CaffeNet [84], which is similar to [97]. When fine-tuning networks in our experiments, two separate networks are fine-tuned: one for the whole object and one for the set of all generated parts. All regularization parameters and the weights for the discriminative combination of parts for fine-tuned networks are determined based on non-fine-tuned features, since fine-tuning makes training decision values overconfident. For part generation we set $k = 5$, $k_1 = 500$, $k_2 = 31$, and use a maximum part size of 25% of the geometric mean of the bounding box dimensions around each keypoint. We defer other implementation details to our code, available on the first author's website.

### 4.5.3 Co-segmentation Results

We first perform an analysis of our co-segmentation method, evaluated on CUB-2011. Results are given in Tab. 4.1. Interestingly, adding in the class foreground appearance

| Method | Jac. Sim. |
|---|---|
| GrabCut [149] | 70.84 |
| +class ≈ [68] | 67.78 |
| +refine | 74.72 |
| +class+refine | 75.47 |

Table 4.1: Co-segmentation results on CUB-2011 as measured by Jaccard similarity of the ground truth foreground with the predicted segmentation. GrabCut is equivalent to our model without the class foreground term or foreground prior refinement, which we add in "+class" and "+refine", respectively, with "+class+refine" corresponding to our full co-segmentation model.

term hurts a pure GrabCut approach, but helps when a refinement step is added in. This happens because with only the additional class term, the foreground model underfits, tending to result in an undersegmentation. However, when the refinement step is added, the learned class term provides for a strong refinement initialization, allowing the per-image term to fit the foreground more accurately. This result highlights the different approach needed for co-segmentation when given bounding boxes, as class appearance terms help substantially in the no bounding box case [68]. Tab. 4.1 also demonstrates the importance of the foreground prior refinement, improving upon GrabCut by nearly 4%. Fig. 4.6 shows qualitative results on CUB-2011 and cars-196, with more results given in the experiments (Fig. 4.11). Our approach is generally able to segment the foreground object well, but understandably has trouble when the foreground and background are too similar.

### 4.5.4 Recognition Results

We first perform a detailed analysis of our method on CUB-2011 before moving on to compare to other methods on both CUB-2011 and cars-196.

**Method Analysis.** Tab. 4.2 details many variants of our method, using the $fc_6$ layer of a CaffeNet [84] for feature extraction. We observe that the part detector method works better than the test-time segmentation method, performing better under both part combination strategies. This indicates that the learned part detectors

| Method | Acc. |
|---|---|
| R-CNN [65] | 58.8 |
| R-CNN+ft | 65.3 |
| CNN+GT BBox | 61.3 |
| CNN+GT BBox+ft | 67.9 |
| TS+concat | 63.4 |
| TS+DCoP | 68.5 |
| PD+concat | 67.6 |
| PD+DCoP | 69.7 |
| PD+DCoP+flip | 71.1 |
| PD+DCoP+flip+ft | 73.7 |
| PD+DCoP+flip+GT BBox | 72.4 |
| PD+DCoP+flip+GT BBox+ft | 74.9 |

Table 4.2: Analysis of different variants of our method on CUB-2011. R-CNN refers to training an R-CNN [65] for birds generically and extracting features on the whole bounding box. "+ft" means that the CNN used to extract features after detection was fine-tuned for classification. "PD" refers to using the generated parts in a part detection framework, and "TS" refers to the method of doing segmentation at test time and aligning the test image with a set of training images (Sec. 4.4.2). "concat" and "DCoP" are the two methods of combining multiple parts, and refer to concatenating the features and the discriminative combination of parts (Sec. 4.4.1), respectively. "+flip" indicates training and testing with both original and horizontally flipped images, averaging the decision values during test, and "+GT BBox" indicates giving the method oracle bounding box information. Performance is measured with 200-way accuracy.

| | CNN Used | |
|---|---|---|
| Method | [84] | [162] |
| R-CNN [65] | 58.8 | 69.0 |
| R-CNN+ft | 65.3 | 72.5 |
| CNN+GT BBox | 61.3 | 70.0 |
| CNN+GT BBox+ft | 67.9 | 75.0 |
| PD+DCoP+flip | 71.1 | 78.8 |
| PD+DCoP+flip+ft | 73.7 | 82.0 |
| PD+DCoP+flip+GT BBox+ft | 74.9 | 82.8 |

Table 4.3: Impact of CNN choice on variants of our method, measured in 200-way accuracy.

| Method | CNN Used | |
|---|---|---|
| | [84] | [162] |
| R-CNN [65] | 51.0 | 57.4 |
| R-CNN+ft | 73.5 | 88.4 |
| CNN+GT BBox | 53.9 | 59.9 |
| CNN+GT BBox+ft | 75.4 | 89.0 |
| PD+DCoP+flip | 65.8 | 75.9 |
| PD+DCoP+flip+ft | 81.3 | 92.6 |
| PD+DCoP+flip+GT BBox+ft | 81.8 | 92.8 |

Table 4.4: Analysis of variations of our method on cars-196, comparing performance when using a CaffeNet [84] versus a CNN with a VGGNet architecture [162]. Performance is measured in 196-way accuracy.

are able to generalize well to unseen images, and motivates our use of the part detector method for the rest of the analysis. Second, combining the parts discriminatively is always better than the concatenation strategy. We note that PD+DCoP, without bounding boxes during test time and without fine-tuning, is already able to outperform a fine-tuned CNN when given the ground truth bounding box, highlighting the importance of reasoning at the level of parts and validating the effectiveness of our approach.

We visualize the parts with the highest weights in the discriminative combination of parts in Fig. 4.7(left) and observe that they both fire consistently and represent a diverse set of parts, with the top two parts (bird heads with varying amounts of context, extremely discriminative on their own) excepted. We furthermore show example images in which our method classifies correctly but an R-CNN with fine-tuning is incorrect in Fig. 4.8. Even in cases of unusual pose or inaccurate detections, our method is still able to accurately localize one or more parts and classify correctly, while the R-CNN is forced to reason at a whole bounding box level, unable to discriminate the fine-grained classes.

Adding in flipped images improves performance by another 1.4%, and fine-tuning improves that further by 2.6%. Granting our method the ground truth bounding box at test time improves results by only 1.3% (1.2% with fine-tuning), suggesting that improvements to the detection model will not impact classification results much on

Figure 4.7: The top parts chosen by our method, excepting the whole object "part", visualized by the highest scoring detections in the test set. Each row is a different part. Shown at left are our top parts for CUB-2011 and at right are the top parts for cars-196.

CUB-2011.

**Impact of CNN Architecture.** We also analyze the effect of network architecture used to extract features at each part, comparing CaffeNet [84] and VGGNet (the 19-layer configuration "E" from [162]) on a subset of method variants. Tab. 4.3 details the results on CUB-2011. In all cases, using a VGGNet significantly improves results, so we present the remainder of the results using a VGGNet for feature extraction.

We show a similar comparison on the cars-196 [95] dataset in Tab. 4.4. As before, using a VGGNet architecture leads to large gains. Particularly striking is the gain from fine-tuning a VGGNet on cars-196 – a basic R-CNN goes from 57.4% to 88.4% accuracy only by fine-tuning, much larger than the already sizeable gain from fine-tuning a CaffeNet [84].

**Comparison to Prior Work on CUB-2011.** In Tab. 4.5 we compare our method to prior work on CUB-2011, listing the amount of annotation each method uses. Our method is best by a large margin among methods which use no annotations at test time, and even outperforms all methods that use part annotations during training, only beaten by the variant of PN-DCN [25] which uses part annotations

| Method | Train Anno. | Test Anno. | Acc. |
|---|---|---|---|
| Alignments [61] | *n/a* | *n/a* | 53.6 |
| **Ours** | **BBox** | ***n/a*** | **82.0** |
| GMTL [144] | BBox | BBox | 44.2 |
| Symbiotic [30] | BBox | BBox | 59.4 |
| Alignments [61] | BBox | BBox | 67.0 |
| **Ours+BBox** | **BBox** | **BBox** | **82.8** |
| PB R-CNN [203] | BBox+Parts | *n/a* | 73.9 |
| PN-DCN [25] | BBox+Parts | *n/a* | 75.7 |
| DPD [205] | BBox+Parts | BBox | 51.0 |
| POOF [18] | BBox+Parts | BBox | 56.8 |
| Nonparametric [66] | BBox+Parts | BBox | 57.8 |
| Symbiotic [30] | BBox+Parts | BBox | 61.0 |
| DPD+DeCAF [49] | BBox+Parts | BBox | 65.0 |
| PB R-CNN [203] | BBox+Parts | BBox | 76.4 |
| Symbiotic [30] | BBox+Parts | BBox+Parts | 69.5 |
| POOF [18] | BBox+Parts | BBox+Parts | 73.3 |
| PB R-CNN [203] | BBox+Parts | BBox+Parts | 82.0 |
| PN-DCN [25] | BBox+Parts | BBox+Parts | 85.4 |

Table 4.5: Comparison of different methods on CUB-2011, sorted by amount of annotation used. "Ours" indicates our full model (PD+DCoP+flip+ft), and "Ours+BBox" grants our method the ground truth bounding box at test time. "Parts" refers to using any annotation at the level of parts at all. Since the exact amount of annotation used varies from work to work, we defer to the original sources for details.

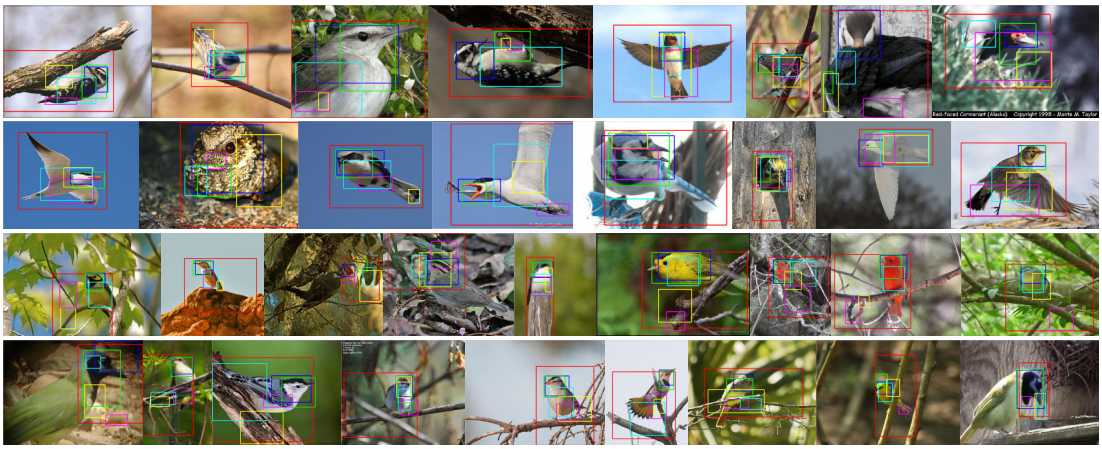


Figure 4.8: Test images in which our method is correct but an R-CNN with fine-tuning is incorrect, visualized with detections for the whole object and five parts with the highest weight in the discriminative combination of parts. The top two rows depicts images where birds have unusual poses, and the bottom two rows show cases where the detection is inaccurate.

at test time (and tied with the variant of PB R-CNN that does the same). Using the weaker CaffeNet architecture, our results are within two percent of the variants of PB R-CNN [203] and PN-DCN [25] that use no annotations at test time but use part supervision during training. We anticipate that improving PB R-CNN and PN-DCN with better CNN architectures will again result in performance higher than our best approach due to their additional supervision. Expert human performance on CUB-2011 is roughly 93% [26].

**Comparison on cars-196** The main advantage of our method is that it allows us to do accurate recognition on classes that do not have part annotations, scaling up to a larger number of fine-grained domains, which methods such as [25, 203, 18] cannot do. To this end, we compare performance on the cars-196 [95] dataset, with results given in Tab. 4.6. All results not reported by prior work use the VGGNet [162] architecture. Our method is able to greatly outperform all previously-reported results [46, 95, 185, 92], setting a new state-of-the-art by 18.7%. Parts chosen in the discriminative combination of parts are shown in Fig. 4.7. These parts tend to either contain information relating to either the general body type of cars (top two parts),

| Method | Train | Test | Acc. |
|---|---|---|---|
| R-CNN [65] | BBox | *n/a* | 57.4 |
| R-CNN+ft | BBox | *n/a* | 88.4 |
| **Ours** | **BBox** | ***n/a*** | **92.6** |
| CNN+GT BBox | BBox | BBox | 59.9 |
| BB [46] | BBox | BBox | 63.6 |
| BB-3D-G [95] | BBox | BBox | 67.6 |
| LLC [185] | BBox | BBox | 69.5 |
| ELLF [92] | BBox | BBox | 73.9 |
| CNN+GT BBox+ft | BBox | BBox | 89.0 |
| **Ours+GT BBox** | **BBox** | **BBox** | **92.8** |

Table 4.6: Comparison of methods on cars-196 [95]. Performance is measured with 196-way accuracy.

or focus on fine details such as the grille or headlights.

### 4.5.5 Additional Visualizations

Here we present additional visualizations of several components of our method.

**Pose Nearest Neighbors** In Fig. 4.9 we show more examples of nearest neighbors using $\text{conv}_4$ features, which is our heuristic for measuring the difference in pose between different images. In most cases the nearest neighbors of an image come from a variety of fine-grained classes and tend to have similar poses, justifying their use as a heuristic. In cases where there are potentially many instances with similar poses (*e.g.* first row, third column, or fifth row, first column), the nearest neighbors may share more than just pose. This heuristic still works reasonably when the pose is relatively unusual (third row, first column, and fourth row, third column), although occasionally small pose differences persist (direction of the head in the third row, third column).

**Foreground Refinement** Additional examples of images where the foreground refinement changes the segmentation are given in Fig. 4.10. Most errors in a GrabCut[149]

+ class model which can be corrected by a foreground refinement are undersegmentations. In the most extreme case, these undersegmentations can actually be empty, which the foreground refinement fixes. In all cases the segmentation after refinement is better than the segmentation before refinement, though the final segmentation may still have imperfections.

**Co-segmentation** We show additional qualitative co-segmentation results in Fig. 4.11. In general, co-segmentation works quite well, but in cases where part of the background is sufficiently different from the rest of the background the segmentation quality can suffer. Segmentation is also difficult at certain car parts, *e.g.* the wheels, since they look very different from the rest of the car. It is also difficult to properly segment the bottom of many cars, since the shadow of the car often looks similar to the foreground.

## 4.6 Conclusions

In this work we have presented a method for fine-grained classification which does not require part annotations at training time, setting a new state-of-the-art on the CUB-2011 and cars-196 datasets. We believe that developing such methods will be important for scaling up fine-grained recognition to an ever-increasing number of visual domains, in which it will be cost-prohibitive to annotate parts.

## 4.7 Acknowledgements

We thank Olga Russakovsky, Serena Yeung, Andre Esteva, Jon Brandt, Scott Cohen, and Brian Price for helpful feedback. This work was partially supported by Adobe and an ONR-MURI grant.


Figure 4.9: Additional visualizations for nearest neighbors with $\text{conv}_4$ features, which tend to preserve pose.

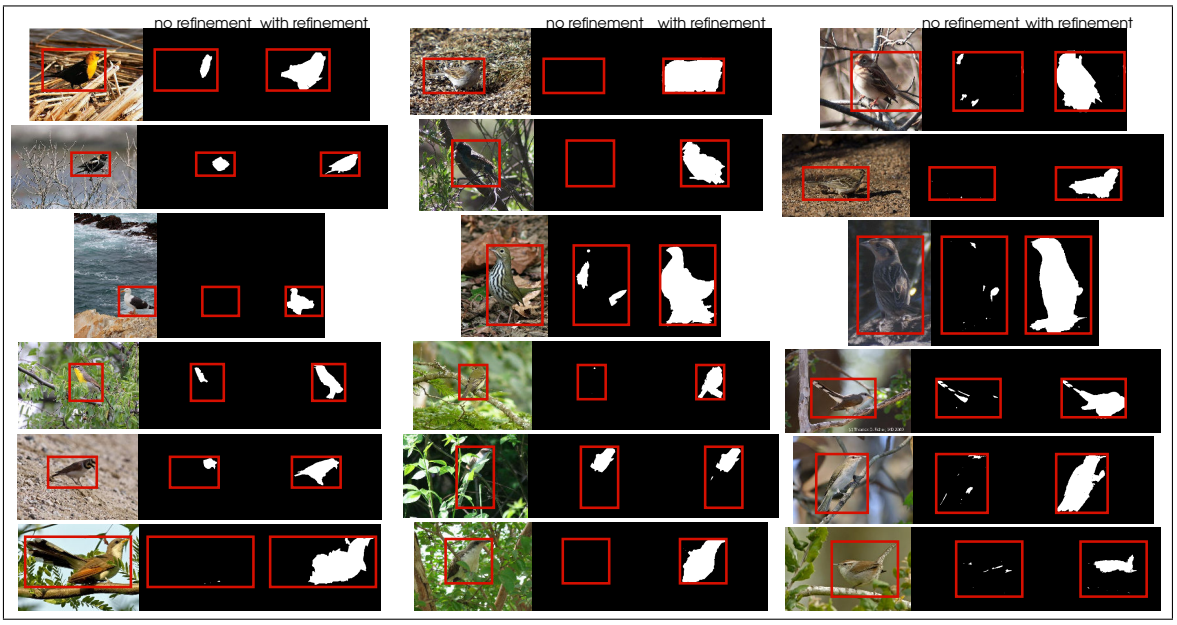



Figure 4.10: Additional visualizations for the effect of foreground refinement. Within each column of images, the first image is the original image, the second is the GrabCut+class model, and the third is GrabCut+class+refine.

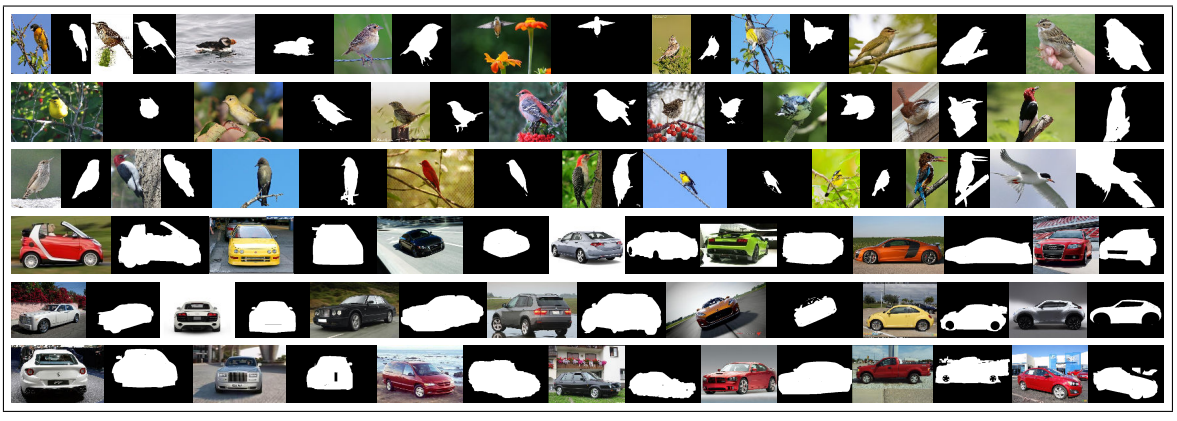


Figure 4.11: Additional visualizations of co-segmentation results. The last results in each row are failure cases.

# Chapter 5

# Using Fine-Grained Recognition to Study People and Society

The organization of this section is as follows: We first present the main results and text in Sec. 5.1. We then present extensive implementation details and supplemental results in Sec. 5.2. Due to the large number of figures and tables, we defer those to the end of this section, in Sec. 5.3. This project is joint work with Timnit Gebru, Yilun Wang, Duyun Chen, Jia Deng, Erez Aiden, and Fei-Fei Li, and an earlier version appeared in [63].

## 5.1 Main Results

Visual records of human life date back to cave paintings, predating written records by more than ten thousand years [153]. These paintings allow anthropologists to study the lives of early humans. Today, the photograph has become the primary source of imagery used to record daily life. More than 1.8 billion images are shared and uploaded to the internet each day [123]. Although these photos are rich sources of information about the human experience, computers inability to understand images has prevented the use of large-scale imagery for social analysis. Pioneering attempts by computer vision researchers [131, 137, 207, 13] have been limited in scale and scope, in contrast to computational social scientists who have analyzed culture using large

quantities of text from sources such as books [125] and social networks [91, 43, 67].

However, recent computer vision systems have started to perform some types of image analyses accurately and efficiently. In 2014, state-of-the-art computer vision algorithms began to classify everyday objects like animals, furniture, or vehicles in images with an accuracy rivaling that of humans [151]. These recent breakthroughs enable computational visual sociology, the use of automated image analysis for discovering and analyzing trends across society.

In this work, we use computer vision algorithms to analyze 50 million Google Street View images from 200 American citiesone image per 6.4 United States residents. In every image, we automatically detect cars and determine their make, model, body type, year, and trim (Fig. 5.1, A and B). Cars are a useful source of visual information because of their rigidity and subsequent ease of automated recognition in images [151], value as reflections of their owners personalities [38], and sheer frequency in street-level imagery. Performing this analysis by hand would take upward of 15 years (at 10 sec/image) and require expertise of all vehicles in common use. Using these detected cars, we analyze and predict zip code-level distributions of household income, crime, race, and voting patternsdata typically obtained from disparate and expensive sources.

**Detecting cars in 50 million images.** In order to train computer vision models for detection and classification of cars, we annotated 543,641 Street View images with the location of each car and labeled 69,562 car instances with one of 2,657 car categories. These categories form a nearly exhaustive list of all visually distinct car types used in the United States since 1990 at the make, model, body type, year, and trim level. To supplement this data, we also annotated 313,099 images of cars from retail websites with a category label.

To efficiently detect and classify cars in 50 million images while retaining high accuracy, we leverage deformable part models (DPMs) [57] for generic car detection and convolutional neural networks (CNNs) [97] for car classification. After optimizing our DPM implementation and augmenting it with a car location spatial prior learned from Street View images, we were able to detect cars in 3.5 million images per day

on our computing cluster, or 2 weeks for all images. Our CNN was trained on a weighted mixture of labeled Street View and retail car images, and we also performed multiple types of data augmentation to make our training data better resemble noisy detections in real-world Street View images.

To validate the end-to-end efficacy of our data collection and computational methods, we compared the distribution of cars we detect in Street View images with the distribution in Boston, Worcester and Springfield, MA (the three Massachusetts cities in our dataset), released by the Massachusetts DMV, the only state to release extensive vehicle registration data [124]. We measured the Pearson correlation coefficient between each detected and registered makes distribution across zip codes (Fig. 5.1, C and D). Twenty five of the top thirty makes have a Pearsons r correlation of r¿0.5 and classifying according to the 2011 national auto sales distribution [28] results in correlation r=0. Beyond Massachusetts, we measure the correlation between our detected car make distribution and the 2011 national distribution of car makes as r=0.97, verifying the efficacy of our approach.

**The automobile shapes the city.** From drive-in movie theaters to the advent of suburbs, cars have helped shape American cities and the lifestyles of their inhabitants [107]. What do cars detected from Street View images tell us about cities personalities?

Using purely visual data, we can answer diverse questions, ranging from which city has the most expensive cars (New York, NY - $11,820.62), highest percentage of Hummers (El Paso, TX - 0.39%), or highest percentage of foreign cars (San Francisco, CA - 60.02%). We can even estimate the likelihood of driving a particular type of car in a city: according to our data, San Francisco car owners are 4.5 times as likely to drive a Toyota Prius as those in Fairbanks, AK.

These car attributes can reveal aspects of a citys character that are not directly car-related. For example, our measurements show Burlington, Vermont to be the greenest city, with the highest average miles per gallon of any city in our dataset (average MPG=25.34). Burlington is also the first city in the United States to be powered by 100% renewable energy [23]. In contrast, we measured the lowest average

MPG for Casper, WY (average MPG=21.28). Wyoming is the highest coal-mining state in the US, producing 39 percent of the nations coal in 2012 and emitting the highest rate of CO2 per person in the country [6]. We aggregate these city-level statistics by state to discover patterns across the country. For example, by mapping our detected average MPG for each state (Fig. 5.2A), we can see that coastal states are greener than inland states, a finding that agrees with published carbon footprints [6] (r=-0.66 between MPG and carbon footprints).

We can also compare cities income-based segregation levels using car detections in images. After calculating the average car price for each GPS point, we measure the level of clustering of similarly priced cars using Morans I statistic [127] for 22 cities – chosen for having dense GPS sampling – in Fig. 5.3A (more segregated cities have a higher Morans I). We visualize statistically significant clusters of expensive and cheap cars using the Getis-Ord Gi statistic, a more local measure of spatial autocorrelation [64].

Chicago is the most segregated city, with two large clusters of expensive and cheap cars on the West and East side of the city respectively. Conversely, Jacksonville is the least segregated city with a Morans I only 33% as large as Chicagos and exhibits little clustering of expensive and cheap cars. As shown in Fig. 5.3B and Fig. 5.3C, Chicagos clusters of expensive and cheap cars fall in high and low income neighborhoods respectively. It is difficult to compare to other measures of segregation due to prior studies aggregation of cities into metro areas, in addition to other challenges. However, our results do agree with findings from the Martin Prosperity Institute [60], ranking Chicago, IL and Philadelphia, PA among the most segregated and Jacksonville, FL among the least segregated American cities.

**Wealthy people drive foreign cars.** The US government spends more than one billion dollars each year to collect demographic data such as income [135]. We predict zip code level median household income and education level for the two hundred cities in our dataset using the attributes of cars we detect in Google Street View images (Fig. 5.4, A and B). Our predictions correlate well with actual income values (r=0.70), and we can classify zip codes as being in the top or bottom 25% income quartiles

with 88.16% accuracy (Tab. S4). Though not as precise as official statistics from the U.S. Census, we show that automated prediction from visual imagery can be used to discover general trends quickly and cheaply.

Using our data, we can quantify the relationship between income and cars. A car has historically been indicative of its owners social class in the United States [107]. While in the 1900s simply owning a car indicated high income [107], social status is now associated with particular car characteristics like price [122]. Although we measured a moderate correlation between income and car price (r=0.44), we found that the car attribute that correlates highest with income is actually the percentage of foreign manufactured cars (r=0.47). According to our estimates, a 1% increase in the percentage of foreign cars (observed on the street) corresponds to both an increase of $1,285 in median household income and a likelihood of having a graduate degree by 0.66%. Experian Automotives 2011 ranking shows that all of the top 10 car models preferred by wealthy individuals were foreign, even when the car itself was comparatively cheap (*e.g.* Honda Accord or Toyota Camry) [129]. We discovered high percentages of Japanese and German cars in a zip code to be particularly good indicators of wealth (Pearson r=0.45 and 0.43 respectively): with Toyota and Honda having the highest correlation (r=0.42 for both).

**White people drive SUVs. Black people drive sedans.** The relationship between cars and race has been the source of speculation, stereotyping, and market research [4, 2]. How well can we predict a neighborhoods racial makeup by visually analyzing its cars? On the whole, our predictions correlate well with actual racial distributions (r 0.76 for all races) and we can classify zip codes in the top and bottom quartiles for each race with accuracy greater than 85% (Fig. 5.4C and Tab. S4). We use our data to test car-related racial stereotypes. It is claimed in popular culture that whites like Subarus [4]. Our data corroborates this claim: Jeep and Subaru are the most highly correlated brands with the percentage of whites (Pearson r=0.29 and 0.24 respectively), with a 1% increase in the number of Jeeps corresponding to an 8% increase in the number of white people.

The best predictor of the percentage of whites in a zip code is the proportion of

SUVs (Pearson r=0.31). Conversely, high proportions of sedans indicate zip codes with a low percentage of whites and a high percentage of blacks (Pearson r=-0.42 and r=0.44 respectively). The car brands most highly correlated with the percentage of blacks are Cadillac, Buick, Mercury, Chrysler, and Lincoln: all US manufactured cars. In accordance with the well-documented historical popularity of Cadillac amongst African American car owners [130], we found a high percentage of Cadillacs in a zip code to be the best indicator of a high proportion of blacks (Pearson r=0.50). We estimate that a 1% increase in the percentage of Cadillacs in a zip code predicts a 23% increase in the number of black people.

While blacks drive US manufactured cars, Asians prefer Asian made cars. We found the proportion of Toyotas in a zip code to be the best predictor of the percentage of Asians (r=0.63), followed by the percentage of Japanese cars (r=0.57): we estimate that owning a Toyota increases the probability of being Asian by 25%. Our visual analysis of Asians preference for Toyota is corroborated by market research [2].

**Democrats buy cars. Republicans buy trucks.** In preparation for elections, major political parties in the USA use data from consumer research firms to find out where to reach their potential constituents [104]. We predict Democratic and republican precincts from Street View images using voting results for the 2008 presidential election as ground truth [11], visualized in Fig. 5.5A. Our visual predictions have correlation r=0.67 with actual voting records, and classify the top and bottom quartiles ranked by the percentage of votes Obama with 88% accuracy (Tab. S4).

Next we investigate which car attributes are good predictors of Democratic and Republican precincts. We found the top predictor of Republican precincts to be the percentage of pickup trucks (r=0.48 and r=0.42 for crew cabs and extended cabs respectively), a result agreeing with market research firm studies [59]. According to our estimates, those who drive pickup trucks with crew cabs are 4.1% less likely to vote for Obama than those who do not, and 5.7% less likely to vote for Obama than those who drive sedans. The percentage of sedans, which we found to be the top predictor for the percentage of blacks in a zip code, is also the top predictor for precincts voting for Obama (r=0.48). According to exit polls, Obama received 95%

of the black vote in the 2008 election [80].

Surprisingly, the two most positively correlated car brands with votes for Obama are makes that were discontinued by General Motors: Oldsmobile (r=0.23), discontinued in 2004, and Pontiac (r=0.22), discontinued in 2009. We estimate that those owning an Oldsmobile are 3% more likely to vote for Obama than those who do not. Current studies such as [59] exclude discontinued cars. However, our results do agree with older studies, which indicate that Pontiacs are the American car of choice for Democrats [171].

**High crimes and vans.** There is an extensive history of drug dealers selling drugs from vans [8]. After using cars detected from Google Street View images to predict crime rates for nine categories of crime (ranging from robbery to homicide) in our 200 cities (visualized in Fig. 5.5B), we found one of the highest predictors of crime rates to be the percentage of vans (r=0.30 for total crime against people and properties). We can classify zip codes in the top and bottom quartile for total crime against people and property with an accuracy of 80.98% and 84.04% respectively (Tab. S4).

The best single predictor of unsafe zip codes in the two hundred cities is the number of cars per image (r=0.31 and r=0.36 for crimes against people and properties respectively). According to studies conducted by law enforcement, many crimes are committed in areas with a high density of cars such as parking lots [163], and some police departments are working with city planners to design neighborhoods with a lower number of parked cars on the street in order to reduce crime [3].

**Computational visual sociology.** Although cars are a single source of visual information, we have shown the sociological use that even this simple channel has in analyzing broad swathes of society. We anticipate further applications of this computational visual sociology in analyzing images of buildings, trees, and even people themselves, *e.g.* visually testing the broken window theory of criminology [187], or the relationship between neighborhood crime rates and levels of vegetation [101]. Computer vision, rapidly becoming more accurate, is poised to reveal the untapped potential of imagery in large-scale societal analysis.

## 5.2 Implementation

Here we present extensive implementation details supporting the results of Sec. 5.1 in addition to supplementary results.

### 5.2.1 Dataset

In this section we detail our efforts at acquiring the data collected for our study. This data is necessary in two respects: first, we require data in order to train our computer vision models that detect and classify cars, and second, we need a large number of images in order to do the analysis itself. This section proceeds by detailing how we obtained the list of car categories used for analysis, how we collected a large number of product shot images used to train our car classifier, how we collected the 50 million Street View images used in our analysis and annotated a subset for training and verifying our model, and concludes with a complete description of the metadata we have for each car category.

#### Car Categories

Before collecting data for our computer vision classifier, we must first decide which classes we are considering in the first place. Our ideal set of classes would contain every type of car in common use, separated into visually distinct categories. This is necessary because the information available when classifying cars is purely visual. To the best of our knowledge, no such list of car categories already exists, so we constructed our own.

We retrieved an initial list of 15,213 car types from the car website *Edmunds.com*, collected in August 2012. These car types are organized in a hierarchy, with levels of the hierarchy organized as make, model, year, body type, and trim, with cars generally becoming more visually similar as one goes deeper into the hierarchy, visualized in Fig. 5.6. This forms a generally complete list of all cars used commonly in the United States that were produced from 1990 onward. Throughout this document we use the term car to refer to all types of automobiles with four wheels, including sedans, coupes, trucks, vans, SUVs, etc., but not including *e.g.* semi-trucks or buses.

This list of categories is not visually distinct, *i.e.* several of these car types are actually indistinguishable from visual appearance alone. For example, most cars do not typically change appearance from year to year, only doing a visual refresh once every several years. Other times two cars might be listed as different car types but only differ by whether their engine runs on gas or diesel.

As a first step toward grouping these categories into a smaller number of visually distinct classes, we made tasks on Amazon Mechanical Turk (AMT) to determine whether certain pairs of the 15k car types were distinguishable. The interface is shown in Fig. 5.7. Within each task we gave six pairs of categories and the user was prompted to determine 1) if the two classes had any visual differences, and 2) if they were different, on which parts they differed. Within each task we had two pairs for which we already knew the correct answer (as determined by hand), and we required that each user on AMT get the answer for those pairs correct in order to count their response. Images for this task were acquired from the handful of example images that *Edmunds.com* provides, and the class pairs generated for these tasks were chosen to be ones where we suspected the classes might be the same based on shared metadata (make, model, year, etc.) or near-identical example images. We first grouped cars with the same make, model, body type and year. Thus, each pair in the tasks consisted of cars only differing at the trim level. Once visually indistinguishable trims within the same hierarchy were merged, the same task was applied at the year level to merge cars with no visual distinction across multiple years.

In practice, we found that these tasks on AMT had a high false negative rate – although when an AMT user said that a pair of classes were different, they were indeed different, they failed to spot the differences in many pairs of classes, which, though very similar to one another, were actually visually distinct. Therefore, the authors grouped together the remaining classes into visually distinct classes by hand, resulting in 3,141 categories of cars, with extremely subtle differences between these fine-grained categories. Fig. 5.8 shows two examples of classes with their constituent groups.

### Product Shot Images

In order to train any sort of model to recognize the cars in our dataset, we need a large number of images of these cars annotated with category labels. One option, commonly used in the computer vision community, is to do web image searches for each category and clean up the query images for each category by hand to ensure they actually contain the category of interest [45]. However, in our case we have so many classes that it is infeasible to do this manually, and since our car categories are so specific, it is similarly infeasible to crowdsource this to the untrained users of AMT or other similar services. Indeed, our initial attempts along these lines produced labels which were both costly and unreliable.

In order to scalably collect data to train classifiers with, then, we leverage data from websites where people sell cars. Specifically, in addition to the example images of each class we have from *Edmunds.com*, we crawled images from *cars.com* and *craigslist.org*, two sites where users are heavily incentivized to list the exact type of car they are selling. Although users of these websites are not car experts in general, they do have very detailed knowledge about their own cars, which is the key information we leverage. In the case of *cars.com*, car categories are represented in a very structured format, so after establishing a mapping from our categories to their format, we were able to simply scrape images for each category. For *craigslist.org*, we scraped posts from the cars+trucks listings of a variety of U.S. regions, then parsed the post titles to determine which category of ours the posts belonged to. In particular, we only kept images from posts where we were able to determine that the car belonged to exactly one of our classes. Since these images are all from websites with the purpose of selling cars, we call these product shot images.

Although these images will almost always contain a car from the correct category, one challenge remains: they may contain images that do not show all of the car. For example, these images may be from a very close-up angle or even from the interior of the car (Fig. 5.9). Thus, we filter out the images which do not contain one central car, viewed from the exterior, where all of the car is visible. Since this type of task is relatively simple, we can crowdsource it with AMT, using [158] for quality control. Our interface for this task is shown in Fig. 5.10.

As one final step of annotation, we collected a bounding box (an axis-aligned rectangle tightly enclosing the object of interest) around the car in each image. This is necessary so that our car classifier is trained only on visual information from the car itself and not on extraneous background. These were collected using the labeling method and UI of [165], but without the step for determining if there is more than one car in the image, since we know from the previous AMT task that each image contains exactly one prominent car.

Since images of some car categories are much more common than images of other categories, we chose to stop image annotation after approximately 200 images for a particular category. Our goal is to recognize as many car types as possible, and with limited resources it is more efficient to add images to categories we do not yet have many images for than to add them to categories for which we already have an excess of data. Our final set of car categories was determined by whether we had at least three disparate sources of data for the category, where one source of data corresponds to one post on any of the websites used. In the end, this filtering criterion resulting in keeping 2,657 of the car categories, making those the final set of categories used.

### Street View Images

In this section we describe our efforts in collecting a large number of images from Google Street View and how we annotated a subset of images for use in training a car detector and car type classifier. This process includes selecting which (latitude, longitude) GPS points to sample from Google Street View, getting images for each such point, annotating a subset of images for the presence of cars, and having car experts annotate cars with category labels.

**Getting GPS Points to Sample** The first step in getting images from Google Street View is determining which (latitude, longitude) points, which we colloquially call GPS points, should be sampled. For each of the 200 cities (two largest in each state and next 100 largest in the United States as determined by population, see Tab. 5.1 for a complete list), we sample potential points in a square grid with length 20km, centered on one point known to lie within the city, with spacing 25 meters

between points. We reverse geocode each of these potential points to determine whether it lies within the city of interest and how far away it is to the nearest road. We keep all points within 12.5 meters of the nearest road. For a handful of cities this did not provide reasonable coverage, so we augmented these points with GPS points taken along road data provided by the U.S. Census Bureau [7].

**Sampling Images from Street View** For each GPS point, we attempt to sample 6 images from Street View, one for each of 6 different camera rotations. This was done via browser emulation and requires only the latitude and longitude of each point. Images retrieved in this way are not yet directly usable: due to the particular way that images in a spherical panorama are stored (a equirectangular projection), they appear warped. However, this transformation is reversible, which we therefore do before any subsequent tasks (see Fig. 5.11 for an example). Each such Street View image we get has dimensions 860 by 573 pixels and a horizontal field of view of roughly 90 degrees. Since the horizontal field of view is larger than the viewpoint change between images of the 6 images per GPS point, the images we obtained have some overlapping content.

In total, we collected 50,881,098 Google Street View images for our 200 cities. The distribution of images across cities is shown in Tab. 5.1. These images were primarily collected between June and December of 2013 with a small fraction (3.1%) collected in November and December of 2014, corresponding to the GPS points from the U.S. Census Bureau.

**Annotations on Amazon Mechanical Turk** In order to train a generic car detector, any computer vision algorithm needs access to training data, which in our case are bounding boxes in Street View images. We use the system of [165] to collect bounding box annotations in a subset of our Street View images. In order to make this more efficient, we first filter out all images containing either zero or more than 10 cars via AMT, using the same interface and setup described in Sec. 1.2. We randomly selected a subset of 399,331 images from our entire collection of Street View images for annotation in this way. We found that 26.6% percent of images were annotated

as having no visible cars and 12.4% had more than 10 cars. The distribution of the number of cars in the remaining images is shown in Fig. 5.12.

In Fig. 5.13(a) we show a plot of bounding box size versus location. Cars in Street View images tend to occupy more space in the image the lower they are, which agrees with intuition that cars further down in the image are closer to the camera and therefore appear larger. Fig. 5.13(b) shows a heatmap of bounding box location for cars in Street View. Most cars are located near the horizon line since that space of the image occupies more 3D space and thus has more space for cars to appear. Above the horizon line there is a sharp dropoff in the distribution of cars.

**Expert Class Annotations** We also collected car category annotations for a subset of the Street View cars annotated with bounding boxes. While this will partially be useful as additional data to train a car classifier with, it also provides a way to quantitatively evaluate how well our classifier works on Street View images. In contrast to the product shot images, in Street View we have no prior information on the car category for any particular image or bounding box. Therefore, we hired expert car annotators to provide category annotations. Expert annotators were solicited primarily via Craigslist ads. Those who were interested in performing our task were first asked to annotate cars in Street View images for one hour, and only those who could annotate at a speed of 1 car per minute and a precision of at least 80% were allowed to annotate further. In total we had 110 expert human annotators who worked for a total of approximately two thousand hours.

The annotation task itself proceeded hierarchically: Fig. 5.14 shows the user interface for the task. Given a Street View bounding box, annotators were first asked to give the make of the car (Fig. 5.14(A)). After performing a selection, annotators were presented with a list of body types for that make (Fig. 5.14(B)). Then, experts were presented with options for selecting the model and a list of trims and years associated with each model, grouped together to preserve visual distinguishability as per Sec. 1.1.

Since differences between categories can be extremely subtle at that final level, we also provided example images from each trim and year grouping for the annotators

benefit (Fig. 5.14(C)). At any point in the process the annotator could declare that he or she did not have enough information to make a selection, so each annotation at this finest level of detail represents a confident selection by a car expert. Furthermore, in order to make the best use of the experts' time, we limited the cars shown to them to only include cars whose bounding boxes exceeded 50 pixels in height, as very small images typically do not contain enough visual information to discriminate fine levels of detail. This accounted for 32.89% of possible car bounding boxes. We collected a total of 69,562 car category annotations in this way.

### Car Metadata

In addition to the images, category labels, and bounding boxes, we also have additional metadata about each class, listed below.

- Make: The make of the car, of 58 possible makes. The makes we consider are: Acura, AM General, Aston Martin, Audi, Bentley, BMW, Buick, Cadillac, Chevrolet, Chrysler, Daewoo, Dodge, Eagle, Ferrari, Fiat, Fisker, Ford, Geo, GMC, Honda, Hummer, Hyundai, Infiniti, Isuzu, Jaguar, Jeep, Kia, Lamborghini, Land Rover, Lexus, Lincoln, Lotus, Maserati, Maybach, Mazda, McLaren, Mercedes-Benz, Mercury, Mini, Mitsubishi, Nissan, Oldsmobile, Panoz, Plymouth, Pontiac, Porsche, Ram, Rolls-Royce, Saab, Saturn, Scion, Smart, Subaru, Suzuki, Tesla, Toyota, Volkswagen, and Volvo.

- Model: The model of the car, of 777 possible models.

- Year: The years the car was made in. Since cars might not change appearance over a small number of years, this is typically a range of years rather than a single year. The minimum year in our dataset is 1990, and the maximum year is 2014.

- Body Type: The body type of the car. The 11 possible values are: convertible, coupe, hatchback, minivan, sedan, SUV, truck (regular-sized cab), truck (extended cab), truck (crew cab), wagon, and van.

- Country: The country of the car manufacturer. The 7 possible countries are: England, Germany, Italy, Japan, South Korea, Sweden, and USA.
- Highway MPG: The typical miles per gallon of the car when driven on highways. If a class contains cars with multiple years, it is annotated with the highway MPG of the oldest car in the group.
- City MPG: The typical miles per gallon of the car when driven on non-highway streets.
- Price: the price of the car in 2012.

This metadata was all acquired via *Edmunds.com* in August 2012, with some missing data (a handful of car prices) filled in by car experts afterward. In cases where a class consists of multiple visually indistinguishable types of cars, the metadata was chosen from the chronologically first car in the set.

#### Dataset Summary

Tab. 5.2 provides a summary of the annotations collected for both product shot and Street View images, which we split into training (50%), validation (10%), and test (40%) sets for use in training our car detector and classifier. Fig. 5.15 shows relative class distributions for these two data sources. We have a much more uniform distribution over product shot images, since we have a great deal of control over how many images we have of each class, while cars in Street View images tend to follow a much more skewed distribution reflecting the actual frequency of cars on the street.

### 5.2.2 Detecting and Classifying Cars

#### Detection

In computer vision, detection is the task of localizing objects within an image, and is most commonly framed as predicting the (x, y, width, height) coordinates of an axis-aligned bounding box around an object of interest. The central challenge for our work is designing an object detector that is 1) fast enough to run on 50 million

images within a reasonable amount of time, and 2) accurate enough to be useful for analysis. Our computation resources consisted of 4 Tesla K40 GPUs and 200 2.1 GHz CPU cores.

Current state of the art methods in generic object detection, *e.g.* R-CNNs [65] first generate a large number of unsupervised region proposals [173] where an object might be, then classify each region with a convolutional neural network to determine the category of the object. By our estimates, though, an R-CNN takes roughly 20 seconds to run on a single Street View image, meaning that running on all 50 million Street View images would take 31 GPU years.

Instead, we turned to the previous state of the art in object detection, deformable part models (DPMs) [57]. For DPMs, there are two main parameters that influence the running time and performance, which are the number of components and the number of parts in the model. Tab. 5.3 provides an analysis of the performance/time tradeoff on our data, measured on the validation set. Based on this analysis, using a DPM with a single component and eight parts strikes the right balance between performance and efficiency, allowing us to detect cars on all 50 million images in two weeks. In contrast, the best performing parameters would have taken two months to run and only increased average precision (AP) by 4.5.

In order to take advantage of the distribution of cars in Street View images, we also introduce a prior on the location and size of predicted bounding boxes. Concretely, we model the distribution of Street View cars in images as a histogram over three variables: the $x$ coordinate of the center of the bounding box, the $y$ coordinate of the center, and $log(area)$. We divide each variable into 20 bins, for a total of 8,000 bins in the histogram, and estimate the probability of each bin from bounding box statistics in the Street View training data, adding a pseudocount of 1 in each bin for regularization. With $P(x, y, \log(area))$ denoting this histogram, we augment each DPM detection score by:

$$score^{DPM+LOC} = score^{DPM} + \alpha \log\left(P(x, y, \log(area))\right) \tag{5.1}$$

where $\alpha$ is a weight on the location prior, determined by the validation set. This

location prior improves detection AP on the validation set by 1.92 at a negligible cost. Fig. 5.13(a) visualizes this prior.

During analysis, these scores are converted into estimated probabilities via isotonic regression [15], learned on the validation set. Isotonic regression learns a probability for each detection score subject to a monotonicity constraint. Concretely, after sorting $n$ validation detection scores $s_1, \ldots, s_n$ such that $s_i \leq s_{i+1}$, and with $y_i$ a binary variable denoting whether detection $i$ is correct (has Jaccard similarity of at least 0.5 with a ground truth car bounding box), isotonic regression solves the following optimization problem:

$$\begin{array}{ll} \underset{p_1,\ldots,p_n}{\text{minimize}} & \sum_{i=1}^{n} \|y_i - p_i\|_2^2 \\ \text{subject to} & p_i \leq p_{i+1}, \qquad 1 \leq i \leq n-1 \end{array} \tag{5.2}$$

Given a new detection score, a probability is estimated by linear interpolation of the $p_i$. We plot the learned mapping from detection scores to probabilities in Fig. 5.16.

A number of further details need to be considered when training and running this car detector in practice. First, we choose to detect only cars that are 50 pixels or greater in width and height, since we need the images given to our car classifier to have sufficient resolution and detail in order to differentiate the different types of cars. Similarly, we use as training instances only the cars which are greater than 50 pixels tall. Also, recognizing the model capacity of a single component DPM, we train on a subset of 13,105 bounding boxes, which brings training time down from a week (projected) to 15 hours and has resulted in negligible changes to performance. The numbers reported in this section are done with a detection threshold of -1.5 (applied before the location prior). At test time, after applying the location prior (which lowers detection decision values), we use a detection threshold of -2.3, which cuts down on the average number of bounding boxes per image we need to classify at test time from 7.9 to 1.5, reduces AP by 0.6 (66.1 to 65.5), and only reduces the probability mass of all detections in an image from 0.477 to 0.462, a 3% drop. Fig. 5.17 shows examples of car detection with our model. Qualitatively, the detections are quite good, with inaccurate detections generally having low estimated probabilities of being correct. The AP of our chosen model on the test set is 65.7, and its precision recall curve is

visualized in Fig. 5.18. To calculate chance performance we use a uniform sample of bounding boxes greater than 50 pixels in width and height.

### Classification

We use a convolutional neural network [103] (CNN) with an architecture of Krizhevsky *et al.* [97] to classify each car detection into one of the 2,657 car categories. Although CNNs generally work well when trained on data from a similar distribution as the test data (in our case, Street View images), due to the large number of categories in our analysis and the comparative cost of obtaining category labels in Street View images, we trained our CNN on a combination of Street View images and the more plentiful product shot images. We made three modifications to the traditional CNN training procedure in order to better fit our setting.

First, taking inspiration from domain adaptation, we approximated the WEIGHTED method of Daumé [44] by duplicating each Street View training example 10x at training time. This makes the number of product shot images and Street View images roughly equal at training time and prevents the classifier from overfitting on product shot images.

Another large difference between product shot and Street View images is image resolution, *i.e.* cars in product shot images tend to occupy a much larger number of pixels in the image. To compensate, we measured the distribution of bounding box resolutions in Street View training images, then while training our classifier we dynamically downsize each input image according to this distribution and rescale it to fit the input dimensions of the CNN. Resolutions are parameterized by the geometric mean of the bounding box width and height, and the probability distribution is given by as a histogram over 35 different such resolutions. The largest resolution is 256, which is the input resolution of the CNN, and thus corresponds to not changing the resolution.

One further challenge when doing classification in Street View images is that our input is noisy detection bounding boxes. This stands in contrast to what would otherwise be the default for training a classifier – ground truth bounding boxes that are tight around each car. To tackle this challenge, we first measured the distribution

of the intersection over union (IOU) overlap of bounding boxes produced by our car detector with the detection validation data. Then, for each Street View training image given to the CNN, we randomly sample the actual image region used according to this IOU distribution, in effect simulating detections as input to the CNN and making the training images more similar to the types of images we encounter during testing.

At test time, we use the whole bounding box of the detection as input to the CNN, and do a single forward pass of the CNN to get the softmax probabilities for each car category. In practice, we only keep the top 20 predictions, since storing a full 2,657-dimensional floating point vector for each bounding box is prohibitively expensive in terms of storage. These top 20 predictions account for 85.5% of the probability mass of the predictions of the softmax layer, on average. We also note that, after extensive code optimization to make this classification step as fast as possible, we are primarily limited by the time spent reading images from disk, especially when doing classification on multiple GPUs simultaneously.

We show confusion matrices for body type, country, make, and model on the test set in Fig. 5.19, Fig. 5.20, and Fig. 5.21, and Fig. 5.22, respectively, generated by taking the prediction as the body type, country, or make of the top-scoring car category, respectively. Body type misclassifications tend to be among similar body types, *e.g.* the most frequent misclassification for "coupe" is "sedan", and the most frequent misclassification for trucks with a regular cab are trucks with an extended cab. Misclassifications at the country and make level, since no classes are necessarily visually similar, tend to be misclassifying the country/make of a car as a more popular country/make. For example, most errors at the country level occur by misclassifying the country as either "Japan" or "USA", the two most popular countries. Due to the large number of classes, the only clear pattern in the model-level confusion matrix is a strong diagonal, indicative of our correct predictions.

### 5.2.3 Experimental Details

#### Comparison to Massachusetts and National Ground Truth

Vehicle data for Massachusetts is available from the Massachusetts Vehicle Census [124] and contains anonymized zip code and model information for all vehicles registered in Massachusetts between 2008 and 2011. Since our comparison with this data was done at the make level, aligning their list of car classes and ours entailed only aligning the list of makes, which was done by hand. We perform our experiments on the intersection of detected and registered makes resulting in a total of 45 makes.

To calculate the distribution of car makes in each zip code, we compute the expected number of each of the 2,657 car classes across the zip code, then simply use the make metadata associated with each car class to compute the expected number of cars for each make within the zip code. Since the expected number of instances of a particular car across a zip code is the sum of the expected number of instances of the car across all images within that zip code, the problem reduces to calculating this expectation for a single image.

With $I$ an image and $c$ one of the 2,657 classes, we decompose the expectation for a single image as

$$\mathbb{E}[\#\text{class } c|I] = \sum_{\text{bbox } b} P(\text{car}|b, I)P(\text{class } c|\text{car}, b, I) \tag{5.3}$$

where we are summing over all bounding boxes $b$ for generic cars detected by our model. We model $P(\text{car}|b, I)$ using isotonic regression (described in Sec. 2.1), and $P(\text{class } c|\text{car}, b, I)$ corresponds to the conditional probabilities output by the softmax layer of our CNN classifier.

To obtain the percentage of each make, we aggregate these category-level expectations by car make and compute percentages using the make-level expectations. We follow this setup in all other experiments, and note that a similar procedure can be used to find distributions or expected values for any other car attribute, including car price, miles per gallon, and country of origin.

For experiments with Massachusetts ground truth data, the Pearson coefficient

for each make is calculated by taking the percentage of that make in one zip code as a single data point. We chose zip codes with greater than 5,000 inhabitants and 500 detected cars in the three Massachusetts cities in our dataset (Boston, Springfield, Worcester), resulting in a total of 37 zip codes. For all experiments, we used registration records that were valid during the second quarter of 2010. As outlined by [124], registration datasets are most complete at that point.

Fig. 5.23 shows the Pearson correlation coefficient between the distribution of registered and detected cars across zip codes for all makes that are in the intersection of our dataset and Massachusetts registration data: in contrast to Fig. 5.1C, this shows all makes instead of the top 30. In addition to comparing the distribution of each registered and detected make across zip codes, we performed two additional experiments. First, we compared the distribution of all detected and registered makes per zip code, computing the Pearson correlation coefficient for each zip code (Fig. 5.24). All zip codes have correlation greater than 0.8. In contrast, classifying according to the national distribution only results in correlations greater than 0.45 for all zip codes. Next, we compared the total distribution of registered and detected cars in all 37 zip codes (Fig. 5.25) and measured a correlation coefficient of 0.94. Prediction using the national sales distribution instead of our approach only has correlation 0.82.

### City-Level Attributes

We calculated city-level attributes by taking the expectation of the various car attributes across all images in each city. In Fig. 5.26 we show additional maps for a subset of other attributes of cars we have meta-data for: average car price and the percentage of foreign cars, BMWs, Chevrolets, Toyota Prius, and Ford F-150s.

### Segregation

Moran's I statistic, which we use to estimate income-based segregation, is defined as [127]:

$$I = \frac{N \sum_{i,j} w_{i,j}(x_i - \bar{x})(x_j - \bar{x})}{\sum_{i,j} w_{i,j} \sum_i (x_i - \bar{x})^2} \tag{5.4}$$

where each $x_i$ is a distinct GPS point, $\bar{x}$ is the average of $x$, computed per city, and $w_{i,j}$ is a weight describing the distance between points $i$ and $j$, which we take to be the square of the inverse of the Euclidean distance between points. A Moran's I of 1 indicates total segregation, -1 indicates no segregation at all, and 0 indicates a random pattern of segregation.

Cities used for segregation analysis were chosen to be 22 cities with good sampling coverage of the top 30 most populated cities in the United States. We can visualize a city's segregation using the Getis-Ord statistic [64, 136], defined at a point $i$ as

$$G_i^* = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{X}\sum_{j=1}^{n} w_{i,j}}{S\sqrt{\frac{n\sum_{j=1}^{n} w_{i,j}^2 - \left(\sum_{j=1}^{n} w_{i,j}\right)^2}{n-1}}} \tag{5.5}$$

where

$$\bar{X} = \frac{1}{n}\sum_{j=1}^{n} x_j \tag{5.6}$$

is the sample mean,

$$S = \sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2 - \bar{X}^2} \tag{5.7}$$

is the sample standard deviation, there are $n$ points, and $w_{i,j}$ is a real-valued weight describing the proximity points $i$ and $j$. In practice, we use the optimized hot spot analysis tool of ArcGIS to determine appropriate weights for our analysis.

In Fig. 5.27 we show plots of the Getis-Ord $G_i^*$ statistic for the other cities used in the segregation analysis, calculated on the average car price within each GPS point. We observe that Chicago exhibits higher amounts of car price-based clustering, Jacksonville exhibits little local clustering at all, and the 20 other cities have some amount of price-based clustering in between the two extremes.

**Challenges While Comparing to Prior Work** We faced the following challenges while comparing our segregation results to prior work.

- Segregation studies such as [145, 60] combine cities into metro areas (*e.g.* Chicago-Naperville-Joliet instead of Chicago) while performing their analyses. Because

we only use data for 200 of the most populous cities in the U.S., we cannot combine cities into the same metro areas those studies used for segregation analysis, and thus cannot compare with those particular studies.

- Most segregation studies use data provided by the U.S. census that is aggregated at different spatial units, *e.g.* census tracts. On the other hand, we use GPS data sampled every 25 meters. As outlined in [145], segregation analyses performed at different levels of data aggregation can come to different conclusions.

### Correlation and Prediction Methodology

In all of our correlations and predictions, unless otherwise noted, we use the following set of 88 car-related attributes:

- The average number of detected cars per image
- Average car price
- Miles per gallon (city and highway)
- Percent of total cars that are hybrids
- Percent of total cars that are electric
- Percent of total cars that are from each of seven countries
- Percent of total cars that are foreign (not from the USA)
- Percent of total cars from each of 11 body types
- Percent of total cars whose year (selected as the minimum of possible year values for the car) fall within each of five year ranges: 1990-1994, 1995-1999, 2000-2004, 2005-2009, and 2010-2014.
- Percent of total cars whose make is each of 58 makes in our dataset

For prediction, we randomly partitioned the zip codes in five sets, iteratively training a model on four of the parts and predicting on the held out set. The model itself varies depending on the variable being predicted; see subsequent sections for details. Regardless of the model, we normalize the features to have zero mean and unit standard deviation (parameters determined on the training set of four parts). We furthermore clip predictions to be within the range of the current training data, preventing predictions from becoming too extreme.

In all experiments at the zip code level we restricted the zip codes used to be ones with a population of at least 5,000 and at least 500 detected cars, which reduces the number of zip codes under consideration from 3,068 to 2,430, mostly as a result of the restriction on the number of detected cars. Classification of zip codes in the top or bottom quartile is done by taking the output of our predictor (typically some type of regressor), and thresholding based on the median of the predictions.

Estimation of the increase in some demographic variable as the result of a 1% increase in one of our car attributes is done via a least squares fit of the target variable in response to the car attribute, where the increase is given by the slope of the least squares line.

### Income

Median household income data was obtained from the American Community Survey (ACS) [1], and was collected between 2008-2012. We used census variable `B19013_001E`, "Median household income in the past 12 months (in 2013 inflation-adjusted dollars)". Correlations between median household income and each of our car attributes are given in Tab. 5.4. Prediction was done via ridge regression, cross-validating the regularization parameter with 41 possible values spaced evenly in logspace between $10^{-5}$ and $10^{5}$, picked via the best cross-validated RMSE (5 folds).

The five car attributes that correlate most positively with median household income are %Foreign (r=0.47), %Country: Japan (r=0.45), Price (r=0.44), %Make: Lexus (0.44), and %Country: Germany (r=0.43). The five car attributes that correlate most negatively with median household income are %Country: USA (r=-0.47),

%Year: 1995-1999 (r=-0.42), %Make: Buick (r=-0.40), %Make: Oldsmobile (r=-0.40), and %Make: Dodge (r=-0.38).

### Education

Education data was also obtained from the ACS, and splits education levels into the following mutually exclusive categories (census codes in parentheses):

- Less than high school graduate (`B06009_002E`)
- High school graduate (includes equivalency) (`B06009_003E`)
- Some college or associate's degree (`B06009_004E`)
- Bachelor's degree (`B06009_005E`)
- Graduate or professional degree (`B06009_006E`)

We show correlations between each of our car attributes and these education levels in Tab. 5.5, Tab. 5.6, Tab. 5.7, Tab. 5.8, and Tab. 5.9, and the five car attributes that correlate most positively and most negatively with each race are given in Tab. 5.10. Predictions for education levels were done via multinomial logistic regression. A plot of our predictions vs actual education levels in Fig. 5.28.

### Race

Racial demographic data was also obtained from the ACS [1], and corresponds to census codes `B02001_002E` ("White alone"), `B02001_003E` ("Black or African American alone"), and `B02001_005E` ("Asian alone"). Correlations between our car attributes and the percentage of each race considered (White, Black, and Asian) are given in Tab. 5.11, Tab. 5.12, and Tab. 5.13, respectively. The five car attributes that correlate most positively and most negatively with each race are given in Tab. 5.14. Prediction for race was done via multinomial logistic regression, adding in a fourth category of race encompassing all others not included by the three in our analysis. Plots of predicted vs actual amount of each race are given in Fig. 5.29.

### Voting

Data for the 2008 U.S. presidential election was provided to us by the authors of [11] and consists of precinct-level vote counts for Barack Obama and John McCain. For all of our analyses, we ignore votes cast for any other person, *i.e.* the count of total votes is determined solely by votes for Obama and McCain. We also restrict our attention to only those precincts with at least 1,000 votes and 100 detected cars. We visualize this raw data in Fig. 5.30. Two aspects of this data deserve special note. First, the vast majority of precincts in our data have greater than a 50% share of votes for Obama. This can be attributed both to the overall larger share for Obama in the country for the 2008 election and to the fact that we are only considering precincts in fairly major cities, which tend to favor candidates from the Democratic party. The second aspect of note is the large number of precincts which have very high ($\geq 95\%$) share of votes for Obama. Although not true of all precincts, a large portion of these precincts have high concentrations of African Americans, who overwhelmingly voted for Obama in the 2008 election.

In contrast to the analyses in the preceding sections, for our political analysis all features were made at the precinct, rather than zip code, level. Otherwise, our experimentation is the same as in the other analyses. We used ridge regression to predict the percentage of votes cast for Obama in each precinct, cross-validating the regularization parameter as in the other experiments. We also attempted to use a generalized linear model with a logit link function to fit the data, but this ultimately had less predictive power than simple ridge regression.

We show correlations between %Obama and all of our car-centric variables in Tab. 5.15, and plot our predictions versus actual voting percentages in Fig. 5.31. The five car attributes that correlate most positively with Obama's percent of votes are Body Type: Sedan (r=0.48), #Cars/Image (r=0.37), MPG Highway (r=0.33), and MPG City (r=0.26). The five car attributes that correlate most negatively are Body Type: Crew Cab (r=-0.48), Body Type: Extended Cab (r=-0.43), Body Type: Regular Cab (r=-0.30), Price (r=-0.28), and Body Type: SUV (r=-0.22).

### Crime

We obtained crime data from [5]. Crime data is difficult to obtain on a zip code level since it is kept track of by local governments and not aggregated by any authoritative source, motivating our use of a less authoritative but complete source [5]. Crime data is aggregated into 9 different categories:

- Aggravated Assault
- Burglary
- Crime Against People
- Crime Against Property
- Homicide
- Larceny
- Motor Vehicle
- Robbery
- Rape

The amount of each type of crime in each zip code is discretized into one of 10 bins, corresponding to different multiples of the national crime rate for that type of crime. The ten bins correspond to the following ranges (all are multiples of the national average): [0, .2] (bin 1), [.2, .33], [.33, .5], [.5, .75], [.75, 1.25], [1.25, 2], [2, 3], [3,5], [5,10], and [10,$\infty$] (bin 10).

Correlations and prediction are done on top of the bin values for each zip code. We found that these were better than crime rates themselves to do analysis and prediction on because they provide for a more uniform distribution of crime. Crime rates themselves have a very skewed distribution, with the majority of crime rates between 0 and 1 (as a multiple of the national average), but including some zip codes which have crime rates 10 times the national average. Prediction was done using ridge regression in an identical setup as was used for income prediction. Quantitative measures of our accuracy in predicting each type of crime are given in Tab. 5.16.

### 5.2.4 Sources of Error

One source of error lies in the quality of the Street View images. Images from Street View may have image stitching artifacts, have street names in the images, and cars in Street View images might not be entirely visible, either due to occlusions with other objects or simply being cut off by the edge of the image. These factors all make car detection and classification in Street View images more difficult. However, images used in our performance analysis are also subject to these issues, and thus our demonstrated performance holds despite these challenges.

Another source of error consists of biases in sampling, consisting of either sampling at certain roads or regions preferentially (due to having incomplete or out of date information about roads in cities) or sampling images taken at different dates. Although being able to properly account for these types of errors would undoubtedly improve and strengthen the analysis, these factors do not diminish or weaken the results we already have, and can be considered a source of noise in the data. For example, addressing these limitations might result in stronger correlations and lower $p$-values, but unless these are systematic errors across the entire United States, does not affect the validity of the results presented in this work.

The data used for demographic, crime, and voting analysis were not collected at the same point in time as images taken in Street View, and thus any drift in those sources over time is also a potential source of error. For example, the data for voting was from the 2008 presidential election, but the majority of Street View images were taken after 2008. While this is an unrecoverable source of error, it is primarily a problem when such statistics change rapidly over time, which is rarely true, and thus we believe this type of error has minimal impact on our analyses.

## 5.3 Figures and Tables

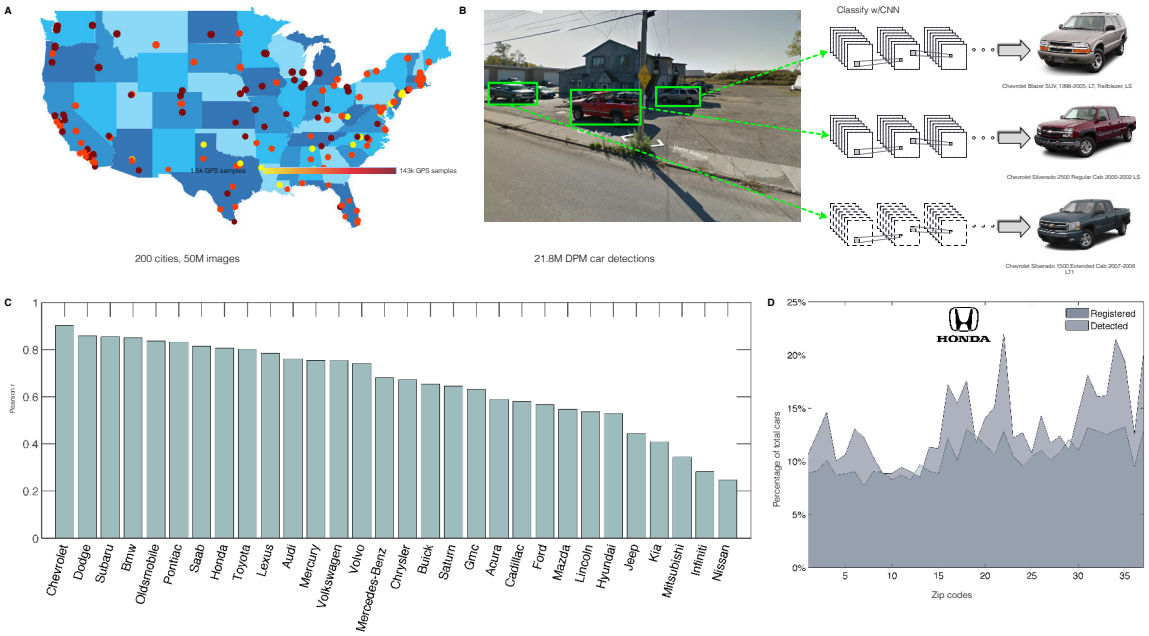



Figure 5.1: We learn about 200 cities using 50 million Google Street View images. (A) The two hundred cities we gather images for. (B) In every image we detect cars using computer vision algorithms based on deformable part models (DPM) and convolutional neural networks (CNN) and categorize them into one of 2,657 classes of cars, revealing information about the city the image was taken from. (C) The correlation between the distribution of detected and registered car makes across zip codes in the three cities in Massachusetts. Twenty five of the top thirty makes have correlation r¿0.5 while classifying randomly or according to the 2011 national car sales distribution [107] results in r=0. (D) The distribution of a single car make, Honda, across zip codes in Springfield, Worcester and Boston, MA, detected using our algorithm and compared with data from the Massachusetts DMV [124]. We identify the 37 zip codes in our dataset with numbers ranging from 0-36 (x-axis).

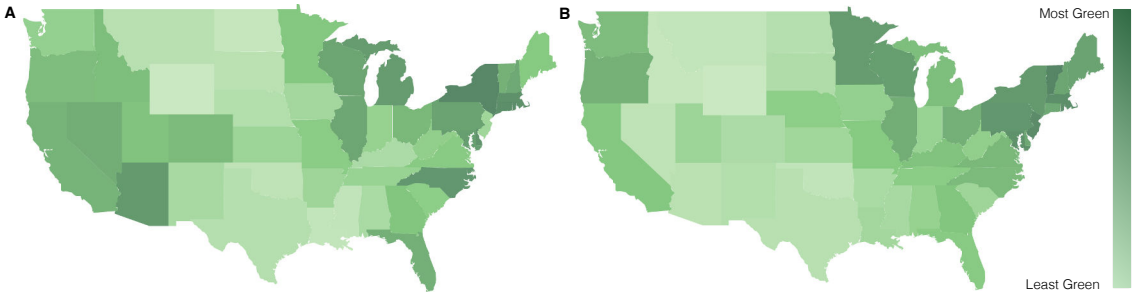



Figure 5.2: Attributes of cars detected in Google Street View images can be aggregated at the state level and mapped to discover patterns. (A) A map ranking each states carbon footprint from the transportation sector in 2012 using data from [6]. (B) A map ranking each states average miles per gallon (MPG) calculated from car attributes detected from Google Street View images from 200 cities. We measure a Pearson correlation coefficient of -0.66 between 2012 state level carbon footprint from the transportation sector and our calculated state average MPGs. Both maps show that coastal states are greener than inland ones.

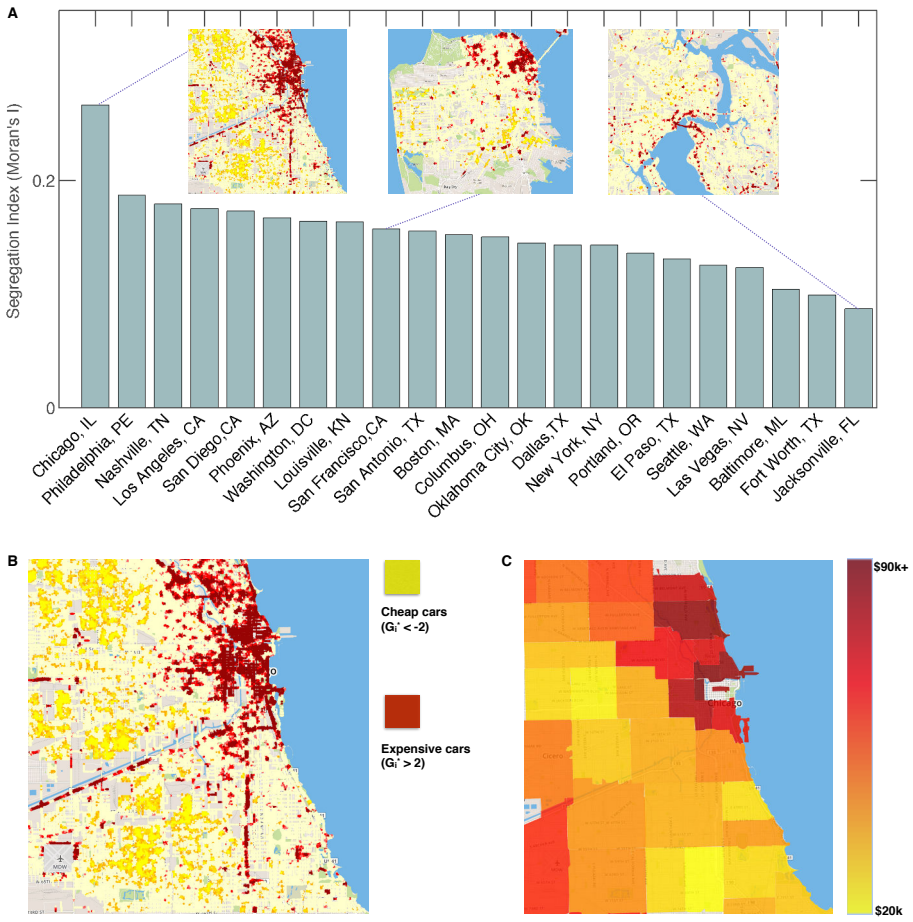



Figure 5.3: Cars detected in Google Street View images give information about city neighborhood characteristics. (A) Twenty two out of thirty of the most populated cities in the United States ranked by segregation of car price. Our segregation index is Morans I statistic [127]. Insets show maps of statistically significant clusters of cars with high prices (red), low prices (yellow) as well as no statistically significant clustering (white) for the cities of Chicago, IL, San Francisco, CA and Jacksonville, FL respectively using the Getis-Ord Gi* statistic [64]. Chicago has large clusters of expensive and cheap cars whereas Jacksonville shows almost no clustering of cars by price. (B) Expensive/Cheap clusters of cars in Chicago. (C) Zip code level median household income in Chicago. Large clusters of expensive cars are in wealthy neighborhoods whereas large clusters of cheap cars are in un-wealthy neighborhoods.

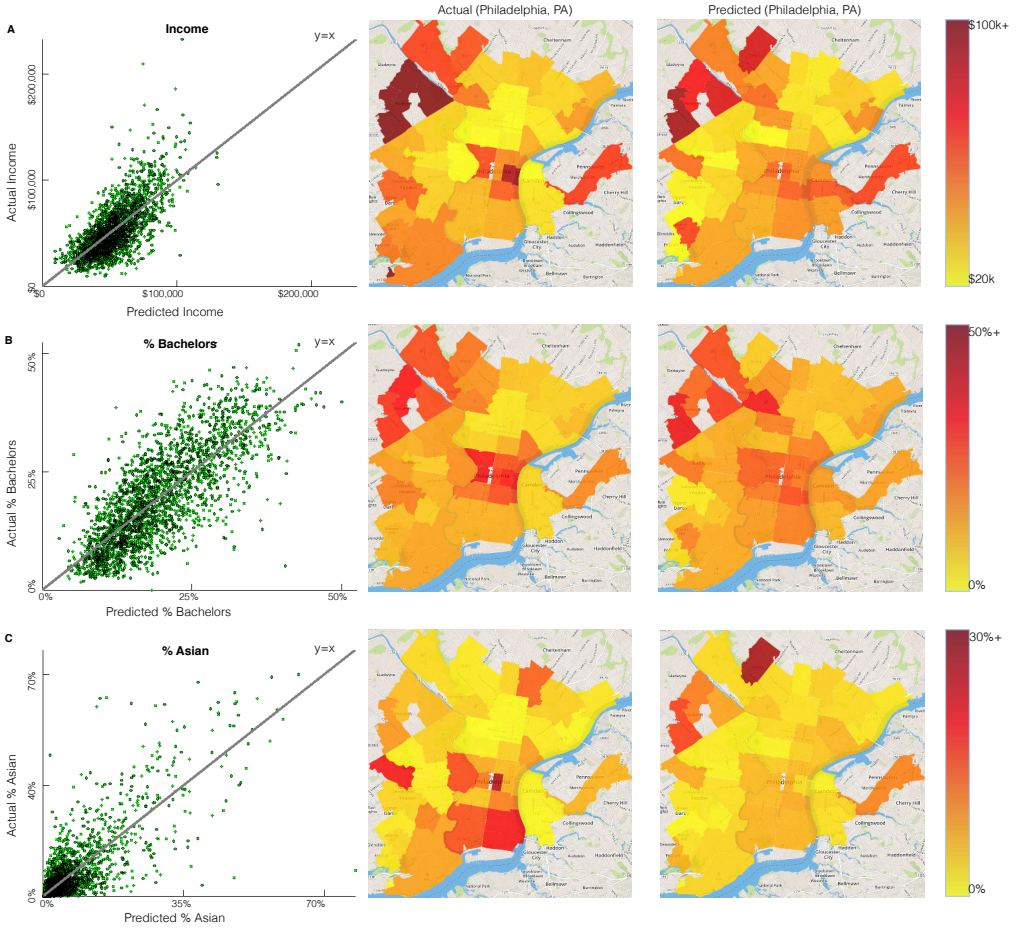



Figure 5.4: Cars detected in Google Street View images can be used to predict demographic variables such as income, education and race. The first column of (A), (B) and (C) plots actual (y-axis) vs. predicted (x-axis) values for zip code level median household income, percentage of people with a bachelors degree and percentage of Asians respectively for all zip codes in our dataset. The second and third columns map actual and predicted values of these variables in Philadelphia, PA.

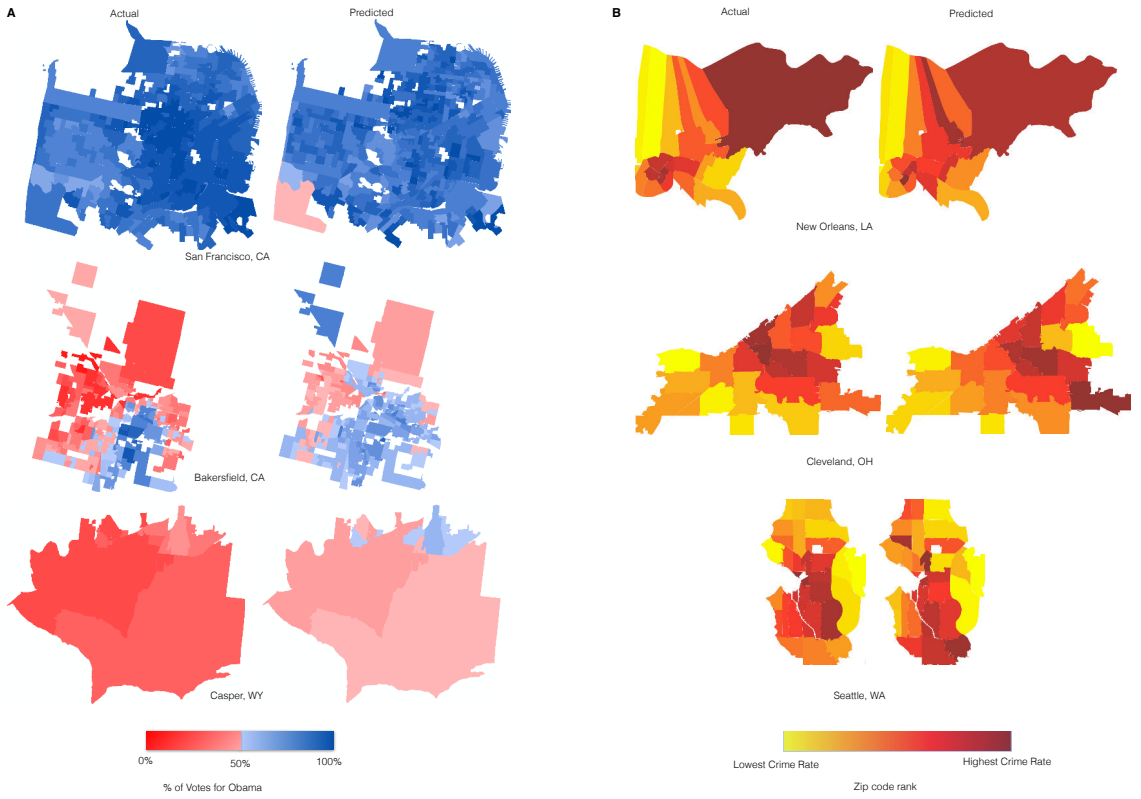



Figure 5.5: Cars detected using Google Street View images can be used to predict voting patterns and crime rates. (A) Maps actual and predicted values of the percentage of people who voted for Barack Obama in the 2008 election for San Francisco, CA, Bakersfield, CA and Casper, WY chosen as examples of mostly democrat, evenly divided and mostly republican cities respectively. Precincts with less than 50% votes for Obama are shown in red and those with greater than 50% votes for Obama are visualized as blue. (B) Shows actual and predicted crime rates for crimes committed against people in New Orleans, LA, Cleveland, OH and Seattle, WA chosen as examples of cities with high, medium and low crime rates respectively. The maps rank zip codes in each city according to their crime rates.

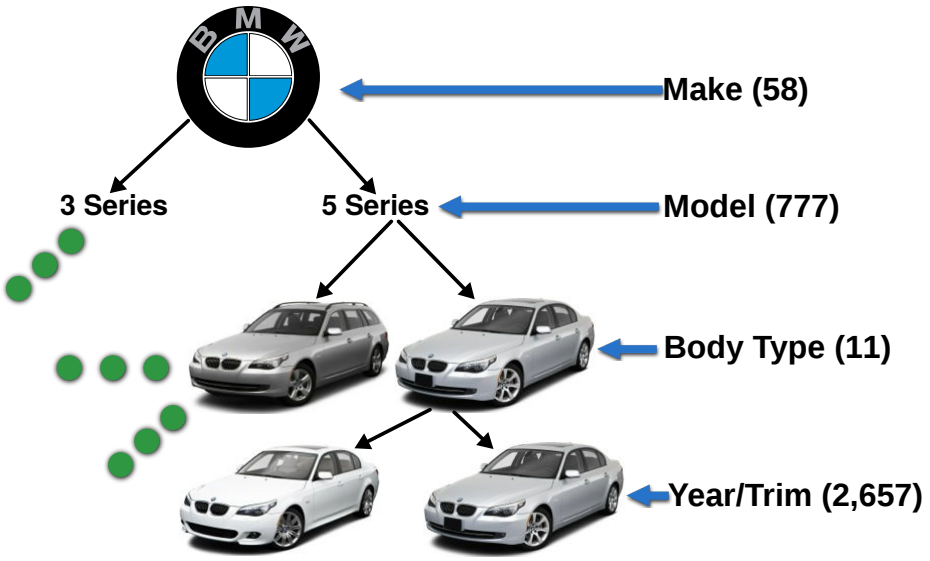



Figure 5.6: The structure of the hierarchy of car categories in our dataset. Numbers in parentheses indicate the number of unique elements at each level of the hierarchy, *i.e.* there are 58 different makes in our dataset, 777 models, 11 body types, and finally 2,657 total classes at the level of year and trim.

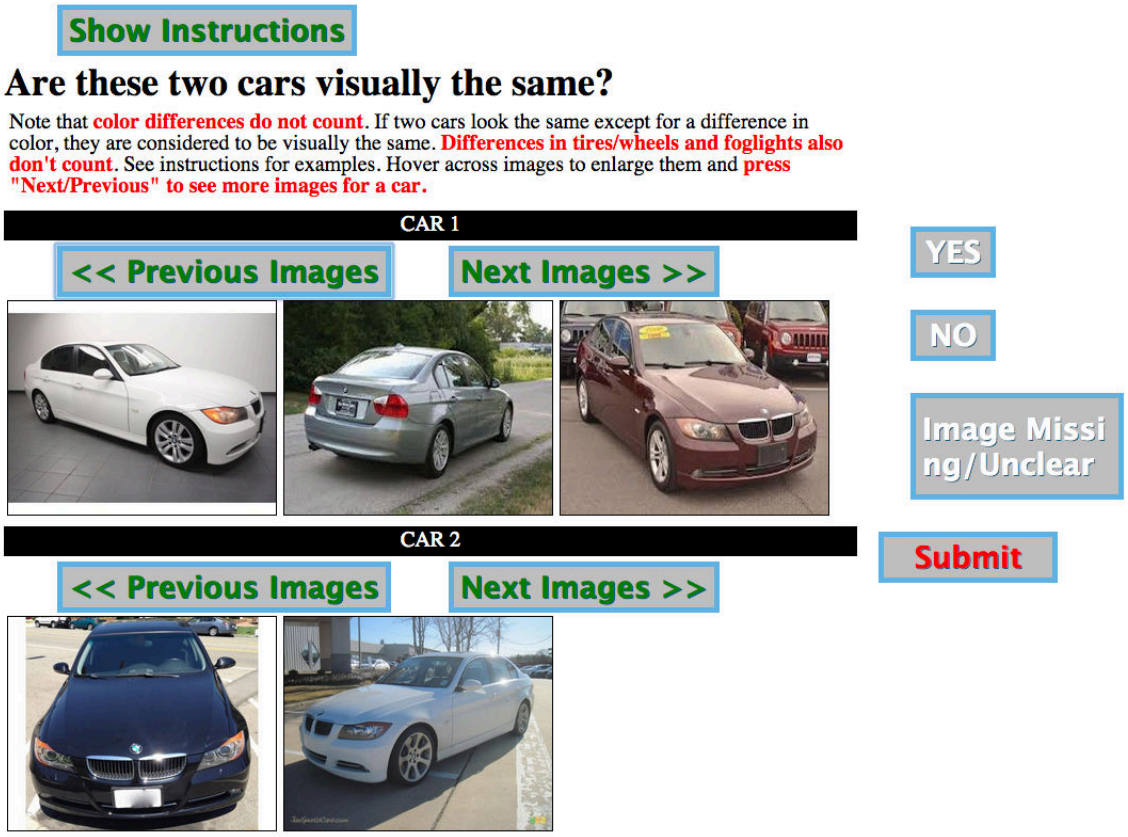



Figure 5.7: The Amazon Mechanical Turk (AMT) user interface for grouping visually indistinguishable pairs of classes. The user is asked whether or not the two cars are visually distinct with an option to view more detailed instructions.

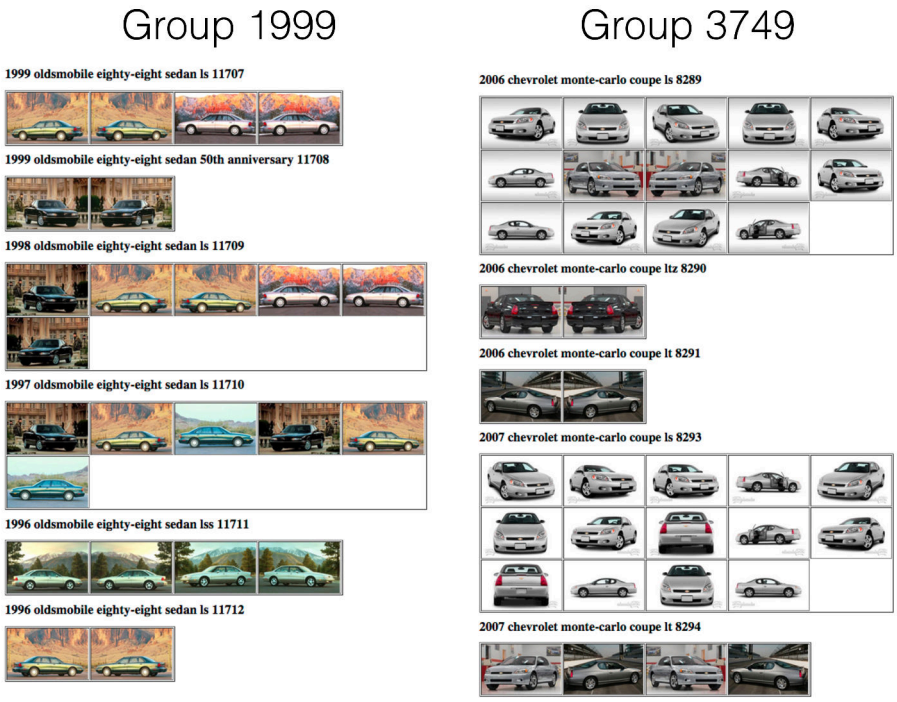



Figure 5.8: Two examples of classes and the different types of visually indistinguishable cars in each class. Each column is a unique class. The first column shows cars assembled into group 1999 whereas the second column shows those in group 3749.


Figure 5.9: Left, Middle: Examples of product shot images unsuitable for our dataset, as they are either extremely close up (left) or are of the interior of the car (middle). In order to be suitable for recognition, an image must be of the exterior of the car and the car must be entirely visible (right).


Figure 5.10: Screenshot of the user interface for labeling images containing a car viewed from the exterior, deployed on Amazon Mechanical Turk. Below the instructions are a set of images, and the user is tasked with clicking on the images containing a single prominent vehicle, viewed from the outside. Images the user clicks are moved to the panel on the right side of the screen, and clicks can be undone by clicking on the image in the right panel.


Figure 5.11: An example of the unwarping that needs to be done on images retrieved from Street View. Left: an image from Street View as initially scraped. The image appears warped (*e.g.* straight lines in the real world are not straight in the image) due to the equirectangular projection used to store spherical panoramas. Right: the result of undoing this projection, which we do before using the images any further.

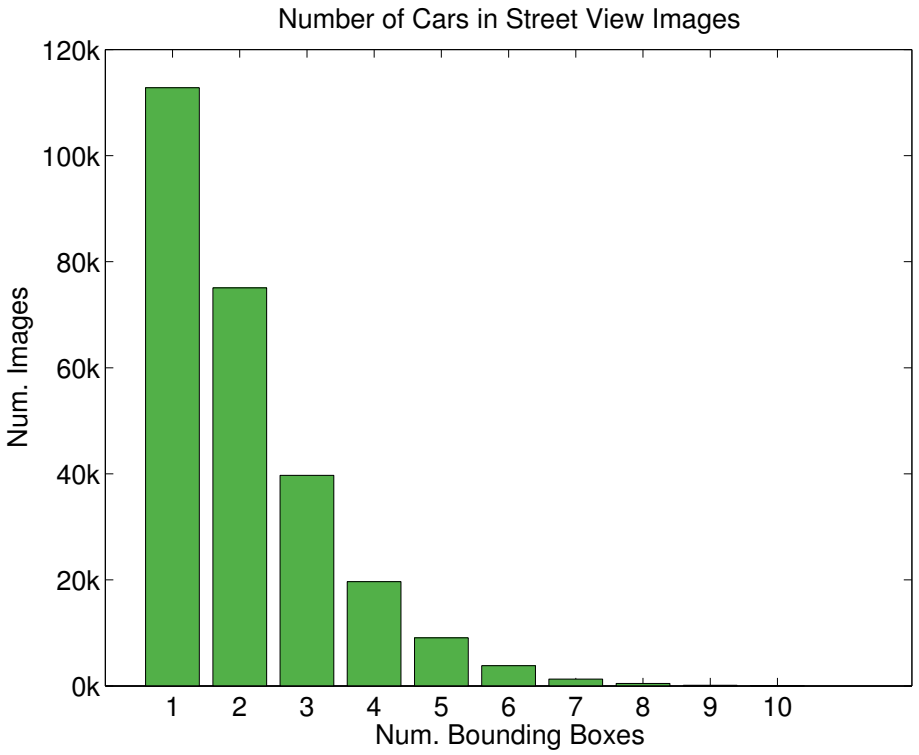



Figure 5.12: Histogram of the number of cars annotated in the Street View images, represented by the number of annotated bounding boxes in each image. Images included in these numbers are those images annotated as containing more than zero and less than 11 cars.

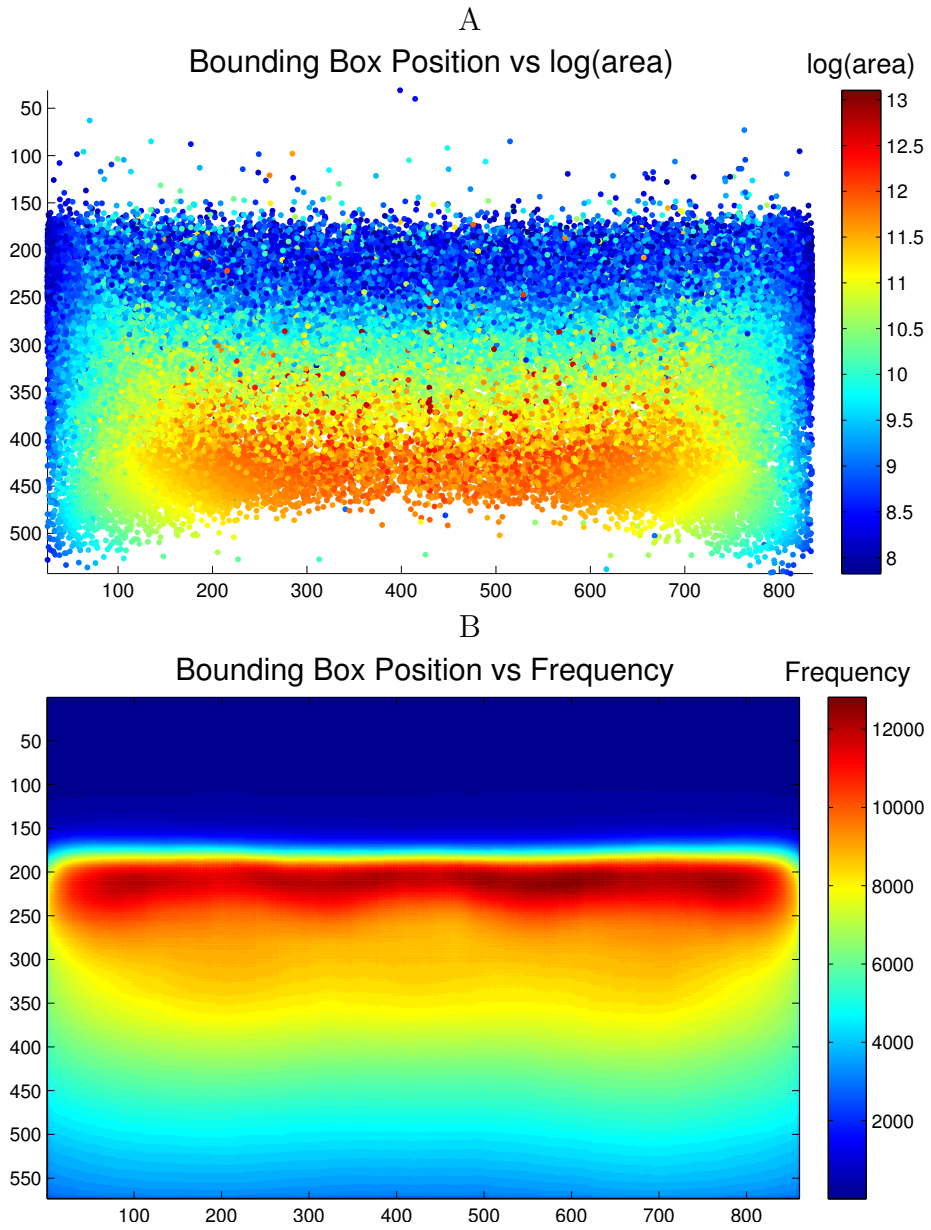



Figure 5.13: (A) Bounding box position vs log(area). Each point corresponds to a single bounding box in our training set of Street View images, and the color corresponds to the log of the number of pixels in the bounding box. (B) Bounding box position vs frequency. The color of each pixel indicates the number of bounding boxes in the training set which overlap that pixel.

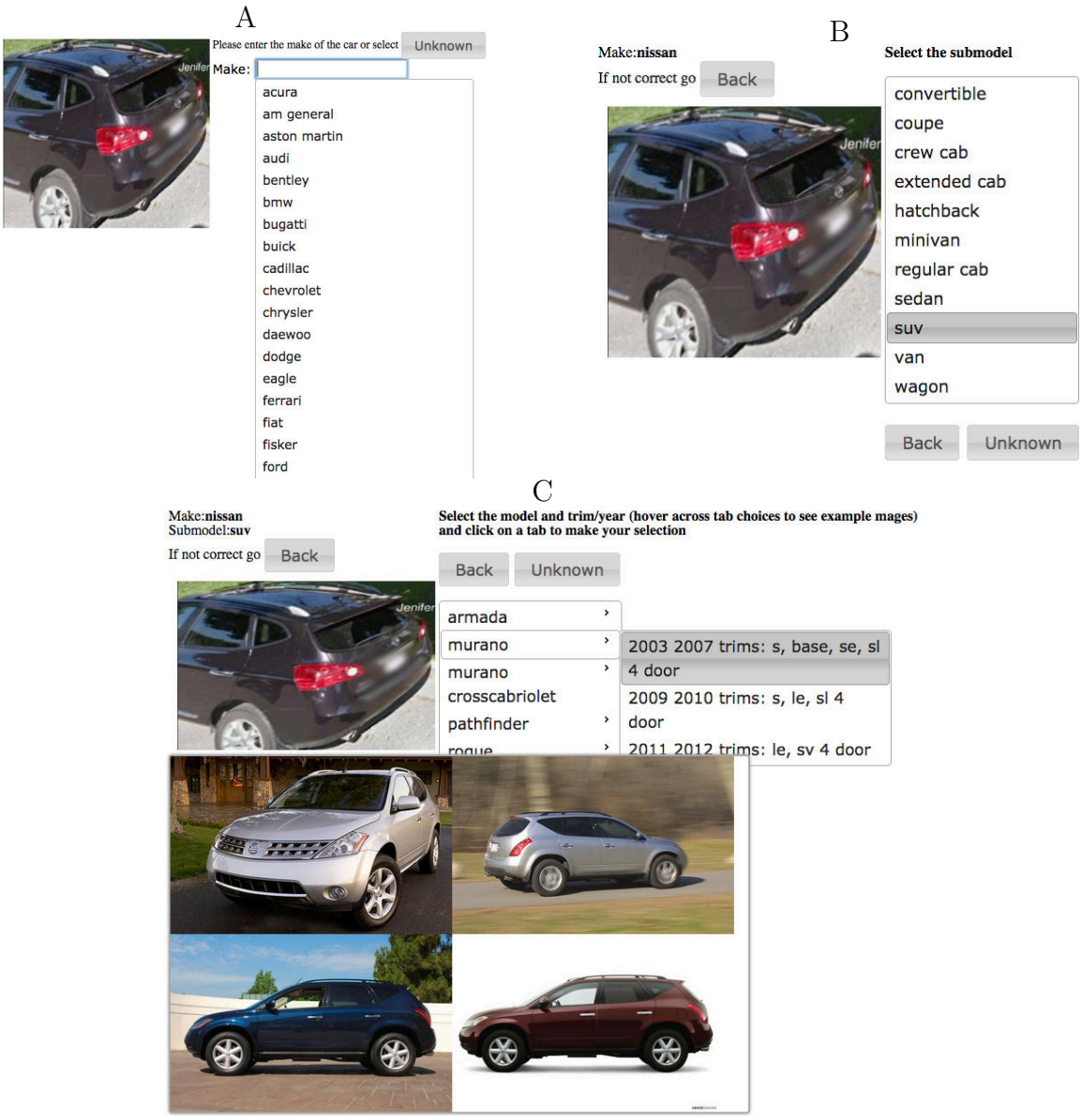



Figure 5.14: Screenshots of the user interface for hierarchically annotating Street View images with car categories. (A) The expert is first asked to identify the make. (B) The next step in the task is to identify the body type of the car which (called "submodel" in the task). (C) Once the body type is identified we provide a list of classes for the selected make and body type. Example images of each class are also shown to aid the user in identification.

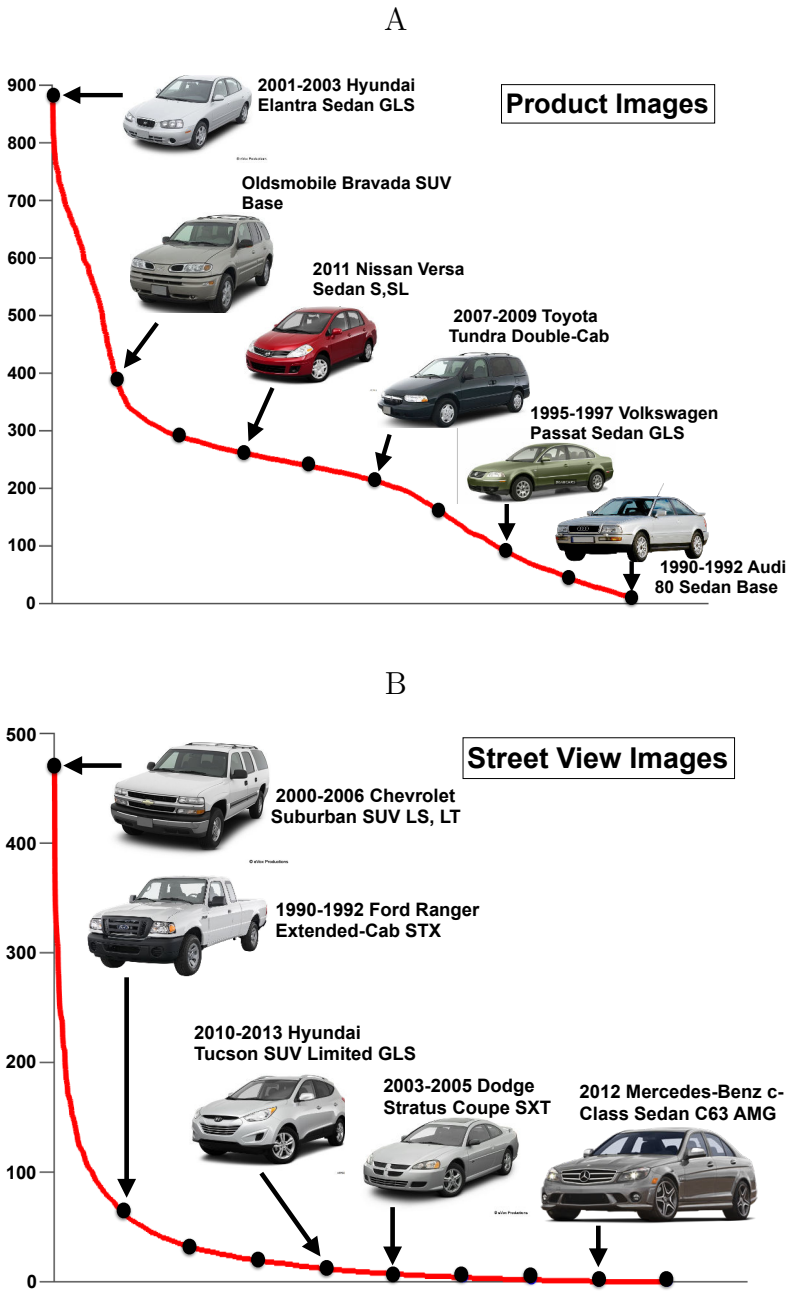



Figure 5.15: (A) Distribution of the number of images per class in our product show images. (B) Distribution of the number of images per class in Street View images.

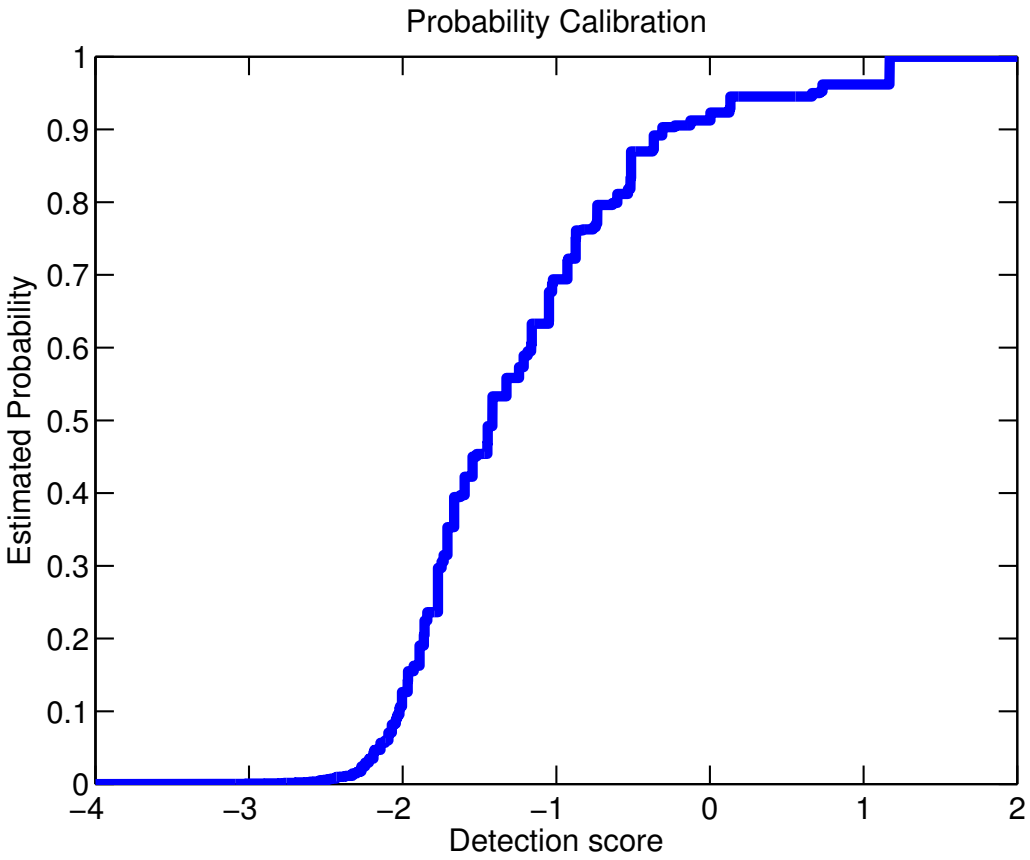



Figure 5.16: The transformation from detection scores to the probability of the detection being correct, learned with isotonic regression on the validation set.

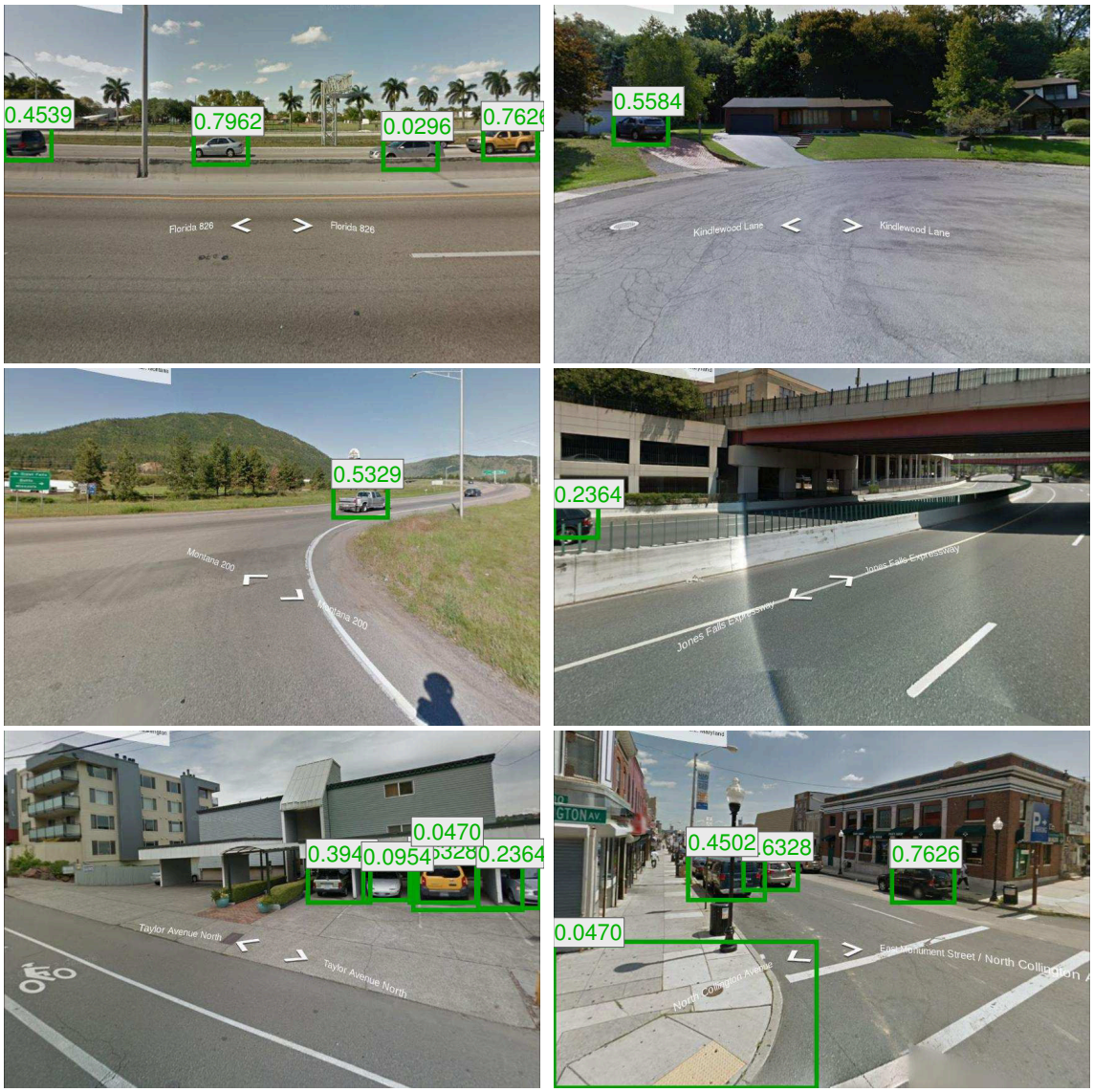



Figure 5.17: Example detections with our model on our testing set. Shown in the box around each detection is our estimated probability of the detection having intersection over union greater than 0.5, *i.e.* counted as correct during detection evaluation.

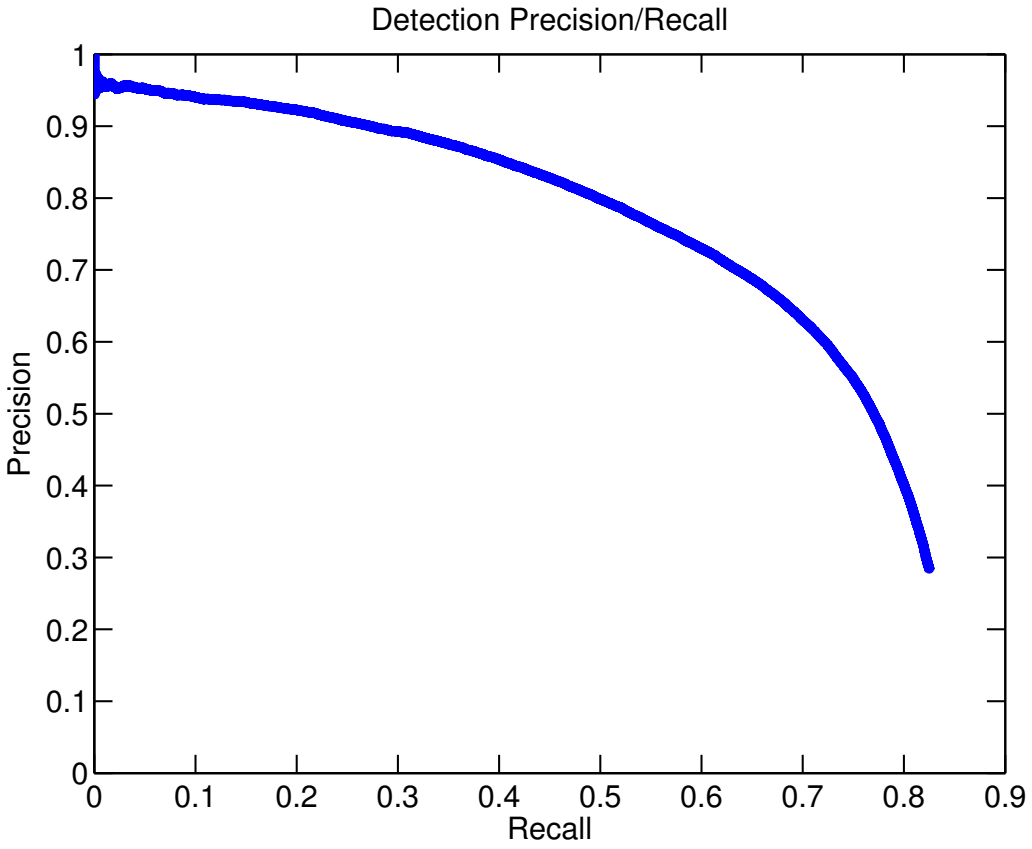



Figure 5.18: Precision/recall curve for our final detection model on the test set.

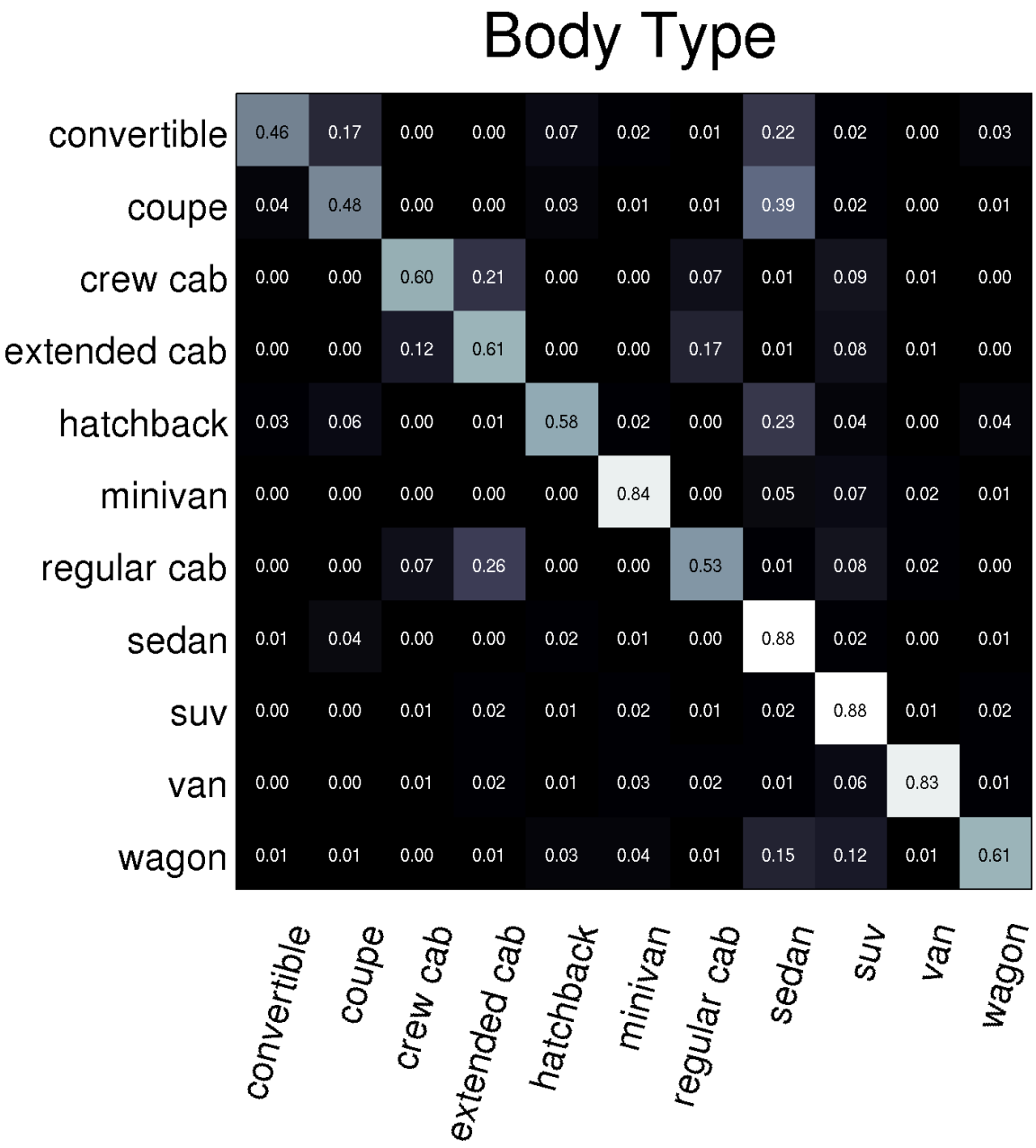



Figure 5.19: Confusion matrix of body type predictions. The entry in row $i$ and column $j$ indicates how many times ground truth body type $i$ was classified as body type $j$.

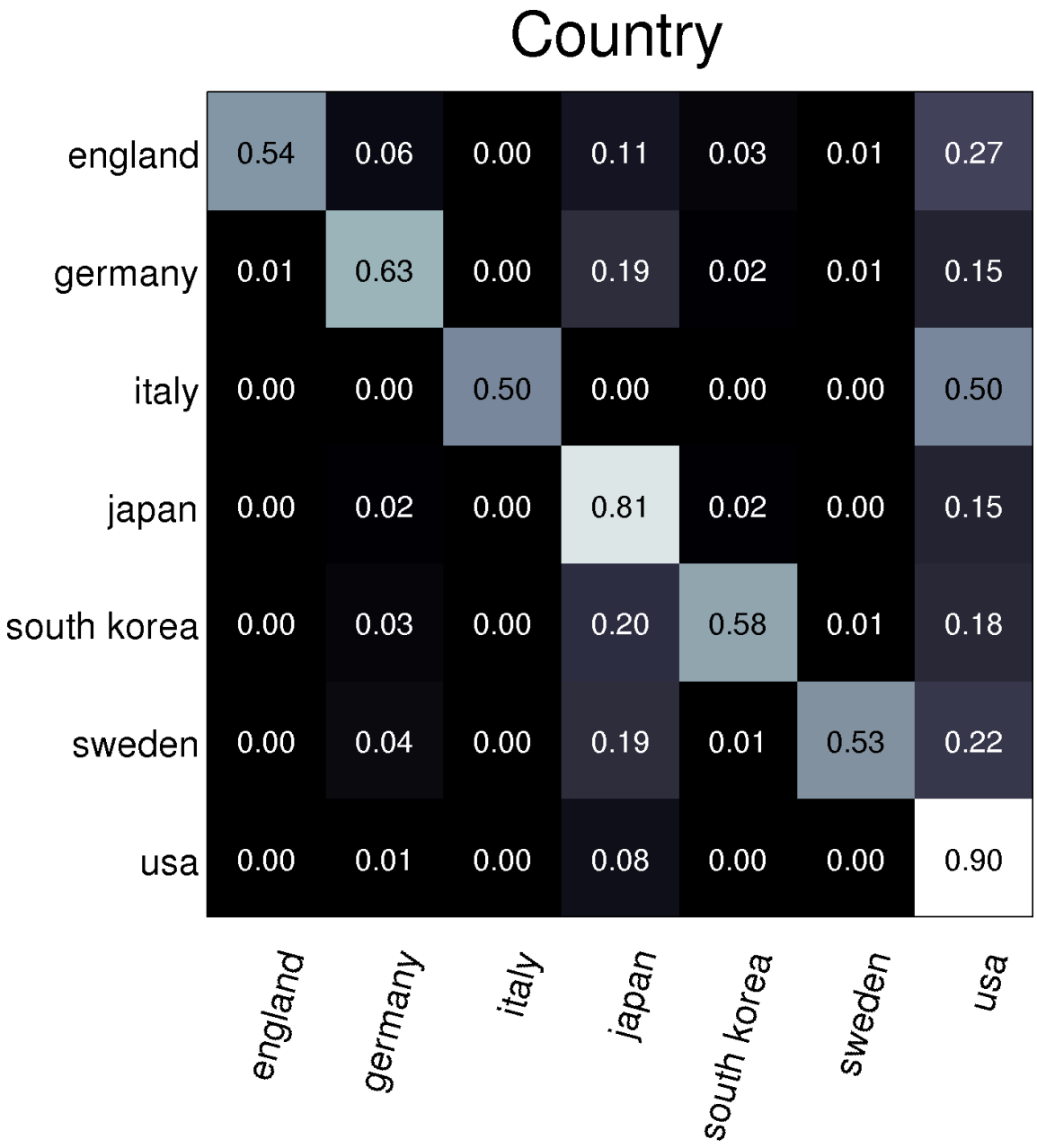



Figure 5.20: Confusion matrix of country predictions. The entry in row $i$ and column $j$ indicates how many times ground truth country $i$ was classified as country $j$.

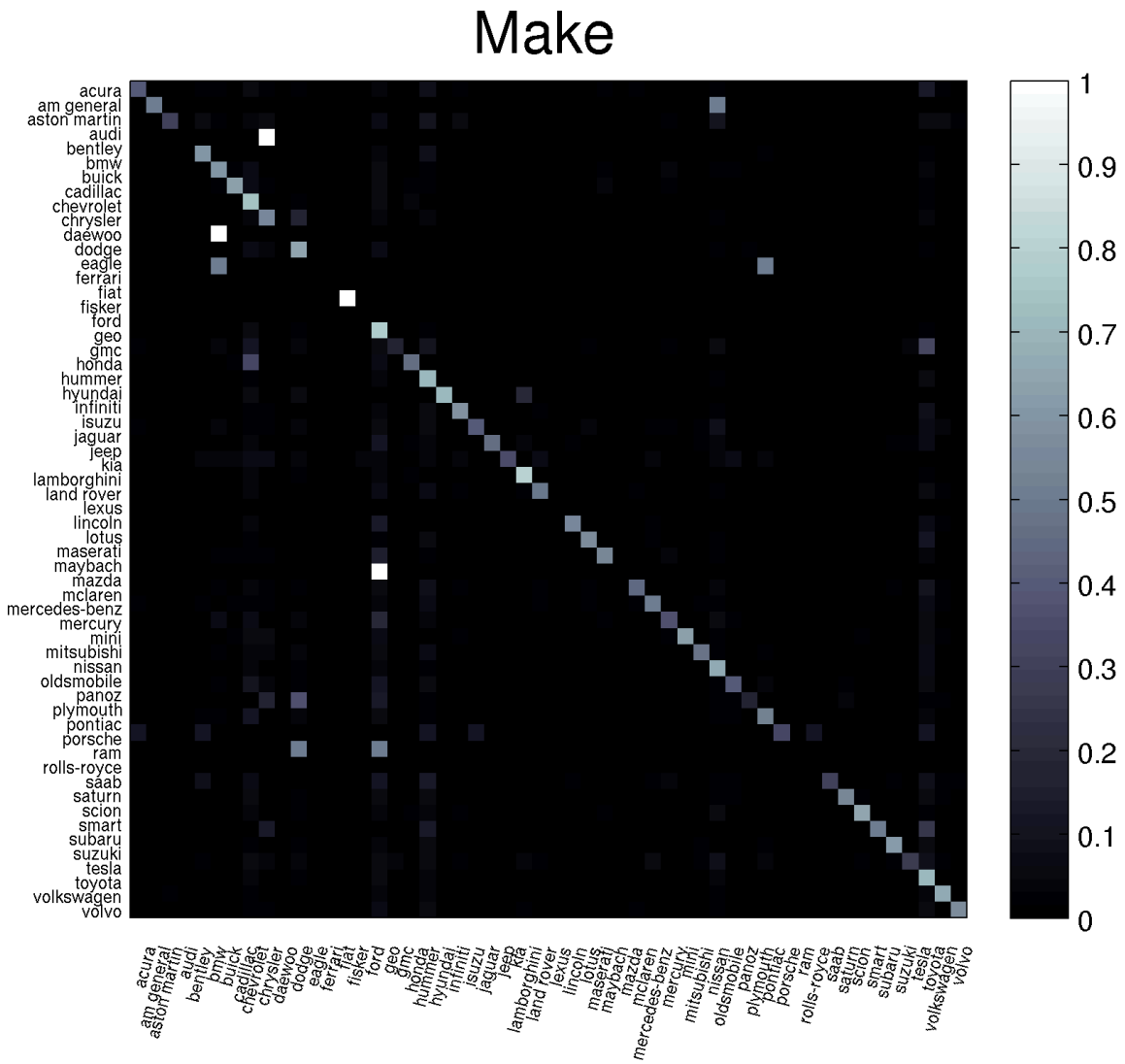



Figure 5.21: Confusion matrix of predictions at the make level. The entry in row $i$ and column $j$ indicates how many times ground truth make $i$ was classified as make $j$.

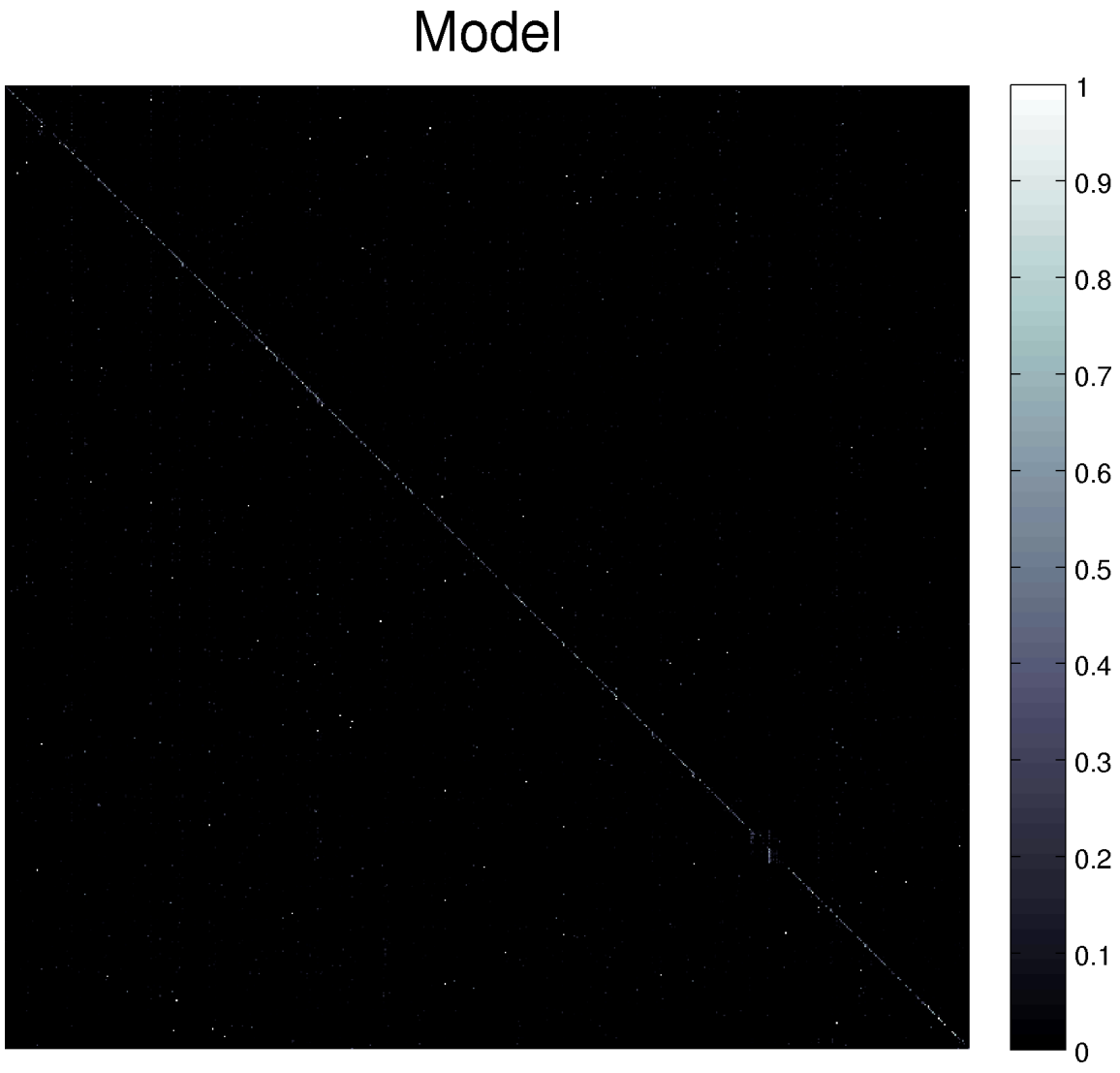



Figure 5.22: Confusion matrix of predictions at the model level. The entry in row $i$ and column $j$ indicates how many times ground truth model $i$ was classified as model $j$.

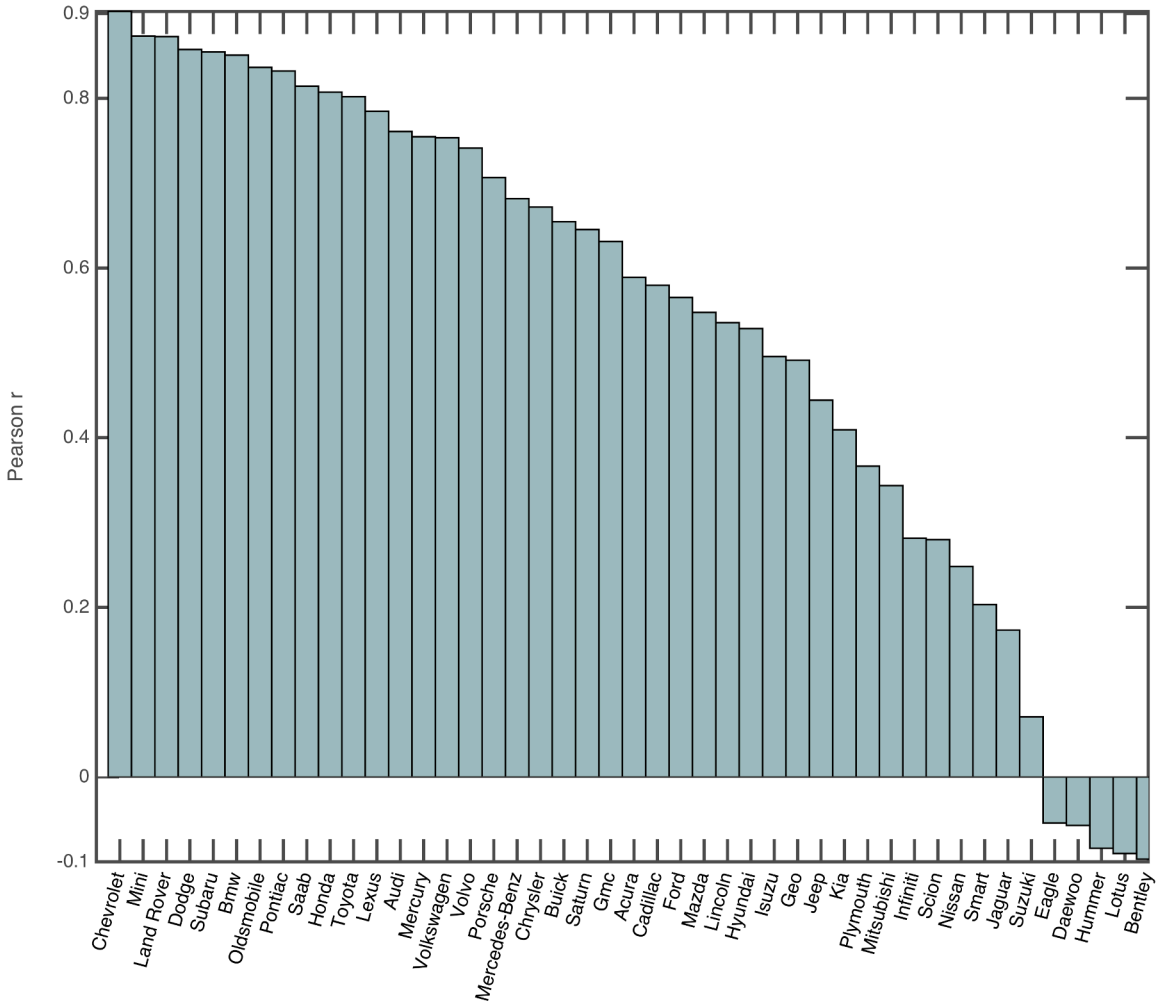



Figure 5.23: The correlation between the distribution of detected and registered car makes across zip codes in the three cities in Massachusetts, Boston, Springfield, and Worcester.

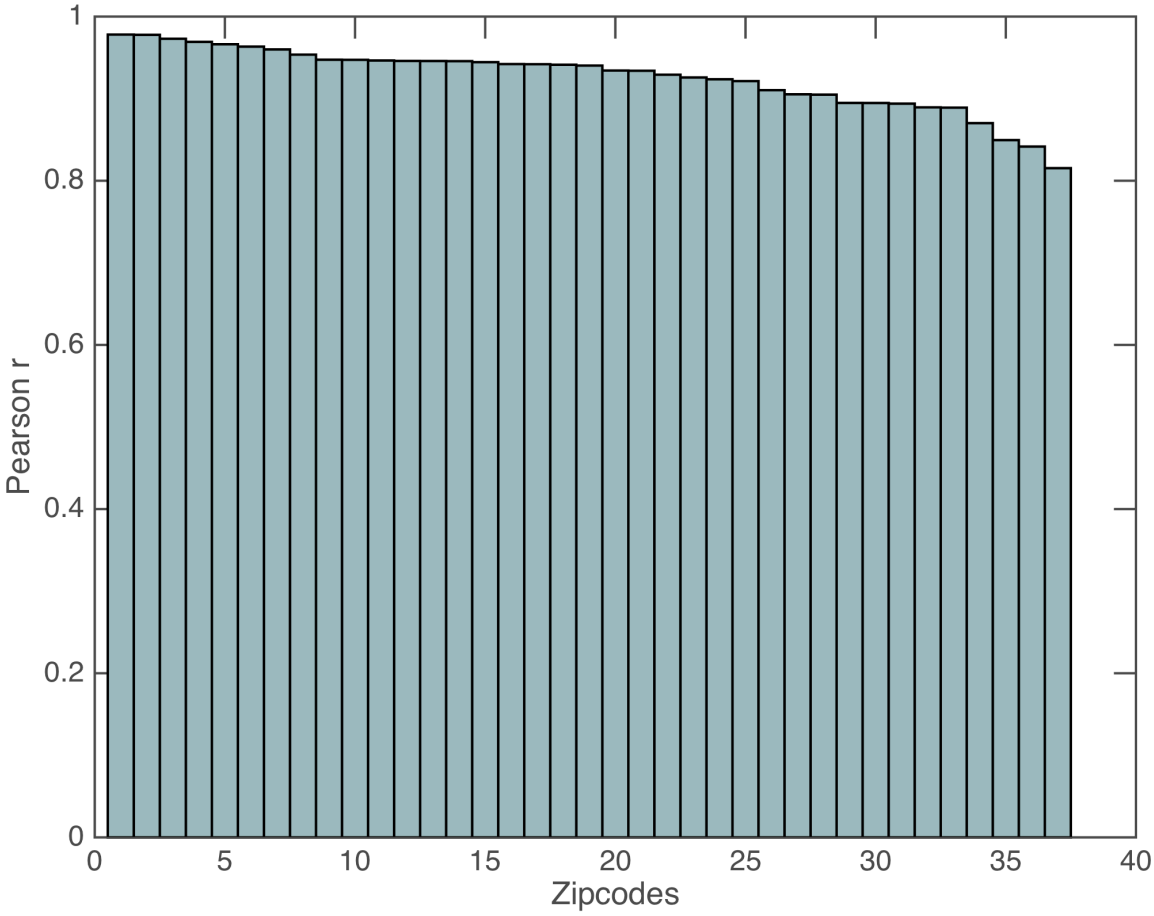



Figure 5.24: The correlation between the distribution of detected and registered car makes in each zip code for the three cities in Massachusetts (Boston, Springfield, Worcester). All zip codes have correlation greater than 0.8.

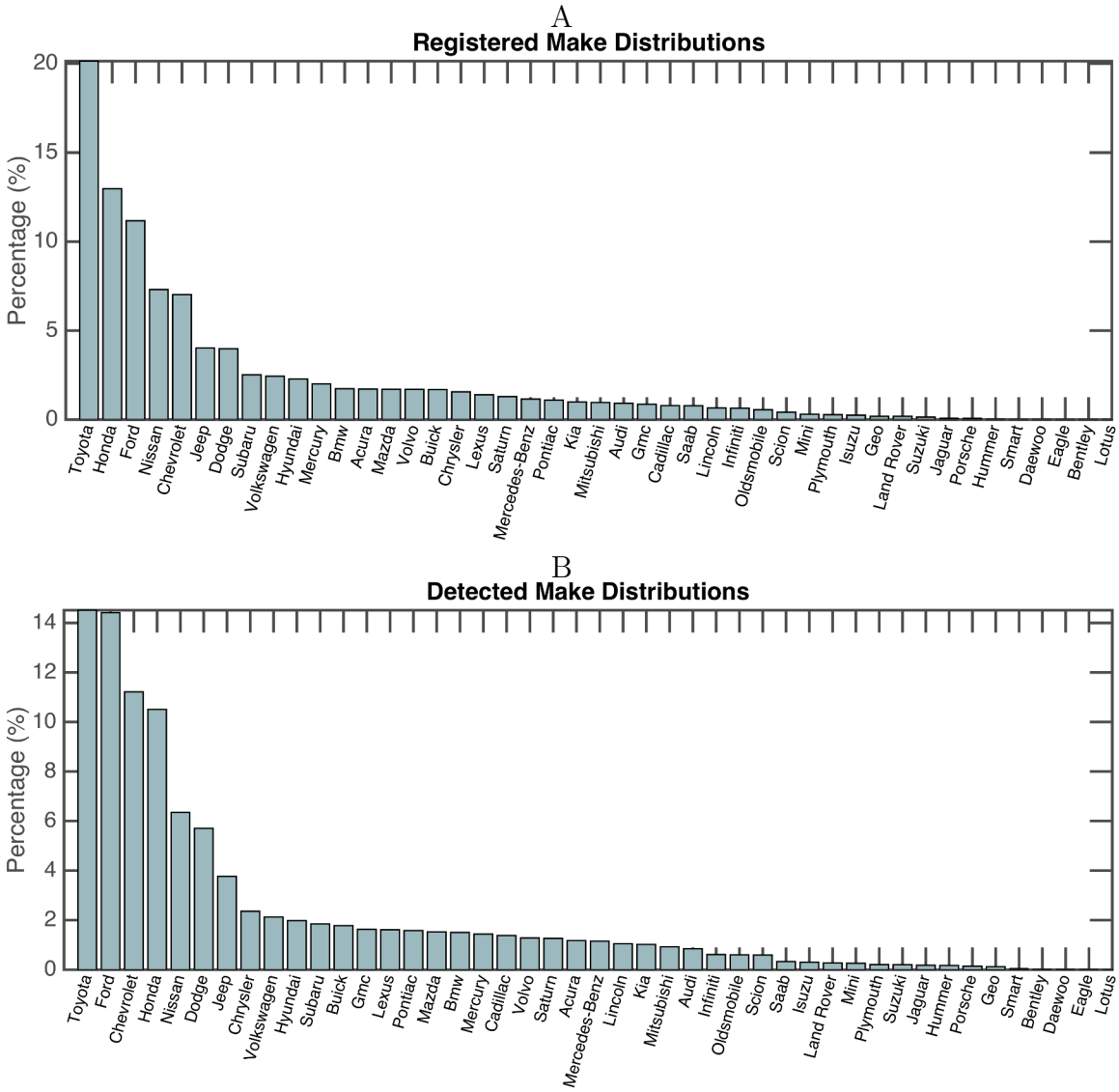
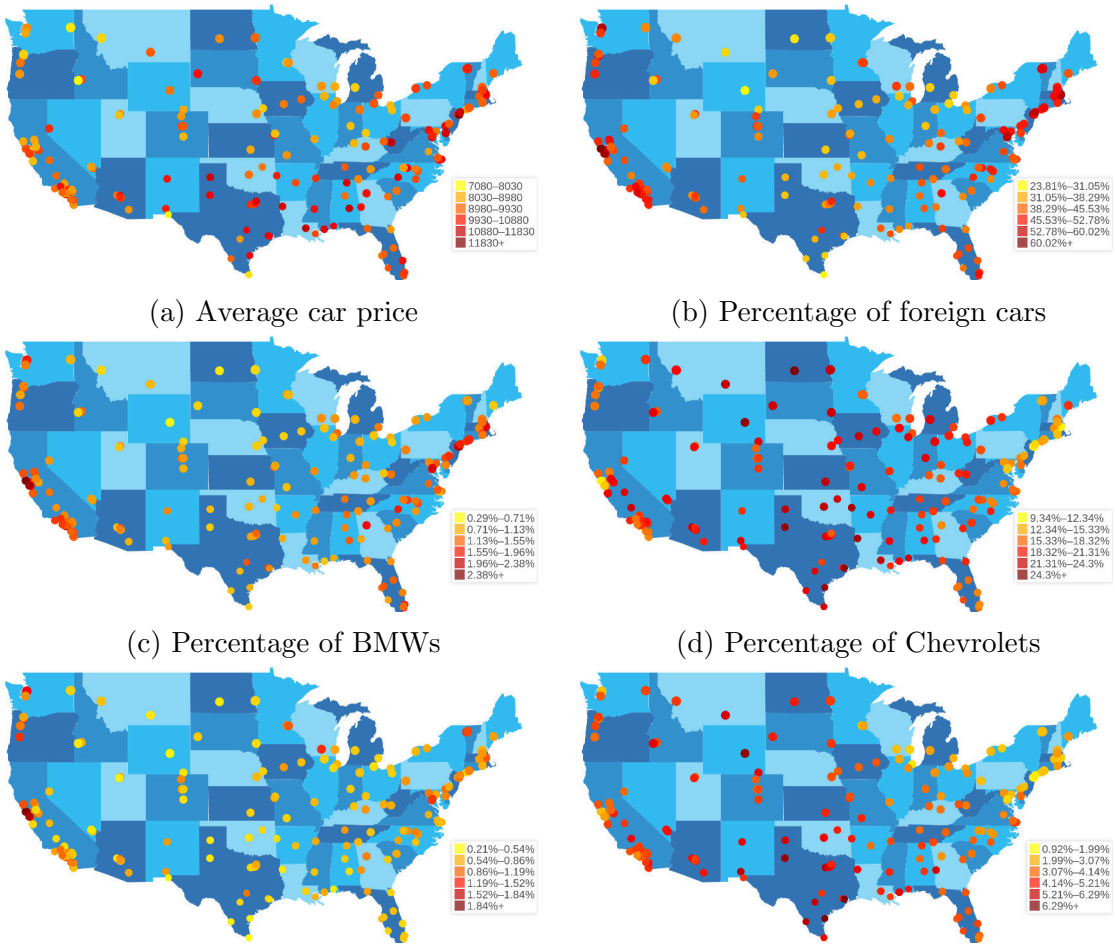



Figure 5.25: (A) The Distribution of registered makes in Boston, Springfield, and Worcester Massachusetts. (B) The distribution of detected makes in Boston, Springfield, and Worcester Massachusetts.




(a) Average car price

(b) Percentage of foreign cars

(c) Percentage of BMWs

(d) Percentage of Chevrolets

(e) Percentage of Toyota Prius

(f) Percentage of Ford F-150

Figure 5.26: Maps of a variety of car attributes as measured across the cities in our dataset. Each point corresponds to one city. Not shown: Alaska and Hawaii.

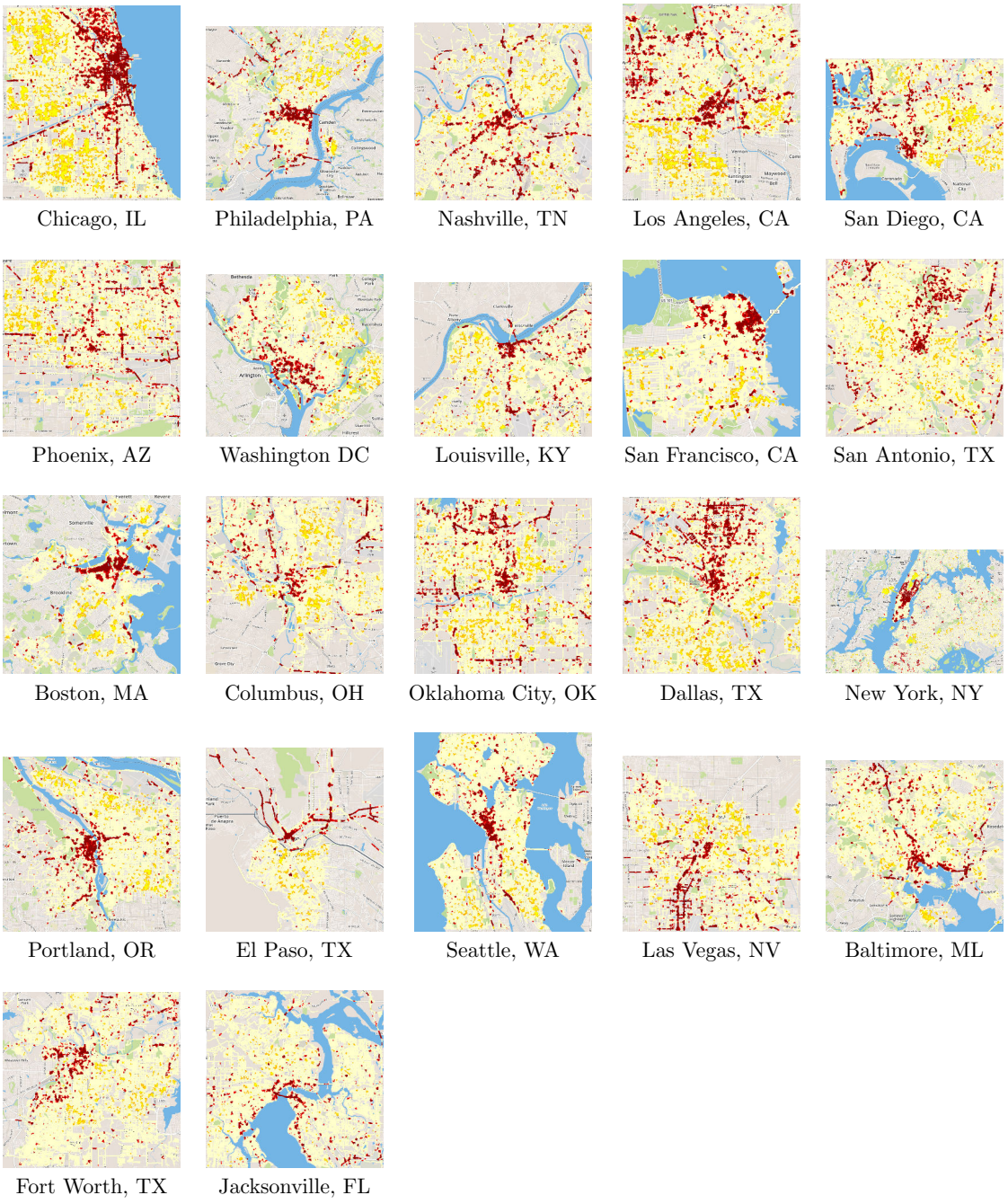



Figure 5.27: Maps of the Getis-Ord $G_i^*$ statistic for all 22 cities in the segregation analysis, ordered by decreasing value of their Moran's I statistic. Red dots signify expensive cars ($G_i^* > 2$), and yellow dots signify cheap cars ($G_i^* < 2$))

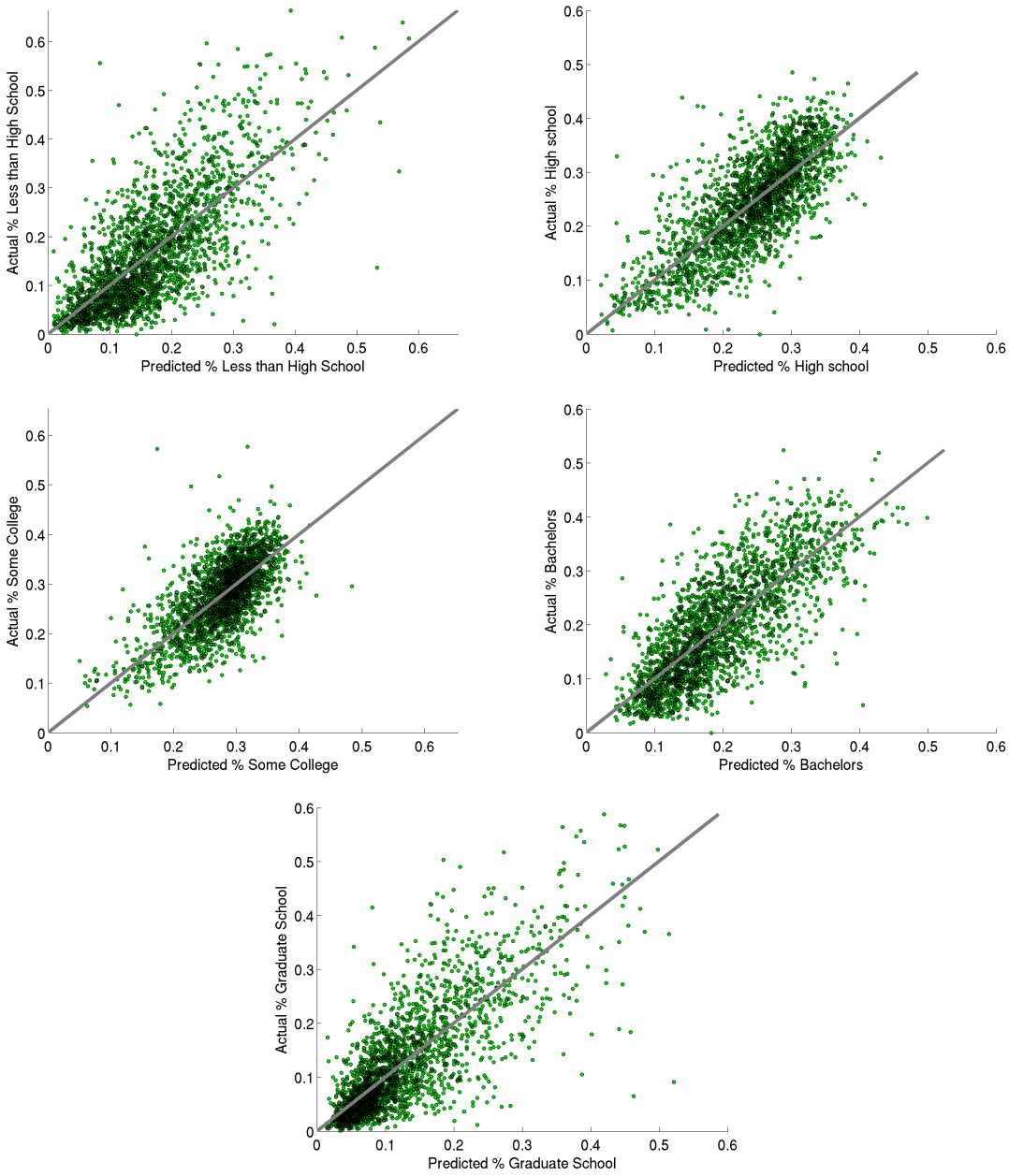



Figure 5.28: Scatter plots of predicted versus actual education levels. Also shown on each plot is the line $y = x$, which corresponds to a perfect predictor.

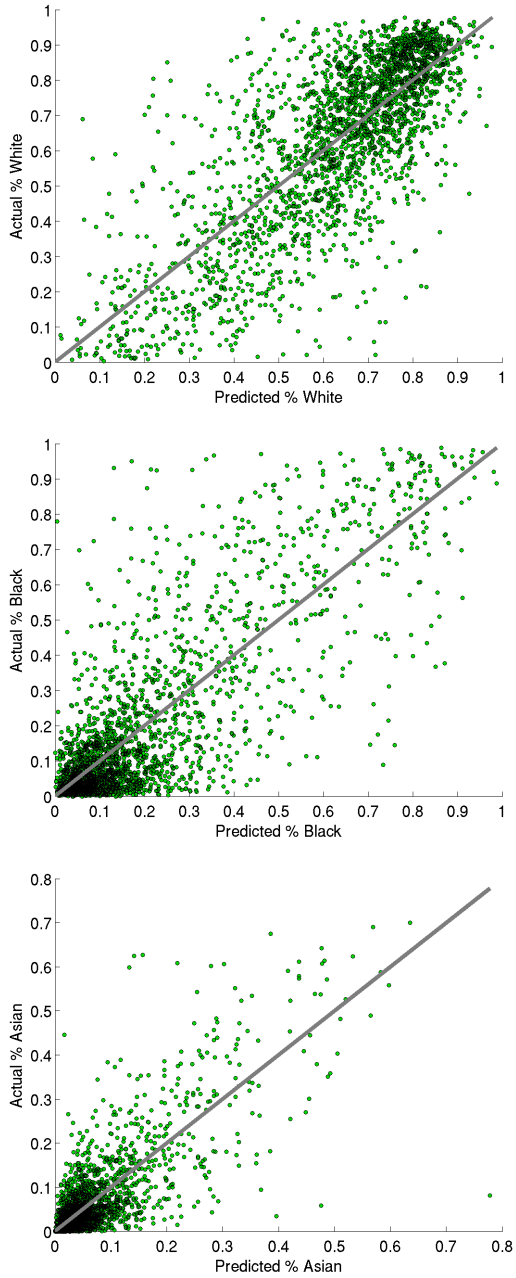



Figure 5.29: Scatter plots of predicted versus actual distributions of race. Also shown on each plot is the line $y = x$, which corresponds to a perfect predictor.

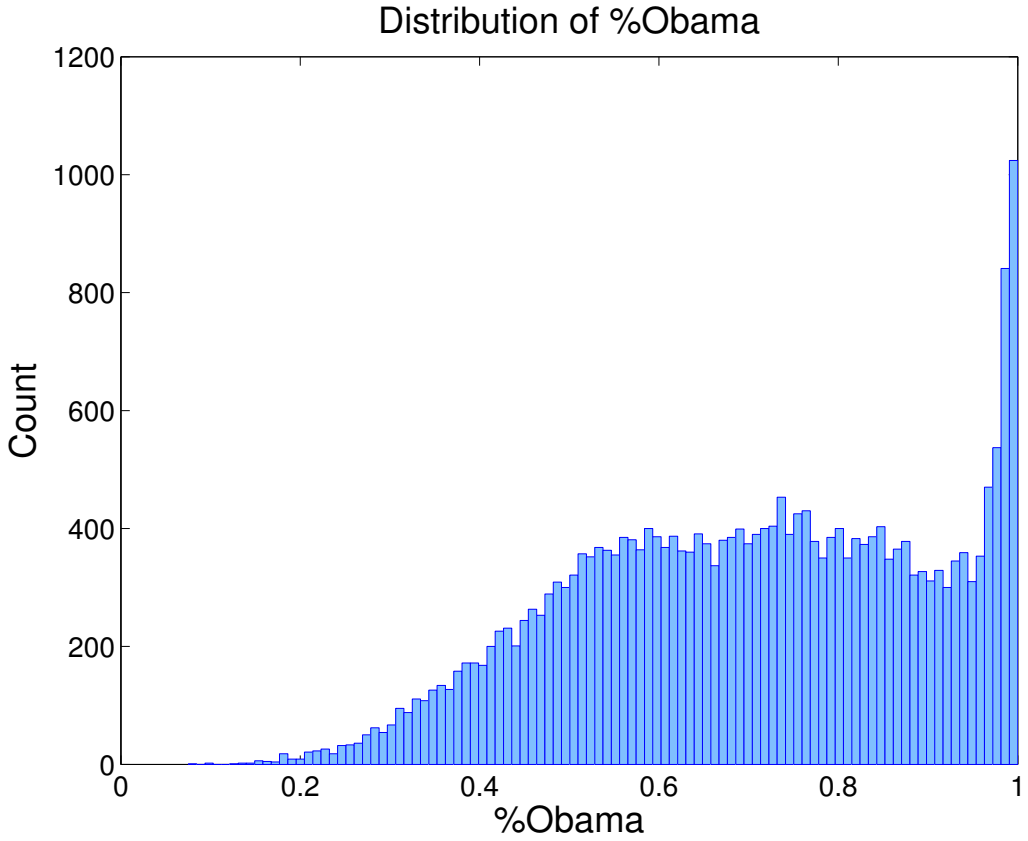



Figure 5.30: Histogram of the fraction of votes cast for Barack Obama vs. John McCain in the 2008 presidential election for the precincts in our dataset.

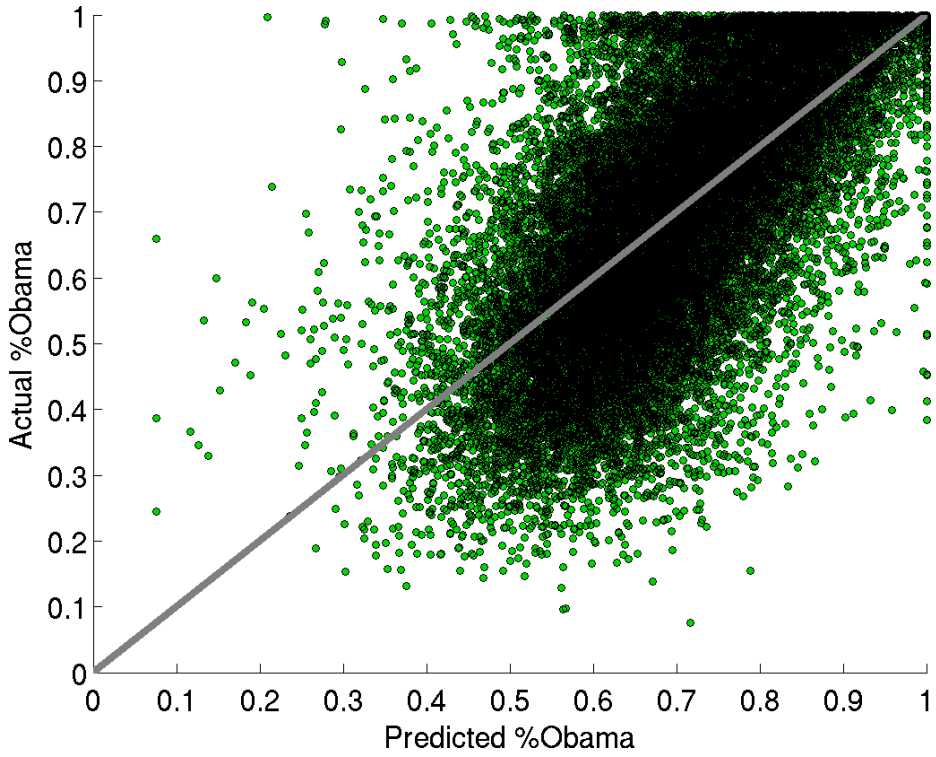



Figure 5.31: Predicted versus actual voting results. Also shown is the line $y = x$, which corresponds to a perfect predictor.

| City | # Im. | City | # Im. | City | # Im. | City | # Im. |
|---|---|---|---|---|---|---|---|
| Birmingham, AL | 484,818 | Santa Ana, CA | 90,030 | Portland, ME | 86,874 | Salem, OR | 102,174 |
| Huntsville, AL | 100,410 | Santa Clarita, CA | 83,298 | Baltimore, MD | 570,360 | Philadelphia, PA | 244,194 |
| Mobile, AL | 45,114 | Santa Rosa, CA | 243,324 | Frederick, MD | 182,388 | Pittsburgh, PA | 682,728 |
| Montgomery, AL | 45,084 | Stockton, CA | 343,662 | Boston, MA | 195,864 | Providence, RI | 130,104 |
| Anchorage, AK | 59,484 | Sunnyvale, CA | 66,318 | Springfield, MA | 116,928 | Warwick, RI | 172,092 |
| Fairbanks, AK | 42,384 | Torrance, CA | 136,260 | Worcester, MA | 197,424 | Charleston, SC | 56,604 |
| Chandler, AZ | 309,414 | Aurora, CO | 143,508 | Detroit, MI | 287,736 | Columbia, SC | 334,914 |
| Gilbert, AZ | 175,242 | Colorado Springs, CO | 492,222 | Grand Rapids, MI | 202,266 | Rapid City, SD | 30,954 |
| Glendale, AZ | 160,146 | Denver, CO | 306,990 | Minneapolis, MN | 654,270 | Sioux Falls, SD | 74,640 |
| Mesa, AZ | 283,620 | Fort Collins, CO | 307,056 | Saint Paul, MN | 164,034 | Chattanooga, TN | 284,214 |
| Peoria, AZ | 135,132 | Bridgeport, CT | 154,092 | Gulfport, MS | 14,898 | Knoxville, TN | 457,434 |
| Phoenix, AZ | 623,892 | New Haven, CT | 62,394 | Jackson, MS | 71,298 | Memphis, TN | 97,572 |
| Scottsdale, AZ | 138,120 | Dover, DE | 22,134 | Kansas City, MO | 577,830 | Nashville, TN | 554,118 |
| Tempe, AZ | 302,958 | Wilmington, DE | 80,754 | Springfield, MO | 395,502 | Amarillo, TX | 85,380 |
| Tucson, AZ | 634,986 | Washington, DC | 375,258 | St. Louis, MO | 426,942 | Arlington, TX | 509,406 |
| Fort Smith, AR | 205,512 | Cape Coral, FL | 309,102 | Billings, MT | 54,768 | Austin, TX | 211,530 |
| Little Rock, AR | 398,094 | Fort Lauderdale, FL | 279,300 | Missoula, MT | 157,254 | Brownsville, TX | 284,826 |
| Anaheim, CA | 133,098 | Hialeah, FL | 143,928 | Lincoln, NE | 444,306 | Corpus Christi, TX | 61,434 |
| Bakersfield, CA | 521,112 | Jacksonville, FL | 770,016 | Omaha, NE | 322,602 | Dallas, TX | 663,006 |
| Chula Vista, CA | 189,204 | Miami, FL | 310,692 | Henderson, NV | 259,416 | El Paso, TX | 205,500 |
| Corona, CA | 238,932 | Orlando, FL | 582,018 | Las Vegas, NV | 521,172 | Fort Worth, TX | 677,214 |
| Elk Grove, CA | 306,600 | Pembroke Pines, FL | 71,274 | North Las Vegas, NV | 197,394 | Garland, TX | 226,140 |
| Escondido, CA | 206,550 | Port St. Lucie, FL | 62,292 | Reno, NV | 104,328 | Grand Prairie, TX | 210,198 |
| Fontana, CA | 167,604 | Saint Petersburg, FL | 83,442 | Manchester, NH | 131,682 | Houston, TX | 337,830 |
| Fremont, CA | 232,608 | Tallahassee, FL | 419,220 | Nashua, NH | 139,890 | Irving, TX | 179,382 |
| Fresno, CA | 135,210 | Tampa, FL | 610,770 | Jersey City, NJ | 78,036 | Laredo, TX | 259,878 |
| Garden Grove, CA | 77,706 | Atlanta, GA | 315,336 | Newark, NJ | 129,948 | Lubbock, TX | 500,760 |
| Glendale, CA | 77,316 | Augusta, GA | 239,994 | Albuquerque, NM | 73,746 | Pasadena, TX | 29,700 |
| Hayward, CA | 207,744 | Columbus, GA | 54,246 | Las Cruces, NM | 82,098 | Plano, TX | 330,186 |
| Huntington Beach, CA | 101,574 | Hilo, HI | 14,406 | Buffalo, NY | 376,806 | San Antonio, TX | 1,034,358 |
| Irvine, CA | 183,474 | Honolulu, HI | 209,010 | New York, NY | 508,860 | Salt Lake City, UT | 272,190 |
| Lancaster, CA | 110,550 | Boise, ID | 42,438 | Rochester, NY | 391,458 | West Valley City, UT | 69,432 |
| Long Beach, CA | 265,806 | Nampa, ID | 231,318 | Yonkers, NY | 27,618 | Burlington, VT | 31,998 |
| Los Angeles, CA | 554,106 | Aurora, IL | 203,256 | Charlotte, NC | 111,510 | Essex, VT | 16,056 |
| Modesto, CA | 32,406 | Chicago, IL | 791,298 | Durham, NC | 359,592 | Alexandria, VA | 69,924 |
| Moreno Valley, CA | 180,516 | Joliet, IL | 118,116 | Fayetteville, NC | 292,296 | Chesapeake, VA | 38,568 |
| Oakland, CA | 326,208 | Rockford, IL | 372,156 | Greensboro, NC | 80,730 | Newport News, VA | 17,862 |
| Oceanside, CA | 129,384 | Fort Wayne, IN | 99,672 | Raleigh, NC | 409,776 | Norfolk, VA | 56,688 |
| Ontario, CA | 142,230 | Indianapolis, IN | 468,780 | Winston-Salem, NC | 457,314 | Richmond, VA | 504,138 |
| Oxnard, CA | 154,074 | Cedar Rapids, IA | 257,178 | Bismarck, ND | 156,912 | Virginia Beach, VA | 40,698 |
| Palmdale, CA | 164,064 | Des Moines, IA | 123,678 | Fargo, ND | 202,422 | Seattle, WA | 529,392 |
| Pomona, CA | 153,798 | Kansas City, KS | 577,830 | Akron, OH | 404,376 | Spokane, WA | 381,684 |
| Rancho Cucamonga, CA | 88,734 | Overland Park, KS | 9,252 | Cincinnati, OH | 511,842 | Tacoma, WA | 331,338 |
| Riverside, CA | 446,412 | Wichita, KS | 569,658 | Cleveland, OH | 416,142 | Vancouver, WA | 292,560 |
| Sacramento, CA | 525,756 | Lexington, KY | 345,516 | Columbus, OH | 568,776 | Charleston, WV | 38,628 |
| Salinas, CA | 175,530 | Louisville, KY | 419,544 | Toledo, OH | 51,444 | Huntington, WV | 42,144 |
| San Bernardino, CA | 124,002 | Baton Rouge, LA | 65,592 | Oklahoma City, OK | 687,234 | Madison, WI | 218,580 |
| San Diego, CA | 472,872 | New Orleans, LA | 456,042 | Tulsa, OK | 541,458 | Milwaukee, WI | 446,172 |
| San Francisco, CA | 215,298 | Shreveport, LA | 100,662 | Eugene, OR | 108,582 | Casper, WY | 43,542 |
| San Jose, CA | 274,848 | Lewiston, ME | 50,562 | Portland, OR | 548,334 | Cheyenne, WY | 211,668 |

Table 5.1: List of cities we collected Street View images for and the number of Street View images we collected for each city.

| **Attribute** | **Training** | **Validation** | **Test** |
|---|---|---|---|
| Street View Images | 199,666 | 39,933 | 159,732 |
| Product Shot Images | 313,099 | - | - |
| Total Images | 512,765 | 39,933 | 159,732 |
| Street View Bounding Boxes | 272,142 | 54,691 | 216,808 |
| Product Shot Bounding Boxes | 313,099 | - | - |
| Total Bounding Boxes | 585,241 | 54,691 | 216,808 |
| Street View Category Labels | 34,753 | 6,921 | 27,888 |
| Product Shot Category Labels | 313,099 | - | - |
| Total Category Labels | 347,852 | 6,921 | 27,888 |

Table 5.2: Dataset statistics for our training, validation, and test splits, separated into Street View and product shot images.

| Comp. | Parts | AP | Time |
|---|---|---|---|
| 1 | 0 | 52.3 | 2.27 |
| 1 | 4 | 63.2 | 3.48 |
| 1 | 8 | 64.2 | 4.84 |
| 3 | 0 | 62.9 | 6.48 |
| 3 | 4 | 66.7 | 12.20 |
| 3 | 8 | 68.4 | 16.47 |
| 5 | 0 | 64.8 | 10.25 |
| 5 | 4 | 67.3 | 16.33 |
| 5 | 8 | 68.7 | 22.07 |
| 6 | 0 | 65.2 | 10.48 |
| 8 | 0 | 66.0 | 11.17 |

Table 5.3: Average Precision (AP) on the Street View validation set for various DPM configurations. Time is measured in seconds per image. Comp. is the number of DPM components, and Parts indicates the number of parts in the model.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | 0.4435 | 1.16e-117 | Make: Fiat | 0.0747 | 0.000226 |
| Cars/Image | 0.0235 | 0.247 | Make: Fisker | 0.0751 | 0.000213 |
| MPG Highway | 0.1642 | 3.72e-16 | Make: Ford | -0.2697 | 9.17e-42 |
| MPG City | 0.2565 | 8.08e-38 | Make: Geo | -0.2051 | 1.73e-24 |
| Hybrid | 0.1169 | 7.58e-09 | Make: GMC | -0.1627 | 7.08e-16 |
| Electric | 0.1589 | 3.35e-15 | Make: Honda | 0.4234 | 2.67e-106 |
| Foreign | 0.4672 | 5.26e-132 | Make: Hummer | 0.0799 | 7.99e-05 |
| Country: England | 0.3126 | 3.15e-56 | Make: Hyundai | 0.1201 | 2.91e-09 |
| Country: Germany | 0.4335 | 6.3e-112 | Make: Infiniti | 0.2649 | 2.58e-40 |
| Country: Italy | 0.2167 | 3.18e-27 | Make: Isuzu | -0.1252 | 5.91e-10 |
| Country: Japan | 0.4471 | 1.01e-119 | Make: Jaguar | -0.0397 | 0.0504 |
| Country: South Korea | 0.0834 | 3.83e-05 | Make: Jeep | 0.0153 | 0.452 |
| Country: Sweden | 0.2553 | 1.8e-37 | Make: Kia | -0.0239 | 0.238 |
| Country: USA | -0.4672 | 5.26e-132 | Make: Lamborghini | 0.1999 | 2.56e-23 |
| Body Type: Convertible | 0.1484 | 1.95e-13 | Make: Land Rover | 0.3000 | 1e-51 |
| Body Type: Coupe | -0.2211 | 2.69e-28 | Make: Lexus | 0.4432 | 1.73e-117 |
| Body Type: Crew Cab | -0.0002 | 0.993 | Make: Lincoln | -0.1652 | 2.54e-16 |
| Body Type: Extended Cab | -0.0943 | 3.19e-06 | Make: Lotus | 0.1232 | 1.11e-09 |
| Body Type: Hatchback | 0.3352 | 7.16e-65 | Make: Maserati | 0.1096 | 6.05e-08 |
| Body Type: Minivan | 0.0833 | 3.94e-05 | Make: Maybach | 0.0570 | 0.00494 |
| Body Type: Regular Cab | -0.2179 | 1.7e-27 | Make: Mazda | 0.2094 | 1.76e-25 |
| Body Type: Sedan | -0.1537 | 2.62e-14 | Make: Mclaren | 0.1002 | 7.54e-07 |
| Body Type: SUV | 0.3136 | 1.34e-56 | Make: Mercedes-Benz | 0.3873 | 8.94e-88 |
| Body Type: Van | -0.0391 | 0.0542 | Make: Mercury | -0.3367 | 1.71e-65 |
| Body Type: Wagon | 0.1776 | 1.14e-18 | Make: Mini | 0.2749 | 2.21e-43 |
| Year: 1990-1994 | -0.3230 | 4.21e-60 | Make: Mitsubishi | -0.0739 | 0.000269 |
| Year: 1995-1999 | -0.4202 | 1.5e-104 | Make: Nissan | 0.0894 | 1.01e-05 |
| Year: 2000-2004 | 0.2043 | 2.56e-24 | Make: Oldsmobile | -0.3964 | 3.12e-92 |
| Year: 2005-2009 | 0.3864 | 2.38e-87 | Make: Panoz | 0.0507 | 0.0124 |
| Year: 2010-2014 | 0.3694 | 2.01e-79 | Make: Plymouth | -0.2496 | 7.85e-36 |
| Make: Acura | 0.3528 | 3.78e-72 | Make: Pontiac | -0.3805 | 1.46e-84 |
| Make: AM General | 0.0008 | 0.969 | Make: Porsche | 0.2967 | 1.45e-50 |
| Make: Aston Martin | 0.0934 | 4e-06 | Make: Ram | -0.0513 | 0.0114 |
| Make: Audi | 0.3420 | 1.19e-67 | Make: Rolls-Royce | 0.0843 | 3.18e-05 |
| Make: Bentley | -0.0319 | 0.115 | Make: Saab | 0.2215 | 2.14e-28 |
| Make: BMW | 0.3939 | 5.53e-91 | Make: Saturn | -0.0793 | 9.11e-05 |
| Make: Buick | -0.3975 | 8.43e-93 | Make: Scion | 0.2463 | 6.47e-35 |
| Make: Cadillac | -0.3248 | 8.39e-61 | Make: Smart | 0.1464 | 4.15e-13 |
| Make: Chevrolet | -0.3553 | 3.08e-73 | Make: Subaru | 0.1727 | 1.03e-17 |
| Make: Chrysler | -0.2720 | 1.81e-42 | Make: Suzuki | -0.0679 | 0.000817 |
| Make: Daewoo | -0.0214 | 0.293 | Make: Tesla | 0.0860 | 2.19e-05 |
| Make: Dodge | -0.3807 | 1.22e-84 | Make: Toyota | 0.4239 | 1.43e-106 |
| Make: Eagle | -0.2009 | 1.55e-23 | Make: Volkswagen | 0.3014 | 3.24e-52 |
| Make: Ferrari | 0.0694 | 0.000619 | Make: Volvo | 0.2398 | 3.9e-33 |

Table 5.4: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between median household income and each car attribute, at the zip code level. $p$-values are with respect to the null hypothesis of no correlation.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | -0.3333 | 4e-64 | Make: Fiat | -0.0721 | 0.000372 |
| Cars/Image | 0.1246 | 7.06e-10 | Make: Fisker | -0.1153 | 1.19e-08 |
| MPG Highway | -0.2920 | 5.69e-49 | Make: Ford | 0.2991 | 2.1e-51 |
| MPG City | -0.3182 | 2.57e-58 | Make: Geo | 0.1532 | 3.18e-14 |
| Hybrid | -0.1331 | 4.43e-11 | Make: GMC | 0.1745 | 4.48e-18 |
| Electric | -0.1508 | 7.82e-14 | Make: Honda | -0.2530 | 8.23e-37 |
| Foreign | -0.2955 | 3.64e-50 | Make: Hummer | 0.0199 | 0.328 |
| Country: England | -0.2434 | 4.13e-34 | Make: Hyundai | -0.3468 | 1.26e-69 |
| Country: Germany | -0.3029 | 1.01e-52 | Make: Infiniti | -0.1423 | 1.8e-12 |
| Country: Italy | -0.1587 | 3.6e-15 | Make: Isuzu | 0.1501 | 1.03e-13 |
| Country: Japan | -0.2301 | 1.45e-30 | Make: Jaguar | -0.0182 | 0.37 |
| Country: South Korea | -0.3456 | 4.03e-69 | Make: Jeep | -0.2030 | 5.22e-24 |
| Country: Sweden | -0.3541 | 1.04e-72 | Make: Kia | -0.1942 | 4.4e-22 |
| Country: USA | 0.2955 | 3.64e-50 | Make: Lamborghini | -0.1403 | 3.68e-12 |
| Body Type: Convertible | -0.2246 | 3.67e-29 | Make: Land Rover | -0.1698 | 3.55e-17 |
| Body Type: Coupe | 0.0892 | 1.06e-05 | Make: Lexus | -0.2421 | 9.4e-34 |
| Body Type: Crew Cab | 0.1080 | 9.47e-08 | Make: Lincoln | 0.1483 | 2.05e-13 |
| Body Type: Extended Cab | 0.2204 | 3.98e-28 | Make: Lotus | -0.0684 | 0.000745 |
| Body Type: Hatchback | -0.3445 | 1.13e-68 | Make: Maserati | -0.0745 | 0.000239 |
| Body Type: Minivan | -0.0206 | 0.31 | Make: Maybach | -0.0537 | 0.00806 |
| Body Type: Regular Cab | 0.2732 | 7.55e-43 | Make: Mazda | -0.2595 | 1.11e-38 |
| Body Type: Sedan | -0.0173 | 0.395 | Make: Mclaren | -0.0764 | 0.000164 |
| Body Type: SUV | -0.2361 | 3.88e-32 | Make: Mercedes-Benz | -0.1895 | 4.47e-21 |
| Body Type: Van | 0.2208 | 3.32e-28 | Make: Mercury | 0.0984 | 1.16e-06 |
| Body Type: Wagon | -0.3888 | 1.55e-88 | Make: Mini | -0.2471 | 3.94e-35 |
| Year: 1990-1994 | 0.3230 | 3.87e-60 | Make: Mitsubishi | 0.1061 | 1.58e-07 |
| Year: 1995-1999 | 0.3715 | 2.12e-80 | Make: Nissan | 0.1217 | 1.78e-09 |
| Year: 2000-2004 | -0.1652 | 2.5e-16 | Make: Oldsmobile | 0.1131 | 2.24e-08 |
| Year: 2005-2009 | -0.3831 | 8.51e-86 | Make: Panoz | -0.1770 | 1.47e-18 |
| Year: 2010-2014 | -0.3296 | 1.17e-62 | Make: Plymouth | 0.1237 | 9.6e-10 |
| Make: Acura | -0.1895 | 4.45e-21 | Make: Pontiac | 0.0520 | 0.0103 |
| Make: AM General | 0.0676 | 0.00085 | Make: Porsche | -0.2387 | 8.08e-33 |
| Make: Aston Martin | -0.1157 | 1.07e-08 | Make: Ram | -0.0817 | 5.54e-05 |
| Make: Audi | -0.3176 | 4.47e-58 | Make: Rolls-Royce | -0.1050 | 2.12e-07 |
| Make: Bentley | 0.0356 | 0.0792 | Make: Saab | -0.2844 | 1.93e-46 |
| Make: BMW | -0.1956 | 2.24e-22 | Make: Saturn | -0.1775 | 1.19e-18 |
| Make: Buick | 0.0353 | 0.0822 | Make: Scion | -0.1481 | 2.14e-13 |
| Make: Cadillac | 0.1866 | 1.81e-20 | Make: Smart | -0.1571 | 6.68e-15 |
| Make: Chevrolet | 0.3183 | 2.46e-58 | Make: Subaru | -0.3597 | 3.82e-75 |
| Make: Chrysler | 0.0264 | 0.194 | Make: Suzuki | -0.0054 | 0.79 |
| Make: Daewoo | 0.0122 | 0.548 | Make: Tesla | -0.0661 | 0.00112 |
| Make: Dodge | 0.2208 | 3.24e-28 | Make: Toyota | -0.1686 | 6.02e-17 |
| Make: Eagle | 0.0612 | 0.00254 | Make: Volkswagen | -0.2975 | 7.55e-51 |
| Make: Ferrari | -0.0584 | 0.00396 | Make: Volvo | -0.3376 | 7.58e-66 |

Table 5.5: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between the percentage of residents who did not graduate high school and each car attribute, at the zip code level.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | -0.4345 | 1.91e-112 | Make: Fiat | -0.0925 | 4.9e-06 |
| Cars/Image | -0.2684 | 2.34e-41 | Make: Fisker | -0.1364 | 1.45e-11 |
| MPG Highway | -0.3294 | 1.33e-62 | Make: Ford | 0.3735 | 2.78e-81 |
| MPG City | -0.4373 | 4.66e-114 | Make: Geo | 0.1799 | 4.06e-19 |
| Hybrid | -0.1288 | 1.88e-10 | Make: GMC | 0.2760 | 1.01e-43 |
| Electric | -0.2963 | 1.92e-50 | Make: Honda | -0.5371 | 1.12e-181 |
| Foreign | -0.6048 | 2.04e-242 | Make: Hummer | -0.0502 | 0.0134 |
| Country: England | -0.4811 | 5.74e-141 | Make: Hyundai | -0.0521 | 0.0102 |
| Country: Germany | -0.6142 | 5.35e-252 | Make: Infiniti | -0.3755 | 3.16e-82 |
| Country: Italy | -0.2462 | 6.92e-35 | Make: Isuzu | 0.0364 | 0.0731 |
| Country: Japan | -0.5670 | 9.84e-207 | Make: Jaguar | -0.0201 | 0.322 |
| Country: South Korea | -0.0235 | 0.247 | Make: Jeep | -0.0247 | 0.223 |
| Country: Sweden | -0.4033 | 9.8e-96 | Make: Kia | 0.0442 | 0.0292 |
| Country: USA | 0.6048 | 2.04e-242 | Make: Lamborghini | -0.1902 | 3.14e-21 |
| Body Type: Convertible | -0.2258 | 1.84e-29 | Make: Land Rover | -0.4198 | 2.38e-104 |
| Body Type: Coupe | 0.1453 | 6.19e-13 | Make: Lexus | -0.4841 | 5.05e-143 |
| Body Type: Crew Cab | 0.1141 | 1.69e-08 | Make: Lincoln | 0.2109 | 7.74e-26 |
| Body Type: Extended Cab | 0.2023 | 7.48e-24 | Make: Lotus | -0.0922 | 5.33e-06 |
| Body Type: Hatchback | -0.5576 | 1.38e-198 | Make: Maserati | -0.1612 | 1.3e-15 |
| Body Type: Minivan | 0.0811 | 6.31e-05 | Make: Maybach | -0.0328 | 0.106 |
| Body Type: Regular Cab | 0.3076 | 2.01e-54 | Make: Mazda | -0.3406 | 4.81e-67 |
| Body Type: Sedan | 0.0382 | 0.0601 | Make: Mclaren | -0.1054 | 1.89e-07 |
| Body Type: SUV | -0.2750 | 1.97e-43 | Make: Mercedes-Benz | -0.4409 | 3.81e-116 |
| Body Type: Van | 0.0753 | 0.000204 | Make: Mercury | 0.3778 | 2.66e-83 |
| Body Type: Wagon | -0.3653 | 1.41e-77 | Make: Mini | -0.4305 | 3.09e-110 |
| Year: 1990-1994 | 0.3364 | 2.25e-65 | Make: Mitsubishi | 0.0650 | 0.00135 |
| Year: 1995-1999 | 0.4220 | 1.51e-105 | Make: Nissan | -0.1273 | 3.02e-10 |
| Year: 2000-2004 | -0.2078 | 4.08e-25 | Make: Oldsmobile | 0.4061 | 3.97e-97 |
| Year: 2005-2009 | -0.4040 | 4.68e-96 | Make: Panoz | -0.0190 | 0.349 |
| Year: 2010-2014 | -0.3541 | 1.06e-72 | Make: Plymouth | 0.3036 | 5.54e-53 |
| Make: Acura | -0.3951 | 1.3e-91 | Make: Pontiac | 0.3608 | 1.34e-75 |
| Make: AM General | 0.0806 | 6.91e-05 | Make: Porsche | -0.3736 | 2.44e-81 |
| Make: Aston Martin | -0.1640 | 4.18e-16 | Make: Ram | 0.1413 | 2.58e-12 |
| Make: Audi | -0.4816 | 2.64e-141 | Make: Rolls-Royce | -0.1119 | 3.19e-08 |
| Make: Bentley | 0.0386 | 0.0569 | Make: Saab | -0.3086 | 8.85e-55 |
| Make: BMW | -0.5318 | 1.81e-177 | Make: Saturn | 0.0705 | 0.000509 |
| Make: Buick | 0.4488 | 9.49e-121 | Make: Scion | -0.2673 | 5e-41 |
| Make: Cadillac | 0.3722 | 1.01e-80 | Make: Smart | -0.2644 | 3.66e-40 |
| Make: Chevrolet | 0.4747 | 8.44e-137 | Make: Subaru | -0.3302 | 6.37e-63 |
| Make: Chrysler | 0.3358 | 3.92e-65 | Make: Suzuki | -0.0052 | 0.8 |
| Make: Daewoo | -0.0043 | 0.832 | Make: Tesla | -0.1519 | 5.15e-14 |
| Make: Dodge | 0.5527 | 1.99e-194 | Make: Toyota | -0.5068 | 9.3e-159 |
| Make: Eagle | 0.1834 | 7.98e-20 | Make: Volkswagen | -0.5290 | 2.39e-175 |
| Make: Ferrari | -0.1523 | 4.5e-14 | Make: Volvo | -0.3879 | 4.55e-88 |

Table 5.6: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between the percentage of residents with a high school degree and each car attribute, at the zip code level.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | -0.2577 | 3.62e-38 | Make: Fiat | -0.0947 | 2.92e-06 |
| Cars/Image | -0.4257 | 1.43e-107 | Make: Fisker | -0.1472 | 3.03e-13 |
| MPG Highway | -0.3847 | 1.5e-86 | Make: Ford | 0.2781 | 2.18e-44 |
| MPG City | -0.3933 | 1.01e-90 | Make: Geo | 0.0910 | 7e-06 |
| Hybrid | -0.1792 | 5.51e-19 | Make: GMC | 0.2901 | 2.46e-48 |
| Electric | -0.2099 | 1.35e-25 | Make: Honda | -0.3382 | 4.53e-66 |
| Foreign | -0.3834 | 6.37e-86 | Make: Hummer | -0.0548 | 0.00693 |
| Country: England | -0.4480 | 2.85e-120 | Make: Hyundai | -0.1058 | 1.72e-07 |
| Country: Germany | -0.4791 | 1.16e-139 | Make: Infiniti | -0.3544 | 7.84e-73 |
| Country: Italy | -0.2939 | 1.25e-49 | Make: Isuzu | 0.0305 | 0.133 |
| Country: Japan | -0.3221 | 8.96e-60 | Make: Jaguar | -0.2449 | 1.61e-34 |
| Country: South Korea | -0.0805 | 7.07e-05 | Make: Jeep | -0.0230 | 0.257 |
| Country: Sweden | -0.2889 | 6.45e-48 | Make: Kia | 0.0052 | 0.799 |
| Country: USA | 0.3834 | 6.37e-86 | Make: Lamborghini | -0.2414 | 1.48e-33 |
| Body Type: Convertible | -0.0883 | 1.29e-05 | Make: Land Rover | -0.3614 | 7.2e-76 |
| Body Type: Coupe | 0.0791 | 9.54e-05 | Make: Lexus | -0.2781 | 2.03e-44 |
| Body Type: Crew Cab | 0.3601 | 2.67e-75 | Make: Lincoln | -0.0891 | 1.08e-05 |
| Body Type: Extended Cab | 0.4391 | 4.61e-115 | Make: Lotus | -0.1283 | 2.21e-10 |
| Body Type: Hatchback | -0.3453 | 5.36e-69 | Make: Maserati | -0.1971 | 1.06e-22 |
| Body Type: Minivan | -0.0567 | 0.00514 | Make: Maybach | -0.0698 | 0.000579 |
| Body Type: Regular Cab | 0.4194 | 3.99e-104 | Make: Mazda | -0.2630 | 1.02e-39 |
| Body Type: Sedan | -0.2780 | 2.2e-44 | Make: Mclaren | -0.1225 | 1.37e-09 |
| Body Type: SUV | -0.1038 | 2.89e-07 | Make: Mercedes-Benz | -0.3848 | 1.29e-86 |
| Body Type: Van | -0.1936 | 5.98e-22 | Make: Mercury | 0.0195 | 0.337 |
| Body Type: Wagon | -0.1893 | 4.75e-21 | Make: Mini | -0.3150 | 4.19e-57 |
| Year: 1990-1994 | 0.2961 | 2.2e-50 | Make: Mitsubishi | 0.0614 | 0.00248 |
| Year: 1995-1999 | 0.1242 | 8.16e-10 | Make: Nissan | -0.1723 | 1.17e-17 |
| Year: 2000-2004 | -0.0270 | 0.183 | Make: Oldsmobile | 0.1694 | 4.19e-17 |
| Year: 2005-2009 | -0.2729 | 9.06e-43 | Make: Panoz | -0.0402 | 0.0474 |
| Year: 2010-2014 | -0.2570 | 5.71e-38 | Make: Plymouth | 0.0452 | 0.0259 |
| Make: Acura | -0.3662 | 5.09e-78 | Make: Pontiac | 0.1203 | 2.66e-09 |
| Make: AM General | 0.1799 | 3.94e-19 | Make: Porsche | -0.3172 | 6.45e-58 |
| Make: Aston Martin | -0.1173 | 6.6e-09 | Make: Ram | 0.1174 | 6.41e-09 |
| Make: Audi | -0.4422 | 6.62e-117 | Make: Rolls-Royce | -0.2056 | 1.33e-24 |
| Make: Bentley | -0.0002 | 0.992 | Make: Saab | -0.3364 | 2.3e-65 |
| Make: BMW | -0.4488 | 9.54e-121 | Make: Saturn | 0.1309 | 9.42e-11 |
| Make: Buick | 0.1495 | 1.28e-13 | Make: Scion | -0.0187 | 0.358 |
| Make: Cadillac | 0.1507 | 8.26e-14 | Make: Smart | -0.2235 | 6.88e-29 |
| Make: Chevrolet | 0.3650 | 1.81e-77 | Make: Subaru | -0.1721 | 1.32e-17 |
| Make: Chrysler | 0.0675 | 0.000878 | Make: Suzuki | -0.0502 | 0.0133 |
| Make: Daewoo | -0.0027 | 0.893 | Make: Tesla | -0.0891 | 1.1e-05 |
| Make: Dodge | 0.4096 | 5.97e-99 | Make: Toyota | -0.2077 | 4.24e-25 |
| Make: Eagle | 0.0409 | 0.0438 | Make: Volkswagen | -0.3294 | 1.31e-62 |
| Make: Ferrari | -0.1439 | 1.02e-12 | Make: Volvo | -0.2526 | 1.08e-36 |

Table 5.7: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between the percentage of residents with a some amount of college-level education and each car attribute, at the zip code level.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | 0.4762 | 9.32e-138 | Make: Fiat | 0.1001 | 7.63e-07 |
| Cars/Image | 0.1769 | 1.53e-18 | Make: Fisker | 0.1594 | 2.75e-15 |
| MPG Highway | 0.4188 | 8.61e-104 | Make: Ford | -0.4310 | 1.68e-110 |
| MPG City | 0.4928 | 6.96e-149 | Make: Geo | -0.1954 | 2.48e-22 |
| Hybrid | 0.1709 | 2.16e-17 | Make: GMC | -0.3045 | 2.61e-53 |
| Electric | 0.2644 | 3.8e-40 | Make: Honda | 0.5044 | 5.55e-157 |
| Foreign | 0.5761 | 7.19e-215 | Make: Hummer | 0.0296 | 0.145 |
| Country: England | 0.4858 | 3.69e-144 | Make: Hyundai | 0.2421 | 9.39e-34 |
| Country: Germany | 0.5990 | 1.41e-236 | Make: Infiniti | 0.3671 | 2.2e-78 |
| Country: Italy | 0.2677 | 3.81e-41 | Make: Isuzu | -0.1013 | 5.61e-07 |
| Country: Japan | 0.5101 | 4.13e-161 | Make: Jaguar | 0.0730 | 0.000314 |
| Country: South Korea | 0.2224 | 1.28e-28 | Make: Jeep | 0.1450 | 6.77e-13 |
| Country: Sweden | 0.4482 | 2.03e-120 | Make: Kia | 0.0877 | 1.51e-05 |
| Country: USA | -0.5761 | 7.19e-215 | Make: Lamborghini | 0.2240 | 5.22e-29 |
| Body Type: Convertible | 0.2604 | 5.8e-39 | Make: Land Rover | 0.3980 | 4.81e-93 |
| Body Type: Coupe | -0.1435 | 1.17e-12 | Make: Lexus | 0.4678 | 2.14e-132 |
| Body Type: Crew Cab | -0.1952 | 2.66e-22 | Make: Lincoln | -0.1939 | 5.15e-22 |
| Body Type: Extended Cab | -0.3285 | 3e-62 | Make: Lotus | 0.1168 | 7.61e-09 |
| Body Type: Hatchback | 0.5563 | 1.63e-197 | Make: Maserati | 0.1667 | 1.33e-16 |
| Body Type: Minivan | -0.0054 | 0.789 | Make: Maybach | 0.0604 | 0.00292 |
| Body Type: Regular Cab | -0.4164 | 1.62e-102 | Make: Mazda | 0.3960 | 4.65e-92 |
| Body Type: Sedan | 0.0435 | 0.0322 | Make: Mclaren | 0.1201 | 2.86e-09 |
| Body Type: SUV | 0.3200 | 5.67e-59 | Make: Mercedes-Benz | 0.4196 | 2.85e-104 |
| Body Type: Van | -0.1251 | 6.18e-10 | Make: Mercury | -0.2760 | 9.64e-44 |
| Body Type: Wagon | 0.4356 | 4.55e-113 | Make: Mini | 0.4333 | 8.48e-112 |
| Year: 1990-1994 | -0.4318 | 5.94e-111 | Make: Mitsubishi | -0.0962 | 2.04e-06 |
| Year: 1995-1999 | -0.4575 | 5.71e-126 | Make: Nissan | 0.0588 | 0.00375 |
| Year: 2000-2004 | 0.2168 | 3.01e-27 | Make: Oldsmobile | -0.3223 | 7.67e-60 |
| Year: 2005-2009 | 0.4910 | 1.18e-147 | Make: Panoz | 0.1190 | 4.04e-09 |
| Year: 2010-2014 | 0.4297 | 8.57e-110 | Make: Plymouth | -0.2432 | 4.71e-34 |
| Make: Acura | 0.3970 | 1.45e-92 | Make: Pontiac | -0.2527 | 1.01e-36 |
| Make: AM General | -0.1202 | 2.76e-09 | Make: Porsche | 0.3995 | 8.61e-94 |
| Make: Aston Martin | 0.1597 | 2.35e-15 | Make: Ram | -0.0559 | 0.00582 |
| Make: Audi | 0.5247 | 4.77e-172 | Make: Rolls-Royce | 0.1492 | 1.43e-13 |
| Make: Bentley | -0.0555 | 0.00625 | Make: Saab | 0.3839 | 3.61e-86 |
| Make: BMW | 0.4894 | 1.41e-146 | Make: Saturn | 0.0338 | 0.0959 |
| Make: Buick | -0.3060 | 7.86e-54 | Make: Scion | 0.2404 | 2.71e-33 |
| Make: Cadillac | -0.3432 | 3.96e-68 | Make: Smart | 0.2721 | 1.68e-42 |
| Make: Chevrolet | -0.5134 | 1.48e-163 | Make: Subaru | 0.3973 | 1.04e-92 |
| Make: Chrysler | -0.2066 | 7.9e-25 | Make: Suzuki | 0.0255 | 0.209 |
| Make: Daewoo | -0.0122 | 0.548 | Make: Tesla | 0.1318 | 7.03e-11 |
| Make: Dodge | -0.5082 | 8.88e-160 | Make: Toyota | 0.4168 | 9.85e-103 |
| Make: Eagle | -0.1532 | 3.08e-14 | Make: Volkswagen | 0.5222 | 4.34e-170 |
| Make: Ferrari | 0.1161 | 9.6e-09 | Make: Volvo | 0.4221 | 1.32e-105 |

Table 5.8: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between the percentage of residents with a bachelor's degree and each car attribute, at the zip code level.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | 0.4799 | 3.15e-140 | Make: Fiat | 0.1333 | 4.16e-11 |
| Cars/Image | 0.2212 | 2.65e-28 | Make: Fisker | 0.2006 | 1.81e-23 |
| MPG Highway | 0.4867 | 9.11e-145 | Make: Ford | -0.4457 | 6.03e-119 |
| MPG City | 0.5451 | 3.44e-188 | Make: Geo | -0.2069 | 6.79e-25 |
| Hybrid | 0.2256 | 2.05e-29 | Make: GMC | -0.3500 | 6.06e-71 |
| Electric | 0.3236 | 2.45e-60 | Make: Honda | 0.5075 | 3.17e-159 |
| Foreign | 0.5775 | 3.65e-216 | Make: Hummer | 0.0311 | 0.125 |
| Country: England | 0.5434 | 8.86e-187 | Make: Hyundai | 0.2812 | 2.12e-45 |
| Country: Germany | 0.6397 | 8.72e-280 | Make: Infiniti | 0.3845 | 1.91e-86 |
| Country: Italy | 0.3448 | 8.8e-69 | Make: Isuzu | -0.1266 | 3.78e-10 |
| Country: Japan | 0.4903 | 3.53e-147 | Make: Jaguar | 0.1406 | 3.33e-12 |
| Country: South Korea | 0.2560 | 1.16e-37 | Make: Jeep | 0.1283 | 2.21e-10 |
| Country: Sweden | 0.5266 | 1.77e-173 | Make: Kia | 0.0941 | 3.34e-06 |
| Country: USA | -0.5775 | 3.65e-216 | Make: Lamborghini | 0.2799 | 5.56e-45 |
| Body Type: Convertible | 0.2634 | 7.39e-40 | Make: Land Rover | 0.4297 | 8.43e-110 |
| Body Type: Coupe | -0.1454 | 6.02e-13 | Make: Lexus | 0.4415 | 1.86e-116 |
| Body Type: Crew Cab | -0.2882 | 1.05e-47 | Make: Lincoln | -0.1016 | 5.15e-07 |
| Body Type: Extended Cab | -0.4198 | 2.35e-104 | Make: Lotus | 0.1358 | 1.8e-11 |
| Body Type: Hatchback | 0.5848 | 6.04e-223 | Make: Maserati | 0.2035 | 3.99e-24 |
| Body Type: Minivan | -0.0024 | 0.905 | Make: Maybach | 0.0808 | 6.63e-05 |
| Body Type: Regular Cab | -0.4727 | 1.7e-135 | Make: Mazda | 0.3948 | 1.91e-91 |
| Body Type: Sedan | 0.1410 | 2.89e-12 | Make: Mclaren | 0.1495 | 1.27e-13 |
| Body Type: SUV | 0.2720 | 1.79e-42 | Make: Mercedes-Benz | 0.4664 | 1.53e-131 |
| Body Type: Van | -0.0593 | 0.00345 | Make: Mercury | -0.1878 | 1.01e-20 |
| Body Type: Wagon | 0.4746 | 9.77e-137 | Make: Mini | 0.4604 | 9.23e-128 |
| Year: 1990-1994 | -0.4526 | 4.82e-123 | Make: Mitsubishi | -0.1283 | 2.21e-10 |
| Year: 1995-1999 | -0.4365 | 1.25e-113 | Make: Nissan | 0.0365 | 0.0721 |
| Year: 2000-2004 | 0.1785 | 7.62e-19 | Make: Oldsmobile | -0.2907 | 1.59e-48 |
| Year: 2005-2009 | 0.5065 | 1.59e-158 | Make: Panoz | 0.1316 | 7.47e-11 |
| Year: 2010-2014 | 0.4501 | 1.58e-121 | Make: Plymouth | -0.2022 | 7.88e-24 |
| Make: Acura | 0.4350 | 9.08e-113 | Make: Pontiac | -0.2142 | 1.3e-26 |
| Make: AM General | -0.1582 | 4.44e-15 | Make: Porsche | 0.4353 | 6.14e-113 |
| Make: Aston Martin | 0.2038 | 3.39e-24 | Make: Ram | -0.0590 | 0.00363 |
| Make: Audi | 0.5867 | 8.92e-225 | Make: Rolls-Royce | 0.2186 | 1.11e-27 |
| Make: Bentley | -0.0199 | 0.326 | Make: Saab | 0.4599 | 1.93e-127 |
| Make: BMW | 0.5301 | 3.5e-176 | Make: Saturn | 0.0157 | 0.441 |
| Make: Buick | -0.2406 | 2.42e-33 | Make: Scion | 0.1821 | 1.49e-19 |
| Make: Cadillac | -0.3113 | 9.66e-56 | Make: Smart | 0.3045 | 2.64e-53 |
| Make: Chevrolet | -0.5376 | 4.31e-182 | Make: Subaru | 0.4357 | 4.02e-113 |
| Make: Chrysler | -0.1708 | 2.34e-17 | Make: Suzuki | 0.0211 | 0.298 |
| Make: Daewoo | 0.0039 | 0.849 | Make: Tesla | 0.1434 | 1.22e-12 |
| Make: Dodge | -0.5311 | 5.83e-177 | Make: Toyota | 0.3773 | 4.59e-83 |
| Make: Eagle | -0.1099 | 5.66e-08 | Make: Volkswagen | 0.5278 | 2.34e-174 |
| Make: Ferrari | 0.1898 | 3.87e-21 | Make: Volvo | 0.4940 | 1.02e-149 |

Table 5.9: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between the percentage of residents with a graduate or professional degree and each car attribute, at the zip code level.

| | %Less Than High School | | %High School | | %Some College | |
|---|---|---|---|---|---|---|
| Rank | Variable | Pearson's $r$ | Variable | Pearson's $r$ | Variable | Pearson's $r$ |
| 1 | Year: 1995-1999 | 0.3715 | Country: USA | 0.6048 | Body Type: Extended Cab | 0.4391 |
| 2 | Year: 1990-1994 | 0.3230 | Make: Dodge | 0.5527 | Body Type: Regular Cab | 0.4194 |
| 3 | Make: Chevrolet | 0.3183 | Make: Chevrolet | 0.4747 | Make: Dodge | 0.4096 |
| 4 | Make: Ford | 0.2991 | Make: Buick | 0.4488 | Country: USA | 0.3834 |
| 5 | Country: USA | 0.2955 | Year: 1995-1999 | 0.4220 | Make: Chevrolet | 0.3650 |
| 84 | Make: Hyundai | -0.3468 | Make: Honda | -0.5371 | Cars/Image | -0.4257 |
| 85 | Country: Sweden | -0.3541 | Body Type: Hatchback | -0.5576 | Make: Audi | -0.4422 |
| 86 | Make: Subaru | -0.3597 | Country: Japan | -0.5670 | Country: England | -0.4480 |
| 87 | Year: 2005-2009 | -0.3831 | Foreign | -0.6048 | Make: BMW | -0.4488 |
| 88 | Body Type: Wagon | -0.3888 | Country: Germany | -0.6142 | Country: Germany | -0.4791 |

| | %Bachelor's Degree | | %Graduate Degree | |
|---|---|---|---|---|
| Rank | Variable | Pearson's $r$ | Variable | Pearson's $r$ |
| 1 | Country: Germany | 0.5990 | Country: Germany | 0.6397 |
| 2 | Foreign | 0.5761 | Make: Audi | 0.5867 |
| 3 | Body Type: Hatchback | 0.5563 | Body Type: Hatchback | 0.5848 |
| 4 | Make: Audi | 0.5247 | Foreign | 0.5775 |
| 5 | Make: Volkswagen | 0.5222 | MPG City | 0.5451 |
| 84 | Year: 1990-1994 | -0.4318 | Year: 1990-1994 | -0.4526 |
| 85 | Year: 1995-1999 | -0.4575 | Body Type: Regular Cab | -0.4727 |
| 86 | Make: Dodge | -0.5082 | Make: Dodge | -0.5311 |
| 87 | Make: Chevrolet | -0.5134 | Make: Chevrolet | -0.5376 |
| 88 | Country: USA | -0.5761 | Country: USA | -0.5775 |

Table 5.10: The five car attributes that correlate most positively and most negatively with the percentage of each education level in zip code.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | 0.2182 | 1.39e-27 | Make: Fiat | 0.0370 | 0.0679 |
| Cars/Image | -0.1478 | 2.42e-13 | Make: Fisker | 0.0113 | 0.579 |
| MPG Highway | -0.0753 | 0.000205 | Make: Ford | 0.0559 | 0.00582 |
| MPG City | -0.0008 | 0.967 | Make: Geo | -0.0557 | 0.00606 |
| Hybrid | -0.0068 | 0.739 | Make: GMC | 0.0536 | 0.00819 |
| Electric | 0.0560 | 0.00575 | Make: Honda | 0.0045 | 0.823 |
| Foreign | 0.0358 | 0.0778 | Make: Hummer | 0.0877 | 1.49e-05 |
| Country: England | 0.0422 | 0.0376 | Make: Hyundai | 0.1538 | 2.46e-14 |
| Country: Germany | 0.0572 | 0.00478 | Make: Infiniti | -0.1273 | 3.03e-10 |
| Country: Italy | 0.0290 | 0.153 | Make: Isuzu | -0.0218 | 0.283 |
| Country: Japan | 0.0075 | 0.712 | Make: Jaguar | -0.2120 | 4.21e-26 |
| Country: South Korea | 0.1635 | 4.96e-16 | Make: Jeep | 0.2893 | 4.6e-48 |
| Country: Sweden | 0.0749 | 0.000219 | Make: Kia | 0.1149 | 1.34e-08 |
| Country: USA | -0.0358 | 0.0778 | Make: Lamborghini | 0.0407 | 0.0451 |
| Body Type: Convertible | 0.0727 | 0.000332 | Make: Land Rover | 0.0686 | 0.000718 |
| Body Type: Coupe | -0.1948 | 3.29e-22 | Make: Lexus | -0.0263 | 0.194 |
| Body Type: Crew Cab | 0.1986 | 4.98e-23 | Make: Lincoln | -0.3003 | 7.7e-52 |
| Body Type: Extended Cab | 0.2041 | 2.97e-24 | Make: Lotus | 0.0438 | 0.0308 |
| Body Type: Hatchback | 0.1702 | 2.95e-17 | Make: Maserati | -0.0308 | 0.129 |
| Body Type: Minivan | -0.0093 | 0.647 | Make: Maybach | -0.0355 | 0.08 |
| Body Type: Regular Cab | 0.1237 | 9.42e-10 | Make: Mazda | 0.0734 | 0.000295 |
| Body Type: Sedan | -0.4181 | 1.84e-103 | Make: Mclaren | 0.0066 | 0.745 |
| Body Type: SUV | 0.3053 | 1.37e-53 | Make: Mercedes-Benz | -0.1011 | 5.86e-07 |
| Body Type: Van | -0.1390 | 6.01e-12 | Make: Mercury | -0.2581 | 2.76e-38 |
| Body Type: Wagon | 0.2153 | 7.21e-27 | Make: Mini | 0.1182 | 5.13e-09 |
| Year: 1990-1994 | -0.1668 | 1.28e-16 | Make: Mitsubishi | -0.0663 | 0.00107 |
| Year: 1995-1999 | -0.2599 | 8.08e-39 | Make: Nissan | -0.1289 | 1.8e-10 |
| Year: 2000-2004 | 0.1514 | 6.23e-14 | Make: Oldsmobile | -0.2065 | 8.01e-25 |
| Year: 2005-2009 | 0.2104 | 1e-25 | Make: Panoz | 0.0313 | 0.123 |
| Year: 2010-2014 | 0.1864 | 1.97e-20 | Make: Plymouth | -0.1291 | 1.69e-10 |
| Make: Acura | -0.0497 | 0.0142 | Make: Pontiac | -0.1455 | 5.67e-13 |
| Make: AM General | 0.0649 | 0.00137 | Make: Porsche | 0.0839 | 3.48e-05 |
| Make: Aston Martin | 0.0098 | 0.629 | Make: Ram | 0.0635 | 0.00174 |
| Make: Audi | 0.1198 | 3.18e-09 | Make: Rolls-Royce | -0.0234 | 0.25 |
| Make: Bentley | -0.1299 | 1.29e-10 | Make: Saab | 0.1443 | 8.86e-13 |
| Make: BMW | -0.0357 | 0.0789 | Make: Saturn | 0.0590 | 0.00359 |
| Make: Buick | -0.2529 | 8.89e-37 | Make: Scion | 0.1415 | 2.42e-12 |
| Make: Cadillac | -0.3535 | 1.93e-72 | Make: Smart | 0.1069 | 1.29e-07 |
| Make: Chevrolet | 0.0097 | 0.632 | Make: Subaru | 0.2397 | 4.34e-33 |
| Make: Chrysler | -0.2134 | 1.96e-26 | Make: Suzuki | 0.0349 | 0.0855 |
| Make: Daewoo | -0.0324 | 0.111 | Make: Tesla | 0.0339 | 0.0946 |
| Make: Dodge | 0.0648 | 0.00139 | Make: Toyota | 0.0034 | 0.867 |
| Make: Eagle | -0.1144 | 1.55e-08 | Make: Volkswagen | 0.1827 | 1.09e-19 |
| Make: Ferrari | -0.0134 | 0.509 | Make: Volvo | 0.0531 | 0.00887 |

Table 5.11: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between the percentage of white residents and each car attribute, at the zip code level.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | -0.1895 | 4.4e-21 | Make: Fiat | -0.0250 | 0.217 |
| Cars/Image | -0.0710 | 0.000459 | Make: Fisker | 0.0304 | 0.134 |
| MPG Highway | 0.0465 | 0.0218 | Make: Ford | 0.0432 | 0.033 |
| MPG City | -0.0832 | 4.05e-05 | Make: Geo | 0.0196 | 0.335 |
| Hybrid | 0.0466 | 0.0217 | Make: GMC | 0.0281 | 0.167 |
| Electric | -0.0942 | 3.32e-06 | Make: Honda | -0.2304 | 1.24e-30 |
| Foreign | -0.2580 | 2.97e-38 | Make: Hummer | -0.0968 | 1.76e-06 |
| Country: England | -0.0640 | 0.00161 | Make: Hyundai | 0.0248 | 0.221 |
| Country: Germany | -0.1650 | 2.75e-16 | Make: Infiniti | 0.0334 | 0.0998 |
| Country: Italy | -0.0033 | 0.87 | Make: Isuzu | -0.0659 | 0.00116 |
| Country: Japan | -0.2855 | 8.43e-47 | Make: Jaguar | 0.2494 | 8.76e-36 |
| Country: South Korea | 0.0273 | 0.178 | Make: Jeep | -0.1016 | 5.19e-07 |
| Country: Sweden | -0.0272 | 0.181 | Make: Kia | 0.0188 | 0.355 |
| Country: USA | 0.2580 | 2.97e-38 | Make: Lamborghini | -0.0171 | 0.399 |
| Body Type: Convertible | -0.0265 | 0.191 | Make: Land Rover | -0.0994 | 9.16e-07 |
| Body Type: Coupe | 0.2151 | 7.87e-27 | Make: Lexus | -0.1338 | 3.58e-11 |
| Body Type: Crew Cab | -0.1996 | 2.93e-23 | Make: Lincoln | 0.4088 | 1.57e-98 |
| Body Type: Extended Cab | -0.2583 | 2.52e-38 | Make: Lotus | -0.0515 | 0.0111 |
| Body Type: Hatchback | -0.2692 | 1.32e-41 | Make: Maserati | 0.0424 | 0.0366 |
| Body Type: Minivan | -0.0707 | 0.00049 | Make: Maybach | 0.0712 | 0.000443 |
| Body Type: Regular Cab | -0.1403 | 3.68e-12 | Make: Mazda | -0.1301 | 1.22e-10 |
| Body Type: Sedan | 0.4421 | 7.52e-117 | Make: Mclaren | -0.0252 | 0.215 |
| Body Type: SUV | -0.2267 | 1.05e-29 | Make: Mercedes-Benz | -0.0204 | 0.314 |
| Body Type: Van | 0.0799 | 7.98e-05 | Make: Mercury | 0.4479 | 3.12e-120 |
| Body Type: Wagon | -0.1386 | 6.71e-12 | Make: Mini | -0.1564 | 9.09e-15 |
| Year: 1990-1994 | 0.0795 | 8.65e-05 | Make: Mitsubishi | 0.0176 | 0.386 |
| Year: 1995-1999 | 0.2483 | 1.86e-35 | Make: Nissan | -0.0285 | 0.16 |
| Year: 2000-2004 | -0.1324 | 5.73e-11 | Make: Oldsmobile | 0.3670 | 2.24e-78 |
| Year: 2005-2009 | -0.1417 | 2.25e-12 | Make: Panoz | 0.1011 | 5.96e-07 |
| Year: 2010-2014 | -0.1408 | 3.11e-12 | Make: Plymouth | 0.2056 | 1.3e-24 |
| Make: Acura | -0.0751 | 0.000212 | Make: Pontiac | 0.3529 | 3.4e-72 |
| Make: AM General | -0.0647 | 0.00142 | Make: Porsche | -0.0925 | 4.9e-06 |
| Make: Aston Martin | 0.0086 | 0.671 | Make: Ram | 0.0667 | 0.000998 |
| Make: Audi | -0.1189 | 4.17e-09 | Make: Rolls-Royce | 0.0955 | 2.39e-06 |
| Make: Bentley | 0.1577 | 5.27e-15 | Make: Saab | -0.0704 | 0.000516 |
| Make: BMW | -0.1151 | 1.29e-08 | Make: Saturn | 0.0510 | 0.0119 |
| Make: Buick | 0.4922 | 1.73e-148 | Make: Scion | -0.2472 | 3.83e-35 |
| Make: Cadillac | 0.5015 | 6.37e-155 | Make: Smart | -0.1127 | 2.58e-08 |
| Make: Chevrolet | 0.1058 | 1.71e-07 | Make: Subaru | -0.1821 | 1.46e-19 |
| Make: Chrysler | 0.4137 | 4.29e-101 | Make: Suzuki | -0.0578 | 0.00436 |
| Make: Daewoo | 0.0187 | 0.358 | Make: Tesla | -0.0672 | 0.000924 |
| Make: Dodge | 0.1010 | 6.02e-07 | Make: Toyota | -0.3335 | 3.34e-64 |
| Make: Eagle | 0.2003 | 2.04e-23 | Make: Volkswagen | -0.2372 | 1.97e-32 |
| Make: Ferrari | 0.0199 | 0.326 | Make: Volvo | -0.0153 | 0.451 |

Table 5.12: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between the percentage of black residents and each car attribute, at the zip code level.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | 0.1130 | 2.32e-08 | Make: Fiat | -0.0092 | 0.651 |
| Cars/Image | 0.3179 | 3.54e-58 | Make: Fisker | -0.0110 | 0.587 |
| MPG Highway | 0.2580 | 2.93e-38 | Make: Ford | -0.3324 | 8.99e-64 |
| MPG City | 0.3379 | 5.95e-66 | Make: Geo | -0.0159 | 0.433 |
| Hybrid | 0.0140 | 0.49 | Make: GMC | -0.2607 | 4.7e-39 |
| Electric | 0.1512 | 6.73e-14 | Make: Honda | 0.5174 | 1.56e-166 |
| Foreign | 0.5162 | 1.34e-165 | Make: Hummer | -0.0058 | 0.775 |
| Country: England | 0.1797 | 4.47e-19 | Make: Hyundai | -0.1209 | 2.28e-09 |
| Country: Germany | 0.3445 | 1.14e-68 | Make: Infiniti | 0.2498 | 6.76e-36 |
| Country: Italy | 0.0566 | 0.00525 | Make: Isuzu | 0.0369 | 0.0687 |
| Country: Japan | 0.5727 | 7.39e-212 | Make: Jaguar | 0.0279 | 0.169 |
| Country: South Korea | -0.1476 | 2.67e-13 | Make: Jeep | -0.2754 | 1.51e-43 |
| Country: Sweden | 0.0857 | 2.33e-05 | Make: Kia | -0.1391 | 5.78e-12 |
| Country: USA | -0.5162 | 1.34e-165 | Make: Lamborghini | 0.0505 | 0.0127 |
| Body Type: Convertible | 0.0381 | 0.0602 | Make: Land Rover | 0.1286 | 1.98e-10 |
| Body Type: Coupe | -0.0233 | 0.251 | Make: Lexus | 0.4142 | 2.16e-101 |
| Body Type: Crew Cab | -0.1247 | 6.92e-10 | Make: Lincoln | -0.1779 | 1e-18 |
| Body Type: Extended Cab | -0.1183 | 4.99e-09 | Make: Lotus | 0.0609 | 0.00265 |
| Body Type: Hatchback | 0.3293 | 1.54e-62 | Make: Maserati | 0.0355 | 0.0804 |
| Body Type: Minivan | 0.1799 | 4.09e-19 | Make: Maybach | -0.0151 | 0.458 |
| Body Type: Regular Cab | -0.1631 | 5.93e-16 | Make: Mazda | 0.2303 | 1.33e-30 |
| Body Type: Sedan | 0.0961 | 2.05e-06 | Make: Mclaren | 0.0742 | 0.00025 |
| Body Type: SUV | -0.1002 | 7.39e-07 | Make: Mercedes-Benz | 0.3549 | 4.99e-73 |
| Body Type: Van | 0.0378 | 0.0628 | Make: Mercury | -0.2835 | 3.67e-46 |
| Body Type: Wagon | 0.0187 | 0.356 | Make: Mini | 0.1919 | 1.41e-21 |
| Year: 1990-1994 | -0.0363 | 0.0739 | Make: Mitsubishi | 0.0137 | 0.5 |
| Year: 1995-1999 | -0.1500 | 1.08e-13 | Make: Nissan | 0.1730 | 8.91e-18 |
| Year: 2000-2004 | 0.0434 | 0.0324 | Make: Oldsmobile | -0.2771 | 4.28e-44 |
| Year: 2005-2009 | 0.0888 | 1.17e-05 | Make: Panoz | -0.1239 | 8.9e-10 |
| Year: 2010-2014 | 0.0879 | 1.43e-05 | Make: Plymouth | -0.1663 | 1.56e-16 |
| Make: Acura | 0.3251 | 6.24e-61 | Make: Pontiac | -0.3100 | 2.91e-55 |
| Make: AM General | -0.0515 | 0.0111 | Make: Porsche | 0.1269 | 3.48e-10 |
| Make: Aston Martin | 0.0266 | 0.19 | Make: Ram | -0.1782 | 8.81e-19 |
| Make: Audi | 0.1654 | 2.31e-16 | Make: Rolls-Royce | -0.0467 | 0.0213 |
| Make: Bentley | -0.0346 | 0.0878 | Make: Saab | 0.0064 | 0.753 |
| Make: BMW | 0.3801 | 2.18e-84 | Make: Saturn | -0.1075 | 1.08e-07 |
| Make: Buick | -0.3356 | 4.73e-65 | Make: Scion | 0.2085 | 2.85e-25 |
| Make: Cadillac | -0.2621 | 1.86e-39 | Make: Smart | 0.0740 | 0.000261 |
| Make: Chevrolet | -0.3734 | 2.81e-81 | Make: Subaru | 0.0245 | 0.226 |
| Make: Chrysler | -0.2971 | 1.06e-50 | Make: Suzuki | 0.0179 | 0.378 |
| Make: Daewoo | 0.0249 | 0.22 | Make: Tesla | 0.0979 | 1.33e-06 |
| Make: Dodge | -0.4053 | 9.68e-97 | Make: Toyota | 0.6340 | 2.59e-273 |
| Make: Eagle | -0.1243 | 7.72e-10 | Make: Volkswagen | 0.2052 | 1.66e-24 |
| Make: Ferrari | 0.0504 | 0.0129 | Make: Volvo | 0.0953 | 2.51e-06 |

Table 5.13: Pearson $r$ correlation coefficients and their associated $p$-values for each car attribute between the percentage of Asian residents and each car attribute, at the zip code level.

| | %White | | %Black | | %Asian | |
|---|---|---|---|---|---|---|
| Rank | Variable | Pearson's $r$ | Variable | Pearson's $r$ | Variable | Pearson's $r$ |
| 1 | Body Type: SUV | 0.3053 | Make: Cadillac | 0.5015 | Make: Toyota | 0.6340 |
| 2 | Make: Jeep | 0.2893 | Make: Buick | 0.4922 | Country: Japan | 0.5727 |
| 3 | Make: Subaru | 0.2397 | Make: Mercury | 0.4479 | Make: Honda | 0.5174 |
| 4 | Price | 0.2182 | Body Type: Sedan | 0.4421 | Foreign | 0.5162 |
| 5 | Body Type: Wagon | 0.2153 | Make: Chrysler | 0.4137 | Make: Lexus | 0.4142 |
| 84 | Make: Mercury | -0.2581 | Foreign | -0.2580 | Make: Ford | -0.3324 |
| 85 | Year: 1995-1999 | -0.2599 | Body Type: Extended Cab | -0.2583 | Make: Buick | -0.3356 |
| 86 | Make: Lincoln | -0.3003 | Body Type: Hatchback | -0.2692 | Make: Chevrolet | -0.3734 |
| 87 | Make: Cadillac | -0.3535 | Country: Japan | -0.2855 | Make: Dodge | -0.4053 |
| 88 | Body Type: Sedan | -0.4181 | Make: Toyota | -0.3335 | Country: USA | -0.5162 |

Table 5.14: The five car attributes that correlate most positively and most negatively with the percentage of each race in a zip code.

| Variable | Pearson's $r$ | $p$-value | Variable | Pearson's $r$ | $p$-value |
|---|---|---|---|---|---|
| Price | -0.2768 | $\leq 10^{-300}$ | Make: Fiat | 0.0165 | 0.00815 |
| Cars/Image | 0.3718 | $\leq 10^{-300}$ | Make: Fisker | 0.0173 | 0.00543 |
| MPG Highway | 0.3307 | $\leq 10^{-300}$ | Make: Ford | -0.1746 | 7.41e-176 |
| MPG City | 0.2597 | $\leq 10^{-300}$ | Make: Geo | 0.0681 | 6.63e-28 |
| Hybrid | 0.0318 | 3.1e-07 | Make: GMC | -0.1675 | 8.16e-162 |
| Electric | 0.0347 | 2.35e-08 | Make: Honda | 0.0705 | 8.59e-30 |
| Foreign | 0.0743 | 5.93e-33 | Make: Hummer | -0.0587 | 3.72e-21 |
| Country: England | 0.1159 | 6.69e-78 | Make: Hyundai | -0.0162 | 0.00938 |
| Country: Germany | 0.1665 | 7.7e-160 | Make: Infiniti | 0.0772 | 2.08e-35 |
| Country: Italy | 0.0563 | 1.32e-19 | Make: Isuzu | 0.0369 | 3.12e-09 |
| Country: Japan | 0.0297 | 1.83e-06 | Make: Jaguar | 0.1180 | 1.09e-80 |
| Country: South Korea | -0.0150 | 0.0158 | Make: Jeep | -0.0563 | 1.39e-19 |
| Country: Sweden | 0.1509 | 1.96e-131 | Make: Kia | -0.0076 | 0.22 |
| Country: USA | -0.0743 | 5.93e-33 | Make: Lamborghini | 0.0366 | 4.03e-09 |
| Body Type: Convertible | 0.0144 | 0.0206 | Make: Land Rover | 0.0438 | 1.87e-12 |
| Body Type: Coupe | 0.1426 | 2.52e-117 | Make: Lexus | -0.0578 | 1.5e-20 |
| Body Type: Crew Cab | -0.4799 | $\leq 10^{-300}$ | Make: Lincoln | 0.1387 | 3.71e-111 |
| Body Type: Extended Cab | -0.4266 | $\leq 10^{-300}$ | Make: Lotus | 0.0147 | 0.0178 |
| Body Type: Hatchback | 0.1193 | 1.6e-82 | Make: Maserati | 0.0291 | 2.94e-06 |
| Body Type: Minivan | 0.0524 | 3.38e-17 | Make: Maybach | 0.0160 | 0.01 |
| Body Type: Regular Cab | -0.3047 | $\leq 10^{-300}$ | Make: Mazda | 0.0801 | 5.49e-38 |
| Body Type: Sedan | 0.4829 | $\leq 10^{-300}$ | Make: Mclaren | 0.0301 | 1.3e-06 |
| Body Type: SUV | -0.2246 | 1e-292 | Make: Mercedes-Benz | 0.0830 | 1.08e-40 |
| Body Type: Van | 0.1154 | 2.73e-77 | Make: Mercury | 0.1830 | 2.67e-193 |
| Body Type: Wagon | 0.1463 | 1.97e-123 | Make: Mini | 0.0700 | 2.19e-29 |
| Year: 1990-1994 | 0.1396 | 1.89e-112 | Make: Mitsubishi | 0.0101 | 0.104 |
| Year: 1995-1999 | 0.2950 | $\leq 10^{-300}$ | Make: Nissan | 0.0212 | 0.000668 |
| Year: 2000-2004 | -0.1728 | 3.4e-172 | Make: Oldsmobile | 0.2250 | 9.64e-294 |
| Year: 2005-2009 | -0.1951 | 7.58e-220 | Make: Panoz | 0.0128 | 0.0392 |
| Year: 2010-2014 | -0.1651 | 2.71e-157 | Make: Plymouth | 0.1696 | 7.47e-166 |
| Make: Acura | 0.1032 | 3.93e-62 | Make: Pontiac | 0.2191 | 3.61e-278 |
| Make: AM General | -0.1182 | 6.02e-81 | Make: Porsche | 0.0089 | 0.153 |
| Make: Aston Martin | 0.0005 | 0.931 | Make: Ram | -0.0855 | 4.51e-43 |
| Make: Audi | 0.0951 | 6.14e-53 | Make: Rolls-Royce | 0.0537 | 6.09e-18 |
| Make: Bentley | 0.0481 | 1e-14 | Make: Saab | 0.0784 | 1.86e-36 |
| Make: BMW | 0.1203 | 8.72e-84 | Make: Saturn | 0.0358 | 8.69e-09 |
| Make: Buick | 0.2177 | 1.57e-274 | Make: Scion | -0.1093 | 1.66e-69 |
| Make: Cadillac | 0.1144 | 6.54e-76 | Make: Smart | 0.0141 | 0.0232 |
| Make: Chevrolet | -0.1842 | 8.44e-196 | Make: Subaru | 0.1414 | 1.72e-115 |
| Make: Chrysler | 0.1708 | 2.51e-168 | Make: Suzuki | 0.0396 | 1.98e-10 |
| Make: Daewoo | 0.0453 | 3.09e-13 | Make: Tesla | -0.0059 | 0.347 |
| Make: Dodge | -0.1010 | 1.43e-59 | Make: Toyota | -0.0545 | 1.97e-18 |
| Make: Eagle | 0.0776 | 9.26e-36 | Make: Volkswagen | 0.1529 | 8.78e-135 |
| Make: Ferrari | 0.0463 | 1.01e-13 | Make: Volvo | 0.1442 | 5.31e-120 |

Table 5.15: Pearson $r$ correlation coefficients and their associated $p$-values between each car attribute and %Obama. The variables "Price", "MPG City", and "MPG Highway" are calculated as expected values for each precinct, and all other variables are expressed as a percent of all cars observed in each precinct.

| Type of Crime | Pearson's $r$ | Spearman's $\rho$ | Quartile Acc. |
|---|---|---|---|
| Aggravated Assault | 0.6076 | 0.6024 | 79.8513 |
| Burglary | 0.7132 | 0.7070 | 88.8147 |
| Crime Against People | 0.6399 | 0.6395 | 83.5681 |
| Crime Against Property | 0.6535 | 0.6418 | 84.0410 |
| Homicide | 0.6115 | 0.6015 | 85.5230 |
| Larceny | 0.6302 | 0.6189 | 81.6974 |
| Motor Vehicle | 0.6473 | 0.6406 | 84.7603 |
| Robbery | 0.6628 | 0.6593 | 84.4774 |
| Rape | 0.6481 | 0.6513 | 82.6440 |

Table 5.16: Measures of the accuracy between our predicted crime levels and actual crime levels for each of the nine predicted types of crime. "Quartile Acc." (see Sec. 5.2.3) is the accuracy of distinguishing between zip codes in the top and bottom quartiles of each type of crime.

# Chapter 6

# Noisy Data for Fine-Grained Recognition

Current approaches for fine-grained recognition do the following: First, recruit experts to annotate a dataset of images, optionally also collecting more structured data in the form of part annotations and bounding boxes. Second, train a model utilizing this data. Toward the goal of solving fine-grained recognition, we introduce an alternative approach, leveraging free, noisy data from the web and simple, generic methods of recognition. This approach has benefits in both performance and scalability. We demonstrate its efficacy on four fine-grained datasets, greatly exceeding existing state of the art without the manual collection of even a single label, and furthermore show first results at scaling to more than 10,000 fine-grained categories. Quantitatively, we achieve top-1 accuracies of 92.3% on CUB-200-2011, 85.4% on Birdsnap, 93.4% on FGVC-Aircraft, and 80.8% on Stanford Dogs without using their annotated training sets. We compare our approach to an active learning approach for expanding fine-grained datasets.

This project was done while interning at Google Research, is joint work with Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duering, James Philbin, and Fei-Fei Li, and was published as [94].

## 6.1 Introduction

Fine-grained recognition refers to the task of distinguishing very similar categories, such as breeds of dogs [88, 115], species of birds [182, 174, 25, 20], or models of cars [195, 95]. Since its inception, great progress has been made, with accuracies on the popular CUB-200-2011 bird dataset [182] steadily increasing from 10.3% [182] to 84.6% [194].

The predominant approach in fine-grained recognition today consists of two steps. First, a dataset is collected. Since fine-grained recognition is a task inherently difficult for humans, this typically requires either recruiting a team of experts [174, 117] or extensive crowd-sourcing pipelines [95, 20]. Second, a method for recognition is trained using these expert-annotated labels, possibly also requiring additional annotations in the form of parts, attributes, or relationships [203, 83, 113, 25]. While methods following this approach have shown some success [25, 203, 113, 92], their performance and scalability is constrained by the paucity of data available due to these limitations. With this traditional approach it is prohibitive to scale up to all 14,000 species of birds in the world (Fig. 6.1), 278,000 species of butterflies and moths, or 941,000 species of insects [74].

In this paper, we show that it is possible to train effective models of fine-grained recognition using noisy data from the web and simple, generic methods of recognition [169, 168]. We demonstrate recognition abilities greatly exceeding current state of the art methods, achieving top-1 accuracies of 92.3% on CUB-200-2011 [182], 85.4% on Birdsnap [20], 93.4% on FGVC-Aircraft [117], and 80.8% on Stanford Dogs [88] *without using a single manually-annotated training label from the respective datasets.* On CUB, this is nearly at the level of human experts [26, 174]. Building upon this, we scale up the number of fine-grained classes recognized, reporting first results on over 10,000 species of birds and 14,000 species of butterflies and moths.

The rest of this paper proceeds as follows: After an overview of related work in Sec. 6.2, we provide an analysis of publicly-available noisy data for fine-grained recognition in Sec. 6.3, analyzing its quantity and quality. We describe a more traditional active learning approach for obtaining larger quantities of fine-grained data

Figure 6.1: There are more than 14,000 species of birds in the world. In this work we show that using noisy data from publicly-available online sources can not only improve recognition of categories in today's datasets, but also scale to very large numbers of fine-grained categories, which is extremely expensive with the traditional approach of manually collecting labels for fine-grained datasets. Here we show 4,225 of the 10,982 categories recognized in this work.

in Sec. 6.4, which serves as a comparison to purely using noisy data. We present extensive experiments in Sec. 6.5, and conclude with discussion in Sec. 6.6.

## 6.2 Related Work

**Fine-Grained Recognition.**

The majority of research in fine-grained recognition has focused on developing improved models for classification [10, 18, 25, 29, 31, 30, 46, 51, 56, 62, 61, 66, 92, 93, 113, 115, 132, 144, 159, 161, 160, 190, 192, 194, 196, 198, 197, 204, 205, 203, 206].

While these works have made great progress in modeling fine-grained categories given the limited data available, very few works have considered the impact of that data [194, 192, 174]. Xu *et al.* [194] augment datasets annotated with category labels and parts with web images in a multiple instance learning framework, and Xie *et al.* [192] do multitask training, where one task uses a ground truth fine-grained dataset and the other does not require fine-grained labels. While both of these methods have shown that augmenting fine-grained datasets with additional data can help, in our work we present results which completely forgo the use of any curated ground truth dataset. In one experiment hinting at the use of noisy data, Van Horn *et al.* [174] show the possibility of learning 40 bird classes from Flickr images. Our work validates and extends this idea, using similar intuition to significantly improve performance on existing fine-grained datasets and scale fine-grained recognition to over ten thousand categories, which we believe is necessary in order to fully explore the research direction.

Considerable work has also gone into the challenging task of curating fine-grained datasets [20, 174, 88, 95, 100, 176, 186, 182, 195] and developing interactive methods for recognition with a human in the loop [26, 181, 179, 183]. While these works have demonstrated effective strategies for collecting images of fine-grained categories, their scalability is ultimately limited by the requirement of manual annotation. Our work provides an alternative to these approaches.

**Learning from Noisy Data.**

Our work is also inspired by methods that propose to learn from web data [48, 35, 36, 154, 109, 58] or reason about label noise [126, 191, 174, 166, 147]. Works that use web data typically focus on detection and classification of a set of coarse-grained categories, but have not yet examined the fine-grained setting. Methods that reason about label noise have been divided in their results: some have shown that reasoning about label noise can have a substantial effect on recognition performance [190], while others demonstrate little change from reducing the noise level or having a noise-aware model [166, 147, 174]. In our work, we demonstrate that noisy data can be surprisingly effective for fine-grained recognition, providing evidence in support of the

latter hypothesis.

## 6.3 Noisy Fine-Grained Data

In this section we provide an analysis of the imagery publicly available for fine-grained recognition, which we collect via web search.[1] We describe its quantity, distribution, and levels of noise, reporting each on multiple fine-grained domains.

### 6.3.1 Categories

We consider four domains of fine-grained categories: birds, aircraft, Lepidoptera (a taxonomic order including butterflies and moths), and dogs. For birds and Lepidoptera, we obtained lists of fine-grained categories from Wikipedia, resulting in 10,982 species of birds and 14,553 species of Lepidoptera, denoted L-Bird ("Large Bird") and L-Butterfly. For aircraft, we assembled a list of 409 types of aircraft by hand (including aircraft in the FGVC-Aircraft [117] dataset, abbreviated FGVC). For dogs, we combine the 120 dog breeds in Stanford Dogs [88] with 395 other categories to obtain the 515-category L-Dog. We evaluate on two other fine-grained datasets in addition to FGVC and Stanford Dogs: CUB-200-2011 [182] and Birdsnap [20], for a total of four evaluation datasets. CUB and Birdsnap include 200 and 500 species of common birds, respectively, FGVC has 100 aircraft variants, and Stanford Dogs contains 120 breeds of dogs. In this section we focus our analysis on the categories in L-Bird, L-Butterfly, and L-Aircraft in addition to the categories in their evaluation datasets.

### 6.3.2 Images from the Web

We obtain imagery via Google image search results, using all returned images as images for a given category. For L-Bird and L-Butterfly, queries are for the scientific name of the category, and for L-Aircraft and L-Dog queries are simply for the category name (*e.g.* "Boeing 737-200" or "Pembroke Welsh Corgi").

[1] Google image search: `http://images.google.com`

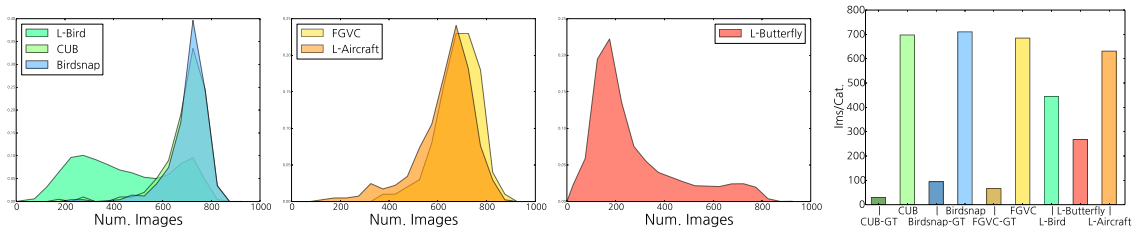



Figure 6.2: Distributions of the number of images per category available via image search for the categories in CUB, Birdsnap, and L-Bird (far left), FGVC and L-Aircraft (middle left), and L-Butterfly (middle right). At far right we aggregate and plot the average number of images per category in each dataset in addition to the training sets of each curated dataset we consider, denoted CUB-GT, Birdsnap-GT, and FGVC-GT.

**Quantifying the Data.**

How much fine-grained data is available? In Fig. 6.2 we plot distributions of the number of images retrieved for each category and report aggregates across each set of categories. We note several trends: Categories in existing datasets, which are typically common within their fine-grained domain, have more images per category than the long-tail of categories present in the larger L-Bird, L-Aircraft, or L-Butterfly, with the effect most pronounced in L-Bird and L-Butterfly. Further, domains of fine-grained categories have substantially different distributions, *i.e.* L-Bird and L-Aircraft have more images per category than L-Butterfly. This makes sense – fine-grained categories and domains of categories that are more common and have a larger enthusiast base will have more imagery since more photos are taken of them. We also note that results tend to be limited to roughly 800 images per category, even for the most common categories, which is likely a restriction placed on public search results.

Most striking is the large difference between the number of images available via web search and in existing fine-grained datasets: even Birdsnap, which has an average of 94.8 images per category, contains only 13% as many images as can be obtained with a simple image search. Though their labels are noisy, web searches unveil an order of magnitude more data which can be used to learn fine-grained categories.

In total, for all four datasets, we obtained 9.8 million images for 26,458 categories, requiring 151.8GB of disk space. All urls have been released at `https://github.`


Figure 6.3: Examples of cross-domain noise for birds, butterflies, airplanes, and dogs. Images are generally of related categories that are outside the domain of interest, *e.g.* a map of a bird's typical habitat or a t-shirt containing the silhouette of a dog.

`com/google/goldfinch/`.

**Noise.**

Though large amounts of imagery are freely available for fine-grained categories, focusing only on scale ignores a key issue: *noise*. We consider two types of label noise, which we call *cross-domain* noise and *cross-category* noise. We define cross-domain noise to be the portion of images that are not of any category in the same fine-grained domain, *i.e.* for birds, it is the fraction of images that do not contain a bird (examples in Fig. 6.3). In contrast, *cross-category* noise is the portion of images that have the wrong label within a fine-grained domain, *i.e.* an image of a bird with the wrong species label.

To quantify levels of cross-domain noise, we manually label a 1,000 image sample from each set of search results, with results in Fig. 6.4. Although levels of noise are not too high for any set of categories (max. 34.2% for L-Butterfly), we notice an interesting correlation: cross-domain noise decreases moderately as the number of images per category (Fig. 6.2) increases. We hypothesize that categories with many search results have a corresponding large pool of images to draw results from, and thus actual search results will tend to be higher-precision.

In contrast to cross-domain noise, cross-category noise is much harder to quantify, since doing so effectively requires ground truth fine-grained labels of query results. To

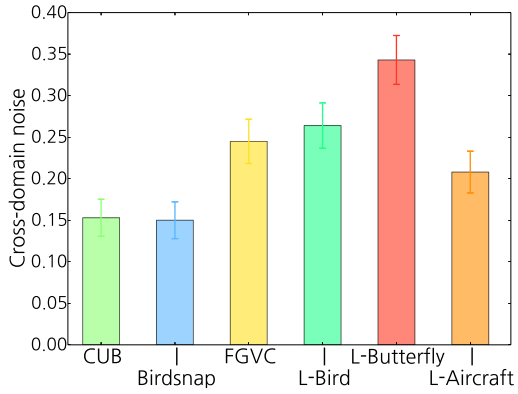



Figure 6.4: The cross-domain noise in search results for each domain.

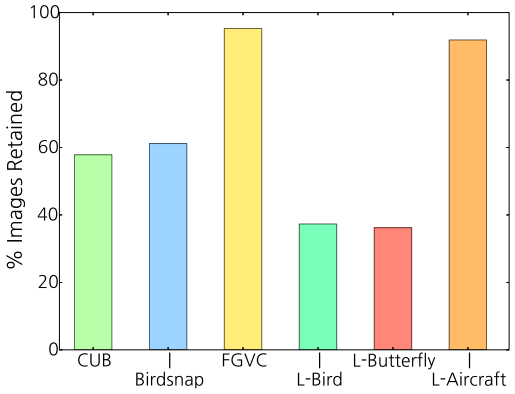



Figure 6.5: The percentage of images retained after filtering.

examine cross-category noise from at least one vantage point, we show the confusion matrix of given versus predicted labels on 30 categories in the CUB [182] test set and their web images in Fig. 6.6, left and right, which we generate via a classifier trained on the CUB training set, acting as a noisy proxy for ground truth labels. In these confusion matrices, cross-category noise is reflected as a strong off-diagonal pattern, while cross-domain noise would manifest as a diffuse pattern of noise, since images not of the same domain are an equally bad fit to all categories. Based on this interpretation, the web images show a moderate amount more cross-category noise than the clean CUB test set, though the general confusion pattern is similar.

We propose a simple, yet effective strategy to reduce the effects of cross-category noise: exclude images that appear in search results for more than one category. This approach, which we refer to as *filtering*, specifically targets images for which there is explicit ambiguity in the category label (examples in Fig. 6.7). As we demonstrate experimentally, filtering can improve results while reducing training time via the use of a more compact training set – we show the portion of images kept after filtering in Fig. 6.5. Agreeing with intuition, filtering removes more images when there are more categories. Anecdotally, we have also tried a few techniques to combat cross-domain noise, but initial experiments did not see any improvement in recognition so we do not expand upon them here. While reducing cross-domain noise should be beneficial, we believe that it is not as important as cross-category noise in fine-grained recognition

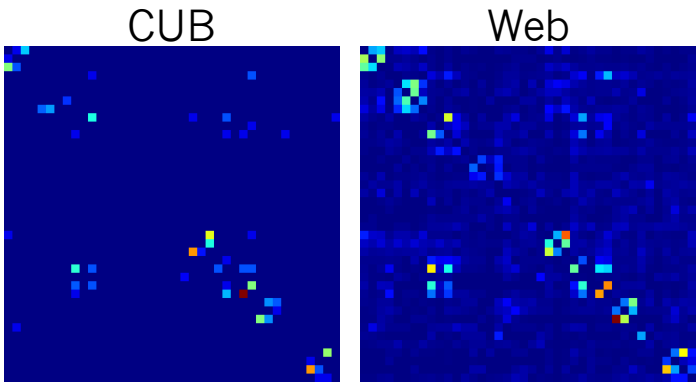



Figure 6.6: Confusion matrices of the predicted label (column) given the provided label (row) for 30 CUB categories on the CUB test set (left) and search results for CUB categories (right). For visualization purposes we remove the diagonal.

due to the absence of out-of-domain classes during testing.

## 6.4 Data via Active Learning

In this section we describe an active learning-based approach for collecting large quantities of fine-grained data. Active learning and other human-in-the-loop systems have previously been used to create datasets in a more cost-efficient way than manual annotation [200, 40, 156], and our goal is to compare this more traditional approach with simply using noisy data, particularly when considering the application of fine-grained recognition. In this paper, we apply active learning to the 120 dog breeds in the Stanford Dogs [88] dataset.

Our system for active learning begins by training a classifier on a seed set of input images and labels (*i.e.* the Stanford Dogs training set), then proceeds by iteratively picking a set of images to annotate, obtaining labels with human annotators, and re-training the classifier. We use a convolutional neural network [103, 168, 81] for the classifier, and now describe the key steps of sample selection and human annotation in more detail.


Figure 6.7: Examples of images removed via filtering and the categories whose results they appeared in. Some share similar names (left examples), while others share similar locations (right examples).

**Sample Selection.**

There are many possible criterion for sample selection [156]. We employ confidence-based sampling: For each category $c$, we select the $b\hat{P}(c)$ images with the top class scores $f_c(x)$ as determined by our current model, where $\hat{P}(c)$ is a desired prior distribution over classes, $b$ is a budget on the number of images to annotate, and $f_c(x)$ is the output of the classifier. The intuition is as follows: even when $f_c(x)$ is large, false positives still occur quite frequently – in Fig. 6.9 left, observe that the false positive rate is about 20% at the highest confidence range, which might have a large impact on the model. This contrasts with approaches that focus sampling in uncertain regions [106, 14, 128, 53]. We find that images sampled with uncertainty criteria are typically ambiguous and difficult or even impossible for both models *and* humans to annotate correctly, as demonstrated in Fig. 6.9 bottom row: unconfident samples are often heavily occluded, at unusual viewpoints, or of mixed, ambiguous breeds, making it unlikely that they can be annotated effectively. This strategy is similar to the "expected model change" sampling criteria [157], but done for each class independently.

**Human Annotation.**

**Interface** Designing an effective rater tool is of critical importance when getting non-experts to rate fine-grained categories. We seek to give the raters simple decisions and to provide them with as much information as possible to make the correct decision in a generic and scalable way. Fig. 6.8 shows our rater interface, which includes the following components to serve this purpose:

**Instructional positive images** inform the rater of within-class variation. These images are obtained from the seed dataset input to active learning. Many rater tools only provide this (*e.g.* [112]), which does not provide a clear class boundary concept on its own. We also provide links to Google Image Search and encourage raters to research the full space of examples of the class concept.

**Instructional negative images** help raters define the decision boundary between the right class and easily confused other classes. We show the top two most confused categories, determined by the active learning's current model. This aids in classification: in Fig. 6.8, if the rater studies the positive class "Bernese mountain dog", they may form a mental decision rule based on fur color pattern alone. However, when studying the negative, easily confused classes "Entlebucher" and "Appenzeller", the rater can refine the decision on more appropriate fine-grained distinctions – in this case, hair length is a key discriminative attribute.

**Batching questions by class** has the benefit of allowing raters to learn about and focus on one fine-grained category at a time. Batching questions may also allow raters to build a better mental model of the class via a human form of semi-supervised learning, although this phenomena is more difficult to isolate and measure.

**Golden questions for rater feedback and quality control.** We use the original supervised seed dataset to add a number of known correct and incorrect images in the batch to be rated, which we use to give short- and long-term feedback to raters. Short-term feedback comes in the form of a pop-up window informing the

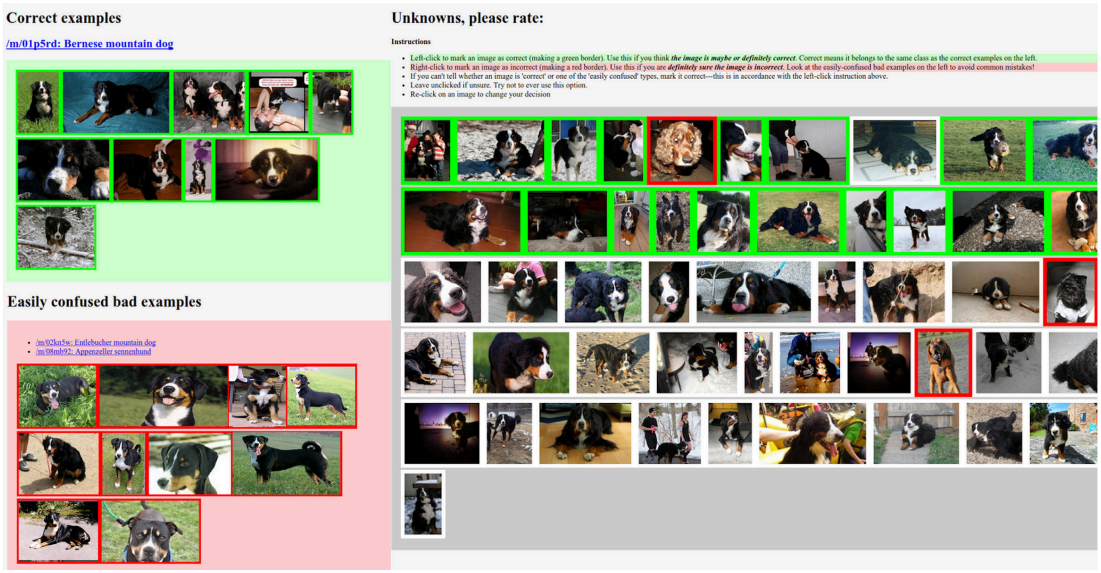



Figure 6.8: Our tool for binary annotation of fine-grained categories. Instructional positive images are provided in the upper left and negatives are provided in the lower left.

rater the moment they make an incorrect judgment, allowing them to update their mental model while working on the task. Long-term feedback summarizes a days' worth of rating to give the rater a summary of overall performance. Final category decisions are made via majority vote of three annotators.

**Rater Quality Improvements** To determine the impact of our annotation framework improvements for fine-grained categories, we performed a control experiment with a more standard crowdsourcing interface, which provides only a category name, description, and image search link. Annotation quality is determined on a set of difficult binary questions (images mistaken by a classifier on the Stanford Dogs test set). Using our interface, annotators were both more accurate and faster, with a 16.5% relative reduction in error (from 28.5% to 23.8%) and a 2.4× improvement in speed (4.1 to 1.68 seconds per image).

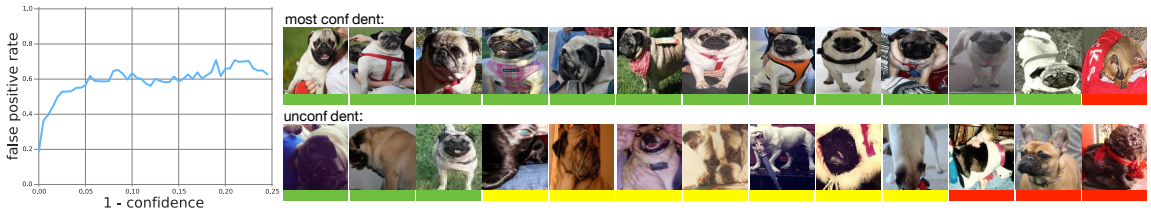



Figure 6.9: **Left:** Classifier confidence versus false positive rate on 100,000 randomly sampled from Flickr images (YFCC100M [170]) with dog detections. Even the most confident images have a 20% false positive rate. **Right:** Samples from Flickr. Rectangles below images denote correct (green), incorrect (red), or ambiguous (yellow). **Top row:** Samples with high confidence for class "Pug" from YFCC100M. **Bottom row:** Samples with low confidence score for class "Pug".

### Annotation Statistics and Examples

In Fig. 6.10 we show the distribution of images judged correct by human annotators after active learning selection of 1000 images per class for Stanford Dogs classes. The categories are sorted by the number of positive training examples collected in the first iteration of active learning. The 10 categories with the most positive training examples collected after both rounds of mining are: Pug, Golden Retriever, Boston Terrier, West Highland White Terrier, Labrador Retriever, Boxer, Maltese, German Shepherd, Pembroke Welsh Corgi, and Beagle. The 10 categories with the fewest positive training examples are: Kerry Blue Terrier, Komondor, Irish Water Spaniel, Curly Coated Retriever, Bouvier des Flandres, Clumber Spaniel, Bedlington Terrier, Afghan Hound, Affenpinscher, and Sealyham Terrier. These counts are influenced by the true counts of categories in the YFCC100M [170] dataset and our active learner's ability to find them.

In Fig. 6.11, we show positive training examples obtained from active learning for select categories, comparing examples obtained in iterations 1 and 2.

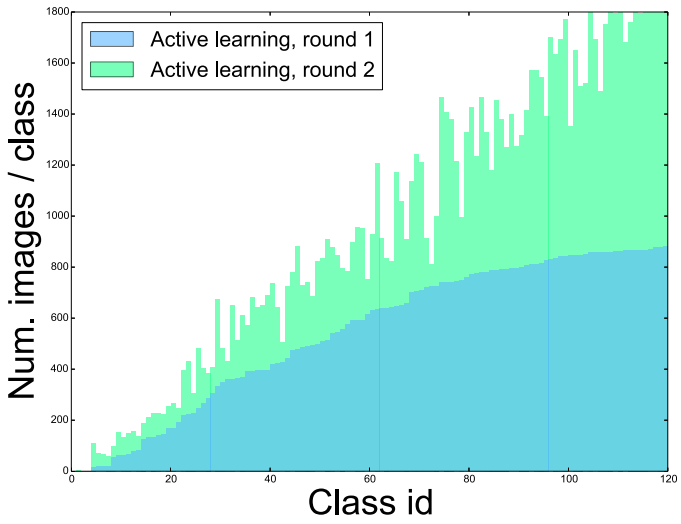



Figure 6.10: Counts of positive training examples obtained per category from active learning, for the Stanford Dogs dataset.

## 6.5 Experiments

### 6.5.1 Implementation Details

The base classifier we use in all noisy data experiments is the Inception-v3 convolutional neural network architecture [169], which is among the state of the art methods for generic object recognition [151, 167, 72]. Learning rate schedules are determined by performance on a holdout subset of the training data, which is 10% of the training data for control experiments training on ground truth datasets, or 1% when training on the larger noisy web data. Unless otherwise noted, all recognition results use as input a single crop in the center of the image.

Our active learning comparison uses the Yahoo Flickr Creative Commons 100M dataset [170] as its pool of unlabeled images, which we first pre-filter with a binary dog classifier and localizer [168], resulting in 1.71 million candidate dogs. We perform up to two rounds of active learning, with a sampling budget $B$ of $10\times$ the original dataset size per round[2]. For experiments on Stanford Dogs, we use the CNN of [81], which is pre-trained on a version of ILSVRC [151, 45] with dog data removed, since

[2]To be released.

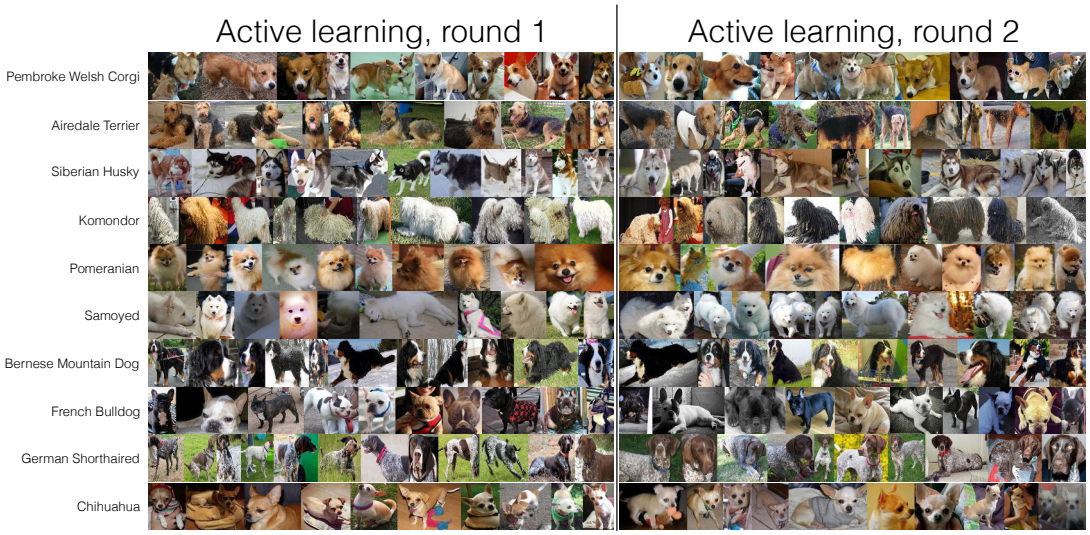



Figure 6.11: Positive training examples obtained from active learning, from the YFCC100M dataset, for select categories from Stanford Dogs.

Stanford Dogs is a subset of ILSVRC training data.

### 6.5.2 Removing Ground Truth from Web Images

One subtle point to be cautious about when using web images is the risk of inadvertently including images from ground truth test sets in the web training data. To deal with this concern, we performed an aggressive deduplication procedure with all ground truth test sets and their corresponding web images.

Our general approach follows Wang *et al.* [184], which is a state of the art method for learning a similarity metric between images. To scale [184] to the millions of images considered in this work, we binarize the output for an efficient hashing-based exact search. Hamming distance corresponds to dissimilarity: identical images have distance 0, images with different resolutions, aspect ratios, or slightly different crops tend to have distances of up to roughly 4 and 8, and more substantial variations, *e.g.* images of different views from the same photographer, or very different crops, roughly have distances up to 10, beyond which the vast majority of image pairs are actually distinct. Qualitative examples are provided in Fig. 6.12. We tuned our dissimilarity threshold for recall and manually verified it – the goal is to ensure that

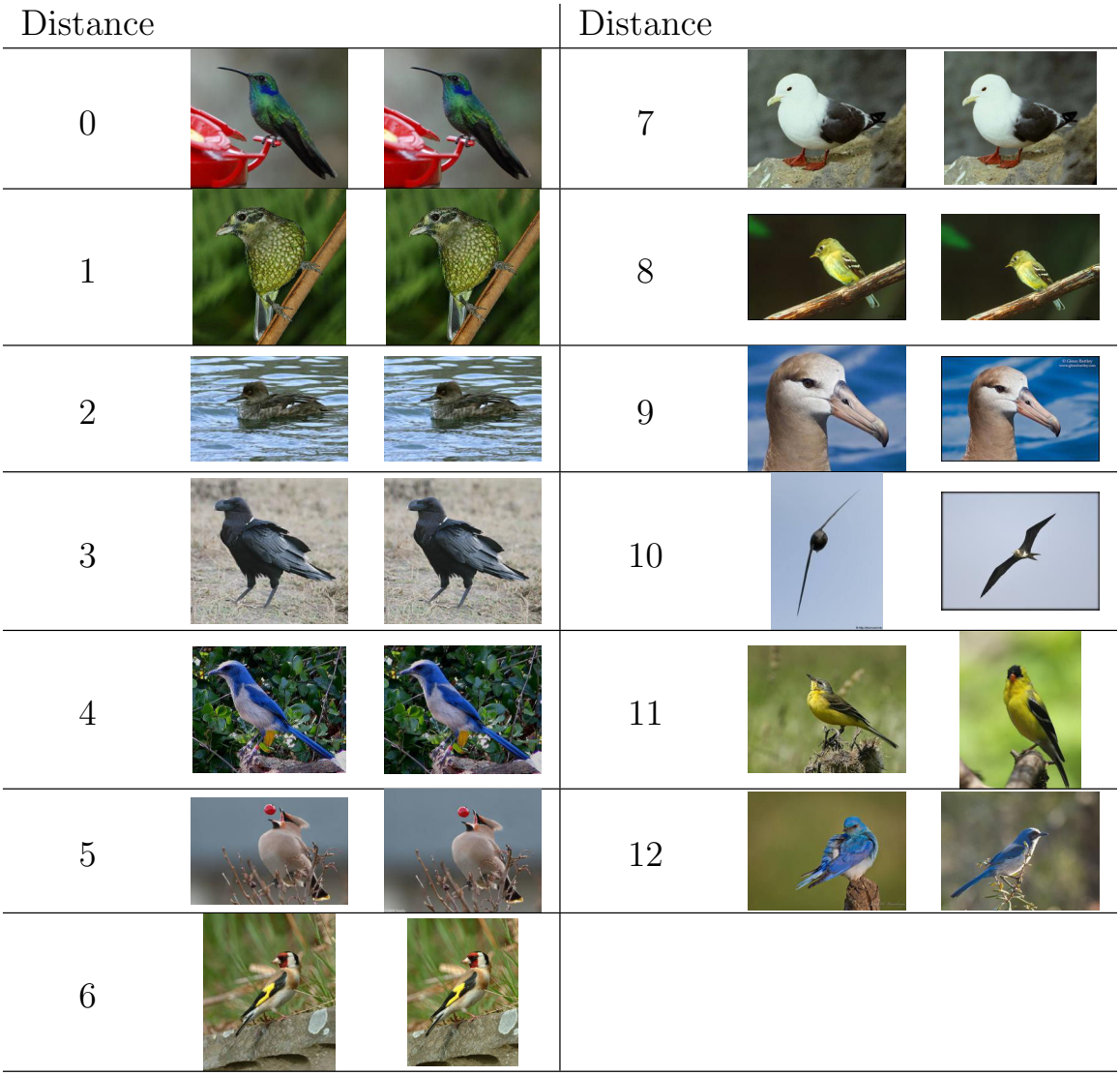



Figure 6.12: Example pairs of images and their distance according to our deduplication method. Distances 1-3 have slight pixel-level differences due to compression and the image pair at distance 4 have different scales. At distances 5 and 6 the images are of different crops, with distance 6 additionally exhibiting slight lighting differences. The images at distance 7 have slightly different scales and compression, at distance 8 there are cropping and lighting differences, and distance 9 features different crops and additional text in the corner of one photo. At distance 10 and higher we have image pairs which have high-level visual similarities but are distinct.

| Training Data | Acc. | Dataset | Training Data | Acc. | Dataset |
|---|---|---|---|---|---|
| CUB-GT | 84.4 | CUB [182] | FGVC-GT | 88.1 | FGVC [117] |
| Web (raw) | 87.7 | | Web (raw) | 90.7 | |
| Web (filtered) | 89.0 | | Web (filtered) | 91.1 | |
| L-Bird | 91.9 | | L-Aircraft | 90.9 | |
| L-Bird(MC) | 92.3 | | L-Aircraft(MC) | 93.4 | |
| L-Bird+CUB-GT | 92.2 | | L-Aircraft+FGVC-GT | 94.5 | |
| L-Bird+CUB-GT(MC) | 92.8 | | L-Aircraft+FGVC-GT(MC) | 95.9 | |
| Birdsnap-GT | 78.2 | Birdsnap [20] | Stanford-GT | 80.6 | Stanford Dogs [88] |
| Web (raw) | 76.1 | | Web (raw) | 78.5 | |
| Web (filtered) | 78.2 | | Web (filtered) | 78.4 | |
| L-Bird | 82.8 | | L-Dog | 78.4 | |
| L-Bird(MC) | 85.4 | | L-Dog(MC) | 80.8 | |
| L-Bird+Birdsnap-GT | 83.9 | | L-Dog+Stanford-GT | 84.0 | |
| L-Bird+Birdsnap-GT(MC) | 85.4 | | L-Dog+Stanford-GT(MC) | 85.9 | |

Table 6.1: Comparison of data source used during training with recognition performance, given in terms of Top-1 accuracy. "CUB-GT" indicates training only on the ground truth CUB training set, "Web (raw)" trains on all search results for CUB categories, and "Web (filtered)" applies filtering between categories within a domain (birds). L-Bird denotes training first on L-Bird, then fine-tuning on the subset of categories under evaluation (*i.e.* the filtered web images), and L-Bird+CUB-GT indicates training on L-Bird, then fine-tuning on Web (filtered), and finally fine-tuning again on CUB-GT. Similar notation is used for the other datasets. "(MC)" indicates using multiple crops at test time (see text for details). We note that only the rows with "-GT" make use of the ground truth training set; all other rows rely solely on noisy web imagery.

images that have even a moderate degree of similarity to test images did not appear in our training set. For example, of a sample of 183 image pairs at distance 16 in the large-scale bird experiments, zero were judged by a human to be too similar, and we used a still more conservative threshold of 18. In the case of L-Bird, 2,996 images were removed as being too similar to an image in either the CUB or Birdsnap test set.

### 6.5.3 Main Results

We present our main recognition results in Tab. 6.1, where we compare performance when the training set consists of either the ground truth training set, raw web images of the categories in the corresponding evaluation dataset, web images after applying

our filtering strategy, all web images of a particular domain, or all images including even the ground truth training set.

On CUB-200-2011 [182], the smallest dataset we consider, even using raw search results as training data results in a better model than the annotated training set, with filtering further improving results by 1.3%. For Birdsnap [20], the largest of the ground truth datasets we evaluate on, raw data mildly underperforms using the ground truth training set, though filtering improves results to be on par. On both CUB and Birdsnap, training first on the very large set of categories in L-Bird results in dramatic improvements, improving performance on CUB further by 2.9% and on Birdsnap by 4.6%. This is an important point: even if the end task consists of classifying only a small number of categories, training with more fine-grained categories yields significantly more effective networks. This can also be thought of as a form of transfer learning within the same fine-grained domain, allowing features learned on a related task to be useful for the final classification problem. When permitted access to the annotated ground truth training sets for additional fine-tuning and domain transfer, results increase by another 0.3% on CUB and 1.1% on Birdsnap.

For the aircraft categories in FGVC, results are largely similar but weaker in magnitude. Training on raw web data results in a significant gain of 2.6% compared to using the curated training set, and filtering, which did not affect the size of the training set much (Fig. 6.5), changes results only slightly in a positive direction. Counterintuitively, pre-training on a larger set of aircraft does not improve results on FGVC. Our hypothesis for the difference between birds and aircraft in this regard is this: since there are many more species of birds in L-Bird than there are aircraft in L-Aircraft (10,982 vs 409), not only is the training size of L-Bird larger, but each training example provides stronger information because it distinguishes between a larger set of mutually-exclusive categories. Nonetheless, when access to the curated training set is available for fine-tuning, performance dramatically increases to 94.5%. On Stanford Dogs we see results similar to FGVC, though for dogs we happen to see a mild loss when comparing to the ground truth training set, not much difference with filtering or using L-Dog, and a large boost from adding in the ground truth training set.

| Method | Training Annotations | Acc. |
|---|---|---|
| Alignments [61] | GT | 53.6 |
| PDD [161] | GT+BB+Parts | 60.6 |
| PB R-CNN [203] | GT+BB+Parts | 73.9 |
| Weak Sup. [206] | GT | 75.0 |
| PN-DCN [25] | GT+BB+Parts | 75.7 |
| Two-Level [190] | GT | 77.9 |
| Consensus [159] | GT+BB+Parts | 78.3 |
| NAC [160] | GT | 81.0 |
| FG-Without [93] | GT+BB | 82.0 |
| STN [83] | GT | 84.1 |
| Bilinear [113] | GT | 84.1 |
| Augmenting [194] | GT+BB+Parts+Web | 84.6 |
| **Noisy Data+CNN** [169] | **Web** | **92.3** |

Table 6.2: Comparison with prior work on CUB-200-2011 [182]. We only include methods which use no annotations at test time. Here "GT" refers to using Ground Truth category labels in the training set of CUB, "BBox" indicates using bounding boxes, and "Parts" additionally uses part annotations.

An additional factor that can influence performance of web models is domain shift – if images in the ground truth test set have very different visual properties compared to web images, performance will naturally differ. Similarly, if category names or definitions within a dataset are even mildly off, web-based methods will be at a disadvantage without access to the ground truth training set. Adding the ground truth training data fixes this domain shift, making web-trained models quickly recover, with a particularly large gain if the network has already learned a good representation, matching the pattern of results for Stanford Dogs.

**Limits of Web-Trained Models.**

To push our models to their limits, we additionally evaluate using 144 image crops at test time, averaging predictions across each crop, denoted "(MC)" in Tab. 6.1. This brings results up to 92.3%/92.8% on CUB (without/with CUB training data), 85.4%/85.4% on Birdsnap, 93.4%/95.9% on FGVC, and 80.8%/85.9% on Stanford Dogs. We note that this is close to human expert performance on CUB, which is estimated to be between 93% [26] and 95.6% [174].

**Comparison with Prior Work.**

We compare our results to prior work on CUB, the most competitive fine-grained dataset, in Tab. 6.5.3. While even our baseline model using only ground truth data from Tab. 6.1 was at state of the art levels, by forgoing the CUB training set and only training using noisy data from the web, our models greatly outperform all prior work. On FGVC, which is more recent and fewer works have evaluated on, the best prior performing method we are aware of is the Bilinear CNN model of Lin *et al.* [113], which has accuracy 84.1% (ours is 93.4% without FGVC training data, 95.9% with), and on Birdsnap, which is even more recent, the best performing method we are aware of that uses no extra annotations during test time is the original 66.6% by Berg *et al.* [20] (ours is 85.4%). On Stanford Dogs, the most competitive related work is [155], which uses an attention-based recurrent neural network to achieve 76.8% (ours is 80.8% without ground truth training data, 85.9% with).

We identify two key reasons for these large improvements: The first is the use of a strong generic classifier [169]. A number of prior works have identified the importance of having well-trained CNNs as components in their systems for fine-grained recognition [113, 83, 93, 203, 25], which our work provides strong evidence for. On all four evaluation datasets, our CNN of choice [169], trained on the ground truth training set alone and without any architectural modifications, performs at levels at or above the previous state-of-the-art. The second reason for improvement is the large utility of noisy web data for fine-grained recognition, which is the focus of this work.

We finally remind the reader that our work focuses on the application-level problem of recognizing a given set of fine-grained categories, which might not come with their own expert-annotated training images. The use of existing test sets serves to provide an accurate measure of performance and put our work in a larger context, but results may not be strictly comparable with prior work that operates within a single given dataset.

| Training Procedure | Acc. |
|---|---|
| Stanford-GT (scratch) | 58.4 |
| A.L., one round (scratch) | 65.8 |
| A.L., two rounds (scratch) | 74.0 |
| Stanford-GT (ft) | 80.6 |
| A.L., one round (ft) | 81.6 |
| A.L., one round (ft, subsample) | 78.8 |
| A.L., two rounds (ft) | 82.1 |
| Web (filtered) | 78.4 |
| Web (filtered) + Stanford-GT | 82.6 |

Table 6.3: Active learning-based results on Stanford Dogs [88], presented in terms of top-1 accuracy. Methods with "(scratch)" indicate training from scratch and "(ft)" indicates fine-tuning from a network pre-trained on ILSVRC, with web models also fine-tuned. "subsample" refers to downsampling the active learning data to be the same size as the filtered web images. Note that Stanford-GT is a subset of active learning data, which is denoted "A.L.".

**Comparison with Active Learning.**

We compare using noisy web data with a more traditional active learning-based approach (Sec. 6.4) under several different settings in Tab. 6.3. We first verify the efficacy of active learning itself: when training the network from scratch (*i.e.* no fine-tuning), active learning improves performance by up to 15.6%, and when fine-tuning, results still improve by 1.5%.

How does active learning compare to using web data? Purely using filtered web data compares favorably to non-fine-tuned active learning methods (4.4% better), though lags behind the fine-tuned models somewhat. To better compare the active learning and noisy web data, we factor out the difference in scale by performing an experiment with subsampled active learning data, setting it to be the same size as the filtered web data. Surprisingly, performance is very similar, with only a 0.4% advantage for the cleaner, annotated active learning data, highlighting the effectiveness of noisy web data despite the lack of manual annotation. If we furthermore augment the filtered web images with the Stanford Dogs training set, which the active learning method notably used both as training data and its seed set of images, performance improves to even be slightly better than the manually-annotated active learning data (0.5% improvement).

These experiments indicate that, while more traditional active learning-based approaches towards expanding datasets are effective ways to improve recognition performance given a suitable budget, simply using noisy images retrieved from the web can be nearly as good, if not better. As web images require no manual annotation and are openly available, we believe this is strong evidence for their use in solving fine-grained recognition.

**Very Large-Scale Fine-Grained Recognition.**

A key advantage of using noisy data is the ability to scale to large numbers of fine-grained classes. However, this poses a challenge for evaluation – it is infeasible to manually annotate images with one of the 10,982 categories in L-Bird, 14,553 categories in L-Butterfly, and would even be very time-consuming to annotate images with the 409 categories in L-Aircraft. Therefore, we turn to an approximate evaluation, establishing a rough estimate on true performance. Specifically, we query Flickr for up to 25 images of each category, keeping only those images whose title strictly contains the name of each category, and aggressively deduplicate these images with our training set in order to ensure a fair evaluation. Although this is not a perfect evaluation set, and is thus an area where annotation of fine-grained datasets is particularly valuable [174], we find that it is remarkably clean on the surface: based on a 1,000-image estimate, we measure the cross-domain noise of L-Bird at only 1%, L-Butterfly at 2.3%, and L-Aircraft at 4.5%. An independent evaluation [174] further measures all sources of noise combined to be only 16% when searching for bird species. In total, this yields 42,115 testing images for L-Bird, 42,046 for L-Butterfly, and 3,131 for L-Aircraft.

Given the difficulty and noise, performance is surprisingly high: On L-Bird top-1 accuracy is 73.1%/75.8% (1/144 crops), for L-Butterfly it is 65.9%/68.1%, and for L-Aircraft it is 72.7%/77.5%. Corresponding mAP numbers, which are better suited for handling class imbalance, are 61.9, 54.8, and 70.5, reported for the single crop setting. We show qualitative results in Fig. 6.13. These categories span multiple continents in space (birds, butterflies) and decades in time (aircraft), demonstrating the breadth of categories in the world that can be recognized using only public sources of noisy

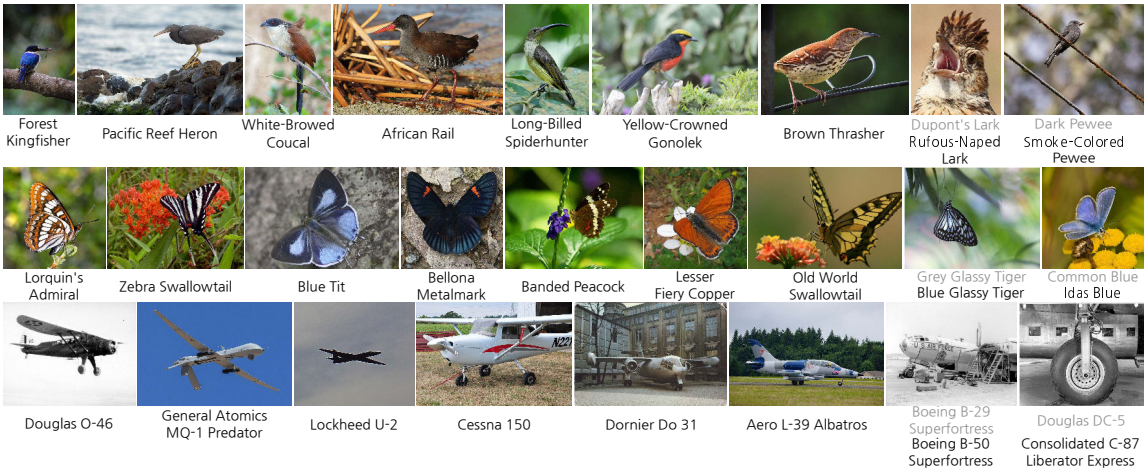



Figure 6.13: Classification results on very large-scale fine-grained recognition. From top to bottom, depicted are examples of categories in L-Bird, L-Butterfly, and L-Aircraft, along with their category name. The first examples in each row are correctly predicted by our models, while the last two examples in each row are errors, with our prediction in grey and correct category (according to Flickr metadata) printed below.

fine-grained data. To the best of our knowledge, these results represent the largest number of fine-grained categories distinguished by any single system to date.

### How Much Data is Really Necessary?

In order to better understand the utility of noisy web data for fine-grained recognition, we perform a control experiment on the web data for CUB. Using the filtered web images as a base, we train models using progressively larger subsets of the results as training data, taking the top ranked images across categories for each experiment. Performance versus the amount of training data is shown in Fig. 6.14. Surprisingly, relatively few web images are required to do as well as training on the CUB training set, and adding more noisy web images always helps, even when at the limit of search results. Based on this analysis, we estimate that one noisy web image for CUB categories is "worth" 0.507 ground truth training images [172].

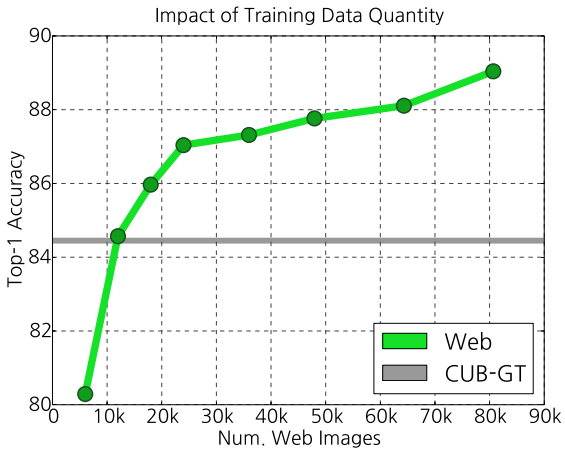



Figure 6.14: Number of web images used for training vs. performance on CUB-200-2011 [182]. We vary the amount of web training data in multiples of the CUB training set size (5,994 images). Also shown is performance when training on the ground truth CUB training set (CUB-GT).

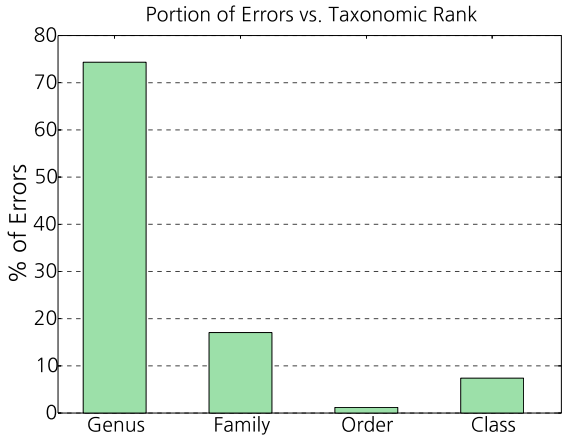



Figure 6.15: The errors on L-Bird that fall in each taxonomic rank, represented as a portion of all errors made. For each error made, we calculate the taxonomic rank of the least common ancestor of the predicted and test category.

**Error Analysis.**

**Taxonomic Distribution** Given the high performance of these models, what room is left for improvement? In Fig. 6.15 we show the taxonomic distribution of the remaining errors on L-Bird. The vast majority of errors (74.3%) are made between very similar classes at the genus level, indicating that most of the remaining errors are indeed between extremely similar categories, and only very few errors (7.4%) are made between dissimilar classes, whose least common ancestor is the "Aves" (*i.e.* Bird) taxonomic class. This suggests that most errors still made by the models are fairly reasonable, corroborating the qualitative results of Fig. 6.13.

**Qualitative** Here we highlight one type of error that our image search model made on CUB [181] – finding errors in the test set. We show an example in Fig. 6.16, where the true species for each image is actually a bird species not in the 200 CUB bird species. This highlights one potential advantage of our approach: by relying

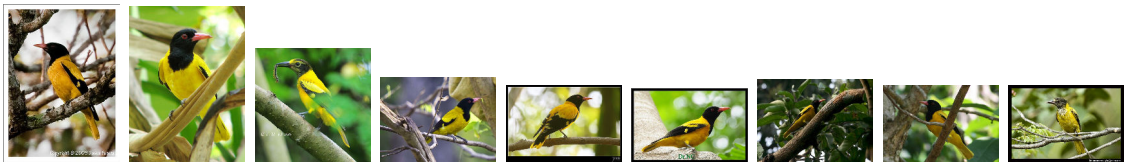


Figure 6.16: Examples of mistakes made by a web-trained model on the CUB-200-2011 [181] test set, whose ground truth label is "Hooded Oriole", but which are actually of another species not in CUB, "Black-Hooded Oriole."

on category names, web training data is tied more strongly to the semantic meaning of a category instead of simply a 1-of-$K$ label. This also provides evidence for the "domain shift" hypothesis when fine-tuning on ground truth datasets, as irregularities like this can be learned, resulting in higher performance on the benchmark dataset under consideration.

### 6.5.4 Network Visualization

In order to examine the impact of web-trained models of fine-grained recognition from another vantage point, here we present one visualization of network internals. Specifically, in Fig. 6.17 we visualize gradients with respect to the square of the norm of the last convolutional layer in the network, backpropagated into the input image, and visualized as a function of training data. This provides some indication of the importance of each pixel with respect to the overall network activation. Though these examples are only qualitative, we observe that the gradients for the network trained on L-Bird are generally more focused on the bird when compared to gradients for the network trained on CUB, indicating that the network has learned a better representation of which parts of an image are discriminative.

## 6.6 Conclusions

In this work we have demonstrated the utility of noisy data toward solving the problem of fine-grained recognition. We found that the combination of a generic classification model and web data, filtered with a simple strategy, was surprisingly effective at

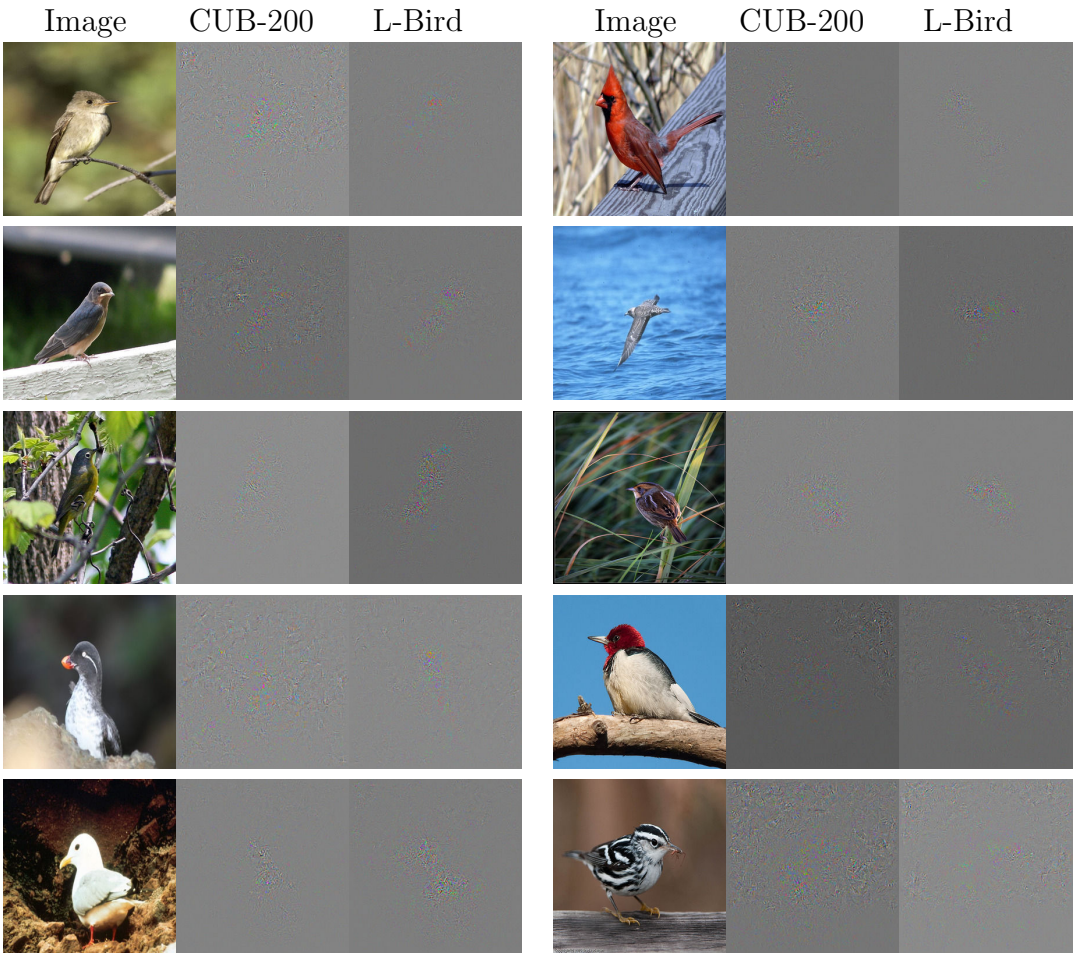



Figure 6.17: Gradients with respect to the squared norm of the last convolutional layer on ten random CUB test set images. Each row contains, in order, an input image, gradients for a model trained on the CUB-200 [181] training set, and gradients for a model trained on the larger L-Bird. Gradients have been scaled to fit in [0,255]. Best viewed in high resolution on a monitor.

discriminating fine-grained categories. This approach performs favorably when compared to a more traditional active learning method for expanding datasets, but is even more scalable, which we demonstrated experimentally on up to 14,553 fine-grained categories. One potential limitation of the approach is the availability of imagery for categories either not found or not described in the public domain, for which an alternative method such as active learning may be better suited. Another limitation

is the current focus on classification, which may be problematic if applications arise where multiple objects are present or localization is otherwise required. Nonetheless, with these insights on the unreasonable effectiveness of noisy data, we are optimistic for applications of fine-grained recognition in the near future.

## 6.7 Acknowledgments

We thank Gal Chechik, Chuck Rosenberg, Zhen Li, Timnit Gebru, Vignesh Ramanathan, Oliver Groth, and the anonymous reviewers for valuable feedback.

# Chapter 7

# Generating Descriptive Image Paragraphs

Recent work on image captioning has made it possible to generate novel sentences describing images in natural language, but compressing an image into a single sentence can only describe visual content in coarse detail. Dense image captioning works on a finer-grained level of detail by detecting many image regions and generating a descriptive phrase for each, but it is unable to produce a coherent story for an image. In this paper we overcome these limitations by generating entire paragraphs for describing images, which can tell detailed, unified stories. We develop a model that decomposes both images and paragraphs into their constituent parts, detecting semantic regions in images and using a hierarchical recurrent neural network to split paragraphs into sentences. Linguistic analysis confirms the complexity of the paragraph generation task, and controlled experiments on a new dataset of image and paragraph pairs demonstrate the effectiveness of our hierarchical model.

This work was done jointly with Justin Johnson, Ranjay Krishna, and Fei-Fei Li, and is currently in submission for publication.

## 7.1 Introduction

Vision is a primary sensory input for human perception, and language is our most powerful tool for communicating with the world. Building systems that can simultaneously understand visual stimuli and describe it in natural language is therefore a core problem in AI. With the advent of large datasets pairing images with natural language descriptions [112] it has become possible to generate novel sentences describing images [37, 50, 87, 120, 178]. This success is encouraging, but these methods can only describe images at a coarse level of detail since they are limited to single high-level sentences.

The recently proposed task of *dense image captioning* [85] overcomes this limitation by detecting many regions of interest in images and generating a descriptive phrase for each. Though dense captioning can describe images in fine-grained detail, it is unable to produce a coherent story for an image. In this paper we address the shortcomings of both traditional image captioning and dense image captioning by introducing the problem of generating entire paragraphs that richly describe images. Like traditional captioning, paragraphs give a coherent natural language description for images, but like dense captioning they can do so in fine-grained detail.

Generating descriptive paragraphs for images is challenging, requiring both fine-grained image understanding and long-term language reasoning. To overcome these challenges we propose a model that decomposes images and paragraphs into their constituent parts. We break images into semantically meaningful parts by detecting objects and other regions of interest, and we model language with a hierarchical recurrent neural network that decomposes paragraphs into sentences. We transfer visual and linguistic knowledge from large-scale region captioning [96] by initializing both visual and linguistic components of our model from a pre-trained dense captioning model.

To validate our method, we collect a new dataset of image and paragraph pairs, complementing the whole-image and region-level annotations of MS-COCO [112] and Visual Genome [96]. Simple linguistic analysis of our paragraph dataset demonstrates

the complexity of the paragraph generation task compared to single-sentence captioning. Comparisons with several baseline methods showcase the benefits of hierarchical modeling for generating descriptive paragraphs.

## 7.2 Related Work

**Image Captioning** Building connections between visual and textual data has been a longstanding goal in computer vision. One line of work treats the problem as a ranking task, using images to retrieve relevant captions from a database and vice-versa [55, 76, 87]. Due to the compositional nature of language, it is unlikely that any database will contain all possible image captions; therefore another line of work attempts to generate novel captions to describe images. Early work uses handwritten templates to generate language [99] while more recent methods train recurrent neural network language models conditioned on image features [37, 50, 87, 120, 178] and sample from them to generate text. Similar methods have also been applied to generate captions for videos [50, 199, 201].

A handful of approaches to image captioning reason not only about whole images but also image regions. Xu *et al.* [193] generate captions using a recurrent network with attention, so that for each generated word the model produces a distribution over image regions. Rather than using semantically meaningful regions, they divide the image into a coarse grid and attend to grid cells. Karpathy and Fei-Fei [87] use a ranking loss to align image regions with sentence fragments, but neither generate text with this model nor operate on descriptions longer than short image captions. Johnson *et al.* [85] perform dense captioning, with a model jointly detecting and describing regions of interest; however these descriptions are independent and do not form a single coherent description.

**Hierarchical Recurrent Networks** In order to generate a paragraph description, a model must reason about long-term linguistic structures spanning multiple sentences. Due to vanishing gradients, recurrent neural networks trained with stochastic gradient descent often struggle to learn long-term dependencies. Alternative recurrent

architectures such as long-short term memory (LSTM) [75] help alleviate this problem through a gating mechanism that improves gradient flow. Another solution is a *hierarchical* recurrent network, where the architecture is designed such that different parts of the model operate on different time scales.

Early pioneering work applied hierarchical recurrent networks to simple algorithmic problems [52]. The recently proposed Clockwork RNN [90] uses a related technique for audio signal generation, spoken word classification, and handwriting recognition; a similar hierarchical architecture was also used in [32] for speech recognition. In these approaches, each recurrent unit is updated on a fixed schedule: some units are updated on every timestep, while other units might be updated every other or every fourth timestep. This type of hierarchy helps reduce the vanishing gradient problem, but the hierarchy of the model does not directly reflect the hierarchy of the output sequence.

More related to our work are hierarchical architectures that mirror the hierarchy of language. Li *et al.* [108] use a hierarchical autoencoder that decomposes paragraphs into sentences, and Lin *et al.* [111] perform hierarchical language modeling where different recurrent units are used to model sentences and words. Our work is most similar to the recent work of Yu *et al.* [201], who generate multi-sentence descriptions for videos using a slightly different hierarchical model that separates sentences and words. Besides model-level differences, they focus on generating sentences temporally tied to different segments of video, while we aim to provide rich descriptions for individual images.

## 7.3 Images and Paragraphs

Is describing images with paragraphs actually different from sentence-level captioning? To investigate, we collected paragraph annotations for a subset of 19,551 MS COCO [112] and Visual Genome [96] images on Amazon Mechanical Turk, using US workers with at least a 98% HIT acceptance rate and 5,000 accepted HITs. We provide an example in Fig. 7.1, along with the five sentence captions for the image from MS COCO. Though it is clear the the paragraph is longer and more descriptive than

***Sentence Captions***

- A girl is eating donuts with a boy in a restaurant
- A boy and girl sitting at a table with doughnuts.
- Two kids sitting a coffee shop eating some frosted donuts
- Two children sitting at a table eating donuts.
- Two children eat doughnuts at a restaurant table.

***Paragraph***

Two children are sitting at a table in a restaurant. The children are one little girl and one little boy. The little girl is eating a pink frosted donut with white icing lines on top of it. The girl has blonde hair and is wearing a green jacket with a black long sleeve shirt underneath. The little boy is wearing a black zip up jacket and is holding his finger to his lip but is not eating. A metal napkin dispenser is in between them at the table. The wall next to them is white brick. Two adults are on the other side of the short white brick wall. The room has white circular lights on the ceiling and a large window in the front of the restaurant. It is daylight outside.

Figure 7.1: Paragraphs are longer, more informative, and more linguistically complex than sentence-level captions. Here we show an image with its sentence-level captions from MS COCO [112] and the paragraph used in this work.

| | Sentences COCO [112] | Paragraphs Ours |
|---|---|---|
| Description Length | 11.30 | 67.50 |
| Sentence Length | 11.30 | 11.91 |
| Diversity | 19.01 | 70.49 |
| Nouns | 33.45% | 25.81% |
| Adjectives | 27.23% | 27.64% |
| Verbs | 10.72% | 15.21% |
| Pronouns | 1.23% | 2.45% |

Table 7.1: Statistics of paragraph descriptions, compared with sentence-level captions used in prior work. Description and sentence lengths are represented by the number of tokens present, diversity is the inverse of the average CIDEr score between sentences of the same image, and part of speech distributions are aggregated from Penn Treebank [121] part of speech tags.

any one sentence, we note further that a single paragraph can be more detailed than *all five* sentence captions, even when combined. This occurs because of redundancy in sentence-level captions – while each caption might use slightly different words to describe the image, since all sentence captions have the goal of describing the image as a whole, they are fundamentally limited in diversity.

We quantify this and other statistics of language in Tab. 7.1. Analyzing length, we find that each paragraph is roughly six times as long as the average sentence caption, with sentences in the paragraphs of comparable length as sentence captions in isolation. To examine the issue of sentence diversity, we compute the average CIDEr [177] similarity between COCO sentences for each image and between the individual sentences in each collected paragraph, with the final diversity score defined as 100 minus the average CIDEr score. By this metric, sentences within paragraphs are substantially more diverse than sentence captions, with a diversity score of 70.49 compared to 19.01. This quantifies the fact that sentences in paragraphs provide distinct information about images.

We also performed a simple linguistic analysis on COCO sentences and our collected paragraphs, annotating each word with a part of speech tag from Penn Treebank using Stanford CoreNLP [119] and aggregating parts of speech into higher-level categories. A few common parts of speech are given in Tab. 7.1. As a proportion, paragraphs have somewhat more verbs and pronouns, a comparable frequency of adjectives, and somewhat fewer nouns. Given the nature of paragraphs, this makes sense – longer descriptions go beyond the presence of a few salient objects and include information about their properties and relationships. We note but do not quantify that paragraphs also exhibit higher frequencies of more complex linguistic phenomena, *e.g.* coreference occurring in Fig. 7.1, wherein sentences refer to either "two children", "one little girl and one little boy", "the girl", or "the boy."

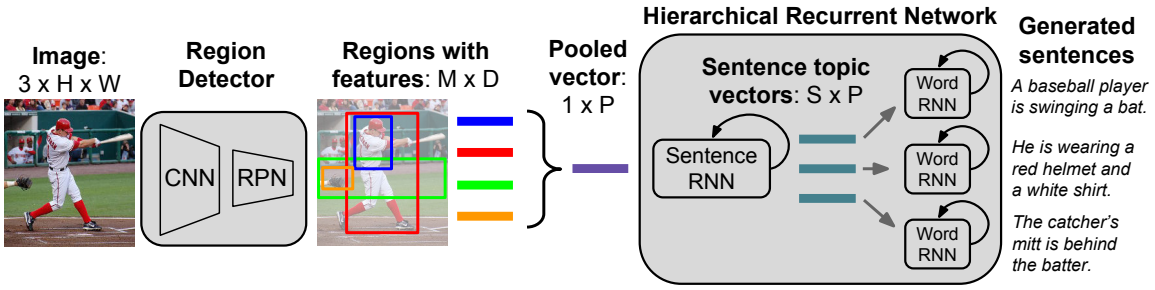



Figure 7.2: Overview of our model. Given an image (left), a region detector (comprising a convolutional network and a region proposal network) detects regions of interest and produces features for each. Region features are pooled to give a compact image representation, and passed to a hierarchical recurrent neural network language model comprising a sentence RNN and a word RNN. The sentence RNN generates sentence topic vectors which are consumed by the word RNN to generate sentences.

## 7.4 Method

**Overview.** Our model takes an image as input and generates a natural-language paragraph describing it, and is designed to take advantage of the compositional structure of both images and paragraphs. Fig. 7.2 provides a model overview. We decompose the input image by detecting objects and other regions of interest, then pool features across these regions to produce a representation richly expressing the image semantics. This feature vector is given to a hierarchical recurrent neural network composed of a sentence RNN and a word RNN. The sentence RNN receives the image features, decides how many sentences will comprise the generated paragraph, and produces an input topic vector for each sentence. Given this topic vector, the word RNN generates the words of a single sentence. We also show how our model can take advantage of knowledge transfer by initializing the region detector and pre-training the RNN from a model trained for dense image captioning [85].

### 7.4.1 Region Detector

The region detector receives an input image of size $3 \times H \times W$, detects regions of interest, and produces a feature vector of dimension $D = 4096$ for each region. Our region detector follows the design of [148, 85], and we provide a summary here for

completeness: The image is resized so that its longest edge is 720 pixels, and is then passed through a convolutional network initialized from the 16-layer VGG network [162]. The resulting feature map is processed by a region proposal network [148], which regresses from a set of anchors to propose regions of interest. These regions of interest are projected onto the convolutional feature map, and the corresponding region of the feature map is reshaped to a fixed size using bilinear interpolation as in [85] and processed by two fully-connected layers to give a vector of dimension $D$ for each region.

Given a dataset of images and ground-truth regions of interest, the region detector can be trained in an end-to-end fashion as in [148] for object detection and [85] for dense captioning. Since paragraph descriptions do not have annotated groundings to regions of interest, we use a region detector trained for dense image captioning on the Visual Genome dataset [96], using the publicly available implementation of [85]. We use a region detector trained for dense captioning rather object detection since an ideal paragraph describes not only objects, but also scenery and relationships between objects, which are captured by a dense captioning task. This produces $M = 50$ detected regions.

### 7.4.2 Region Pooling

The region detector produces a set of vectors $v_1, \ldots, v_M \in \mathbb{R}^D$, each describing a different region in the input image. We wish to aggregate these vectors into a single pooled vector $v_p \in \mathbb{R}^P$ that compactly describes the content of the image. To this end, we learn a projection matrix $W_{pool} \in \mathbb{R}^{P \times D}$ and bias $b_{pool} \in \mathbb{R}^P$; the pooled vector $v_p$ is computed by projecting each region vector using $W_{pool}$ and taking an elementwise maximum, so that $v_p = \max_{i=1}^{M}(W_{pool} v_i + b_{pool})$. Alternative approaches for representing collections of regions, such as spatial attention [193], may also be possible, and are complementary to the hierarchical model presented in this paper.

### 7.4.3 Hierarchical Recurrent Network

We input the pooled region vector $v_p \in \mathbb{R}^P$ to a hierarchical neural language model composed of two modules: a *sentence RNN* and a *word RNN*. The sentence RNN is responsible for deciding the number of sentences $S$ that should be in the generated paragraph and for producing a $P$-dimensional *topic vector* for each of these sentences. Given a topic vector for a sentence, the word RNN generates the words of that sentence. We adopt the standard LSTM architecture [75] for both the word RNN and the sentence RNN.

As an alternative to this hierarchical approach, one could instead use a non-hierarchical recurrent language model to directly generate the words of a paragraph, treating the end-of-sentence token as another word in the vocabulary. Our hierarchical model is advantageous because it reduces the length of time over which the recurrent networks must reason. Our paragraphs contain an average of 67.5 words, so a non-hierarchical approach must reason over dozens of time steps. However, since our paragraphs contain an average of 5.7 sentences, each with an average of 11.9 words, both the paragraph and sentence RNNs need only reason over much shorter time-scales.

**Sentence RNN** The sentence RNN is a single-layer LSTM with hidden size $H = 512$ and initial hidden and cell states set to zero. At each time step, the sentence RNN receives the pooled region vector $v_p$ as input, and in turn produces a sequence of hidden states $h_1, \ldots, h_S \in \mathbb{R}^H$, one for each sentence in the paragraph. Each hidden state $h_i$ is used in two ways: First, a linear projection from $h_i$ and a logistic classifier are used to produce a distribution $p_i$ over the two states $\{\texttt{CONTINUE} = 0, \texttt{STOP} = 1\}$ which determine whether the $i$th sentence is the last sentence in the paragraph. Second, the hidden state $h_i$ is fed through a two-layer fully-connected network to produce the topic vector $t_i \in \mathbb{R}^P$ for the $i$th sentence of the paragraph, which is the direct input to the word RNN.

**Word RNN** The word RNN is a two-layer LSTM with hidden size $H = 512$, which, given a topic vector $t_i \in \mathbb{R}^P$ from the sentence RNN, is responsible for generating the

words of a sentence. We follow the input formulation of [178]: the first and second inputs to the RNN are the topic vector and a special `START` token, and subsequent inputs are learned embedding vectors for the words of the sentence. At each timestep the hidden state of the last LSTM layer is used to predict a distribution over the words in the vocabulary. A special `END` token signals the end of a sentence.

### 7.4.4 Training and Sampling

Training data consists of pairs $(x, y)$, with $x$ an image and $y$ a ground-truth paragraph description for that image and where $y$ has $S$ sentences, the $i$th sentence has $N_i$ words, and $y_{ij}$ is the $j$th word of the $i$th sentence. After computing the pooled region vector $v_p$ for the image, we unroll the sentence RNN for $S$ timesteps, giving a distribution $p_i$ over the $\{\texttt{CONTINUE}, \texttt{STOP}\}$ states for each sentence. We feed the sentence topic vectors to $S$ copies of the word RNN, unrolling the $i$th copy for $N_i$ timesteps, producing distributions $p_{ij}$ over each word of each sentence. Our training loss $\ell(x, y)$ for the example $(x, y)$ is a weighted sum of two cross-entropy terms: a *sentence loss* $\ell_{sent}$ on the stopping distribution $p_i$, and a *word loss* $\ell_{word}$ on the word distribution $p_{ij}$:

$$\ell(x, y) = \lambda_{sent} \sum_{i=1}^{S} \ell_{sent}(p_i, \mathbf{I}[i = S]) + \lambda_{word} \sum_{i=1}^{S} \sum_{j=1}^{N_i} \ell_{word}(p_{ij}, y_{ij}) \tag{7.1}$$

To generate a paragraph for an image, we run the sentence RNN forward until the stopping probability $p_i(\texttt{STOP})$ exceeds a threshold $T_{\texttt{STOP}}$ or after $S_{MAX}$ sentences, whichever comes first. We then sample sentences from the word RNN, choosing the most likely word at each timestep and stopping after choosing the `STOP` token or after $N_{MAX}$ words. We set the parameters $T_{\texttt{STOP}} = 0.5$, $S_{MAX} = 6$, and $N_{MAX} = 50$ based on validation set performance.

### 7.4.5 Transfer Learning

Transfer learning has become pervasive in computer vision. For tasks such as object detection [148], image captioning [50, 87, 178, 193], and dense image captioning [85],

it has become standard practice not only to process images with convolutional neural networks, but also to initialize the weights of these networks from weights that had been tuned for image classification, such as the 16-layer VGG network [162]. Initializing from a pre-trained convolutional network allows a form of knowledge transfer from large classification datasets, and is particularly effective on datasets of limited size.

We utilize transfer learning in two ways. First, we initialize our region detection network from a model trained for dense image captioning [85]; although our model is end-to-end differentiable, we keep this sub-network fixed during training to prevent overfitting and for efficiency. Second, we initialize the word embedding vectors, recurrent network weights, and output linear projection of the word RNN from a language model that had been trained on region-level captions [85]. Parameters for tokens not present in the region model are initialized from the parameters for the `UNK` token. This initialization strategy allows our model to utilize linguistic knowledge learned on large-scale region caption datasets [96] to produce better paragraph descriptions.

## 7.5 Experiments

In this section we describe our paragraph generation experiments on the collected data described in Section 7.3, which is divided into 14,575 training, 2,487 validation, and 2,489 testing images.

### 7.5.1 Baselines

**Sentence-Concat:** To demonstrate the difference between sentence-level and paragraph captions, this baseline samples and concatenates five sentence captions from a model [87] trained on MS COCO captions [112]. The first sentence uses beam search (beam size = 2) and the rest are sampled.

**Image-Flat:** This model is equivalent to NeuralTalk [87], and takes the whole image as input, decoding into a paragraph token by token. We use the publically available

implementation of [87], which uses the 16-layer VGG network [162] to extract CNN features and projects them as input into an LSTM [75], training the whole model jointly end-to-end.

**Template:** This method represents an alternative approach to generating paragraphs, similar in style to an open-world version of more classical methods like BabyTalk [99], which converts a structured representation of an image into text via a handful of manually specified templates. The first step of our template-based baseline is to detect and describe many regions in a given target image using DenseCap [85], which produces a set of region descriptions tied with bounding boxes and detection scores. The region descriptions are parsed into a set of subjects, verbs, objects, and various modifiers according to part of speech tagging and a handful of TokensRegex [33] rules, which we find suffice to parse the vast majority of the fairly simplistic and short region-level descriptions.

Each parsed word is scored by the sum of its detection score and the log probability of the generated tokens in the original region description. Words are then merged into a coherent graph representing the scene, where each node combines all words with the same text and overlapping bounding boxes. Finally, text is generated using the top $N = 25$ scored nodes, prioritizing `subject-verb-object` triples first in generation, and representing all other nodes with existential "there is/are" statements.

**Other Baselines:** "Regions-Flat-Scratch" uses a flat language model for decoding and initializes it from scratch. The language model input is the projected and pooled region-level image features. "Regions-Flat-Pretrained" uses a pre-trained language model. These baselines are included to show the benefits of decomposing the image into regions and pre-training the language model.

### 7.5.2 Implementation Details

All baseline neural language models use two layers of LSTM [75] units with 512 dimensions. The feature pooling dimension $P$ is 1024, and we set $\lambda_{sent} = 5.0$ and $\lambda_{word} = 1.0$ based on validation set performance. Training is done via stochastic

| | METEOR | CIDEr | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 |
|---|---|---|---|---|---|---|
| Sentence-Concat | 12.05 | 6.82 | 31.11 | 15.10 | 7.56 | 3.98 |
| Template | 14.31 | 12.15 | 37.47 | 21.02 | 12.30 | 7.38 |
| Image-Flat ([87]) | 12.82 | 11.06 | 34.04 | 19.95 | 12.20 | 7.71 |
| Regions-Flat-Scratch | 13.54 | 11.14 | 37.30 | 21.70 | 13.07 | 8.07 |
| Regions-Flat-Pretrained | 14.23 | 12.13 | 38.32 | 22.90 | 14.17 | **8.85** |
| Regions-Hierarchical (ours) | **15.95** | **13.52** | **41.90** | **24.11** | **14.23** | 8.69 |
| Human | 19.22 | 28.55 | 42.88 | 25.68 | 15.55 | 9.66 |

Table 7.2: Main results for generating paragraphs. Our Region-Hierarchical method is scored with five baseline models and human performance along six language metrics.

gradient descent with Adam [89] learning rate updates. Of critical note is that model checkpoint selection is based on the best combined METEOR and CIDEr score on the validation set – although models tend to minimize validation loss fairly quickly, it takes much longer training for METEOR and CIDEr scores to stop improving.

### 7.5.3 Main Results

We present our main results at generating paragraphs in Tab. 7.2, which are evaluated across six language metrics: CIDEr [177], METEOR [47], and BLEU-{1,2,3,4} [138]. The Sentence-Concat method performs poorly, achieving the lowest scores across all metrics. Its lackluster performance provides further evidence of the stark differences between single-sentence captioning and paragraph generation. Surprisingly, the hard-coded template-based approach performs reasonably well, particularly on CIDEr, METEOR, and BLEU-1, where it is competitive with some of the neural approaches. This makes sense: the template approach is provided with a strong prior about image content since it receives region-level captions [85] as input, and the many expletive "there is/are" statements it makes, though uninteresting, are safe, resulting in decent scores. However, its relatively poor performance on BLEU-3 and BLEU-4 highlights the limitation of reasoning about regions in isolation – it is unable to produce much text relating regions to one another, and further suffers from a lack of "connective tissue" that transforms paragraphs from a series of disconnected thoughts into a coherent whole.

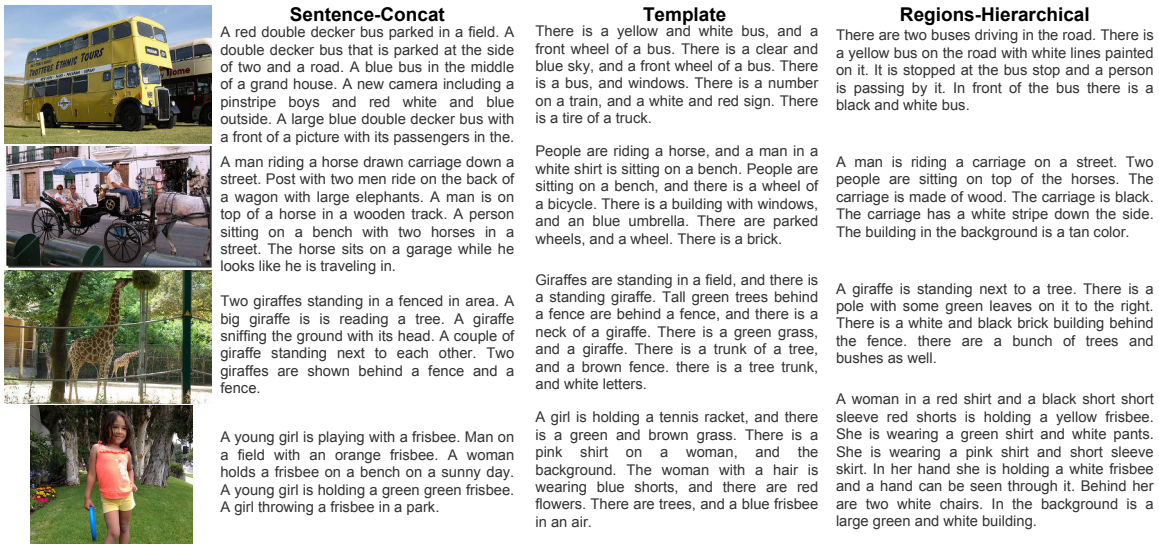



Figure 7.3: Example paragraph generation results for our model and two baselines. The first three rows are positive results and the last row is a failure case.

Image-Flat, trained on the task of paragraph generation, outperforms Sentence-Concat, and the region-based reasoning of Regions-Flat-Scratch improves results further on all metrics. Pre-training results in improvements on all metrics, and our full model, Regions-Hierarchical, achieves the highest scores among all methods on every metric except BLEU-4.

To make these metrics more interpretable, we performed a human evaluation by collecting an additional paragraph for 500 randomly chosen images, with results in the last row of Tab. 7.2. As expected, humans produce superior descriptions to any automatic method, performing better on all language metrics considered. Of particular note is the large gap between humans our the best model on CIDEr and METEOR, which are both designed to correlate well with human judgment [177, 47].

### 7.5.4 Qualitative Results

We present qualitative results from our model and the Sentence-Concat and Template baselines in Fig. 7.3. Some interesting properties of our model's predictions include its use of coreference in the first example ("a bus", "it", "the bus") and its ability to capture relationships between objects in the second example. Also of note is the order

| | Average Length | Std. Dev. Length | Diversity | Nouns | Verbs | Pronouns | Vocab Size |
|---|---|---|---|---|---|---|---|
| Sentence-Concat | 56.18 | 4.74 | 34.23 | 32.53 | 9.74 | 0.95 | 2993 |
| Template | 60.81 | 7.01 | 45.42 | 23.23 | 11.83 | 0.00 | 422 |
| Regions-Hierarchical | 70.47 | 17.67 | 40.95 | 24.77 | 13.53 | 2.13 | 1989 |
| Human | 67.51 | 25.95 | 69.92 | 25.91 | 14.57 | 2.42 | 4137 |

Table 7.3: Language statistics of test set predictions. Part of speech statistics are given as percentages, and diversity is calculated as in Section 7.3. "Vocab Size" indicates the number of unique tokens output across the entire test set, and human numbers are calculated from ground truth. Note that the diversity score for humans differs slightly from the score in Tab. 7.1, which is calculated on the entire dataset.

in which our model chooses to describe the image: the first sentence tends to be fairly high level, middle sentences give some details about scene elements mentioned earlier in the description, and the last sentence often describes something in the background, which other methods are not able to capture.

The failure case in the last row highlights another interesting phenomenon: even though our model was generally wrong about the semantics of the image, calling the girl "a woman", it has learned that "woman" is consistently associated with female pronouns ("she", "she", "her hand", "behind her").

### 7.5.5 Paragraph Language Analysis

In Tab. 7.3 we show statistics of the language generated by a representative spread of methods. Our hierarchical approach generates text of similar average length and variance as human descriptions, with Sentence-Concat and the template approach somewhat shorter and less varied in length. Sentence-Concat is also the least diverse method, and all automatic methods remain far less diverse than human sentences, indicating opportunity for improvement. Both our hierarchical approach and the template method produce text with similar portions of nouns and verbs as human paragraphs, though only the hierarchical approach was able to generate a reasonable quantity of pronouns. Our hierarchical method also had a much wider vocabulary compared to the template approach, though Sentence-Concat, trained on hundreds of thousands of MS COCO [112] captions, is a bit larger.

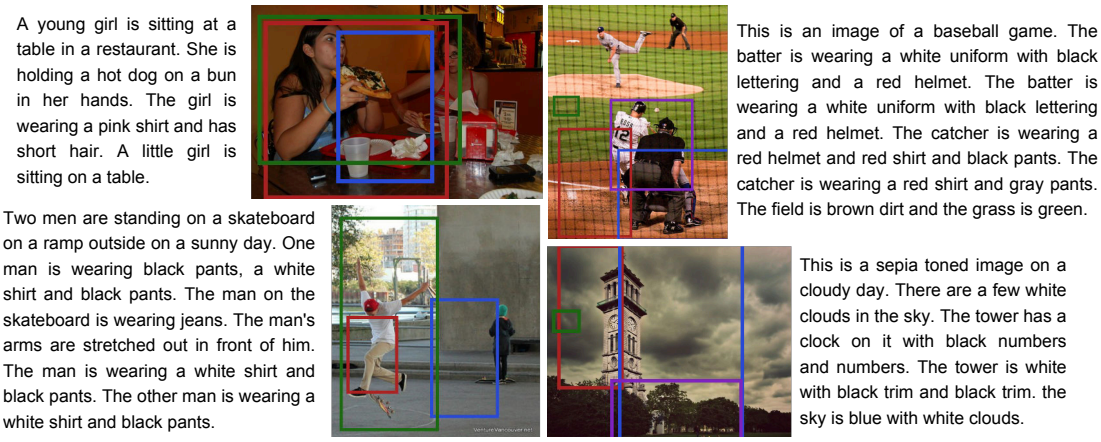



Figure 7.4: Examples of paragraph generation from only a few regions. Though out of sample, the model is able to focus on the regions of interest while ignoring the rest of the image.

### 7.5.6 Generating Paragraphs from Fewer Regions

As an exploratory experiment, we investigate generating paragraphs from a smaller number of regions than the $M = 50$ used in the rest of this work. In this experiment, shown in Fig. 7.4, we keep only the top few detected regions as input into the pooling stage of our model. Although input is extremely out of sample, results are still quite reasonable – the model generates a paragraph describing the detected regions without much mention of objects or scenery outside of the detections. For instance, in the top-right example, the paragraph generated by our model mentions the batter, catcher, dirt, and grass, which all appear in the top detected regions, but does not pay heed to the pitcher.

## 7.6 Conclusions

In this paper we have introduced the task of describing images with long, descriptive paragraphs, and presented a hierarchical neural approach for generation that leverages the compositional structure of both images and language. We have demonstrated experimentally the advantages of our approach over traditional image captioning methods and shown how region-level knowledge can be effectively transferred

to the paragraph captioning task. We anticipate further opportunities for knowledge transfer in tasks at the intersection of vision and language, and project that visual and lingual compositionality will continue to lie at the heart of effective paragraph generation.

# Chapter 8

# Conclusions

## 8.1 Overview

In this dissertation I described my work in tackling fine-grained recognition, including all three major components of the field: Data, Recognition, and Application.

I began by approaching the problem of recognizing fine-grained categories from multiple angles. In Chapter 2 I demonstrated a general approach for bringing two-dimensional representations for fine-grained recognition into 3D, leveraging freely available CAD models for rigid objects such as cars. This approach predicted the 3D geometry of novel images containing cars, which was used to normalize viewpoint and draw correspondence across images.

In Chapters 3 and 4 I presented two approaches for automatically learning parts for fine-grained recognition. Both works had the benefits of improved recognition via pose normalization without requiring annotations, making recognition more scalable.

In Chapter 6 I made fine-grained recognition even more scalable by removing the requirement of manually collecting expert labels at training time. This work made use of freely and publicly available images from the web, recognizing that data is actually the limiting factor in methods for fine-grained recognition. As a result, performance on existing fine-grained categories was dramatically improved, nearing expert performance, and scalability was improved to the point that recognition on over 10,000 fine-grained categories was possible for the first time.

Chapter 5 presented an application of fine-grained recognition in understanding society, using the metadata present in fine-grained categories of cars to analyze and predict demographics across 200 cities in the United States. Our approach not only allowed for effective prediction, but also was interpretable – we were able to analyze which attributes of cars best predicted or otherwise correlated with different demographic factors.

Finally, in Chapter 7 I introduced the problem of image captioning at the level of paragraphs, extending the traditional image captioning problem into a much greater amount of detail. On the algorithmic side, our approach decomposed both images and paragraphs into their constituent parts.

## 8.2 Open Questions

Is fine-grained recognition solved? Not quite. Due to the large amount of progress made in recent years, caused only partially by the contributions contained in this dissertation, it is tempting to say that no work is left to be done in fine-grained recognition. Nonetheless, there are several important questions left to consider:

**Difficult Categories.** Not all object categories can be effectively recognized with current state-of-the-art methods for fine-grained recognition. Despite our best efforts, we are still a long ways away from recognizing all 941,000 species of insects [74], and doing so will require both algorithmic and data-level innovations. There are even fine-grained domains with a comparatively small number of categories that are challenging to tackle – consider the case of medical imaging, where data is completely private and the relevant experts who would need to annotate it are doctors. While approaching such a domain may be possible, it will require an enormous amount of effort to succeed. For other fine-grained domains, particularly man-made ones, it may even be difficult to define a category list, a challenge encountered in Chapters 2 and 5.

**Fine-Grained Detection.** Fine-grained recognition has thus far focused on the classification problem. What advances need to be made in order to push fine-grained

recognition into other problems, *e.g.* detection, segmentation, or video understanding? One key limiting factor here is data: since it is much easier to verify a fine-grained category rather than label it from scratch, most fine-grained datasets today are collected by first querying web search engines for category names. As a result, images in fine-grained datasets tend to be object-centric, featuring a single, salient object occupying the majority of the image. In contrast, datasets useful for detection may have objects of a variety of sizes and many objects per image. While this has been partially addressed in Chapter 5, more work remains. One interesting research direction along these lines is to combine generic object detection with fine-grained classification, possibly doing so in a domain adaptation framework.

**Category Knowledge.** Fine-grained categories aren't simply 1 of $K$ labels – they are real-world objects that have valuable semantic properties. For example, a Northern Cardinal is a bird species that is mostly red except for a black mask on its face, black eyes, and orange beak. Fine-grained categories are exactly the domain where these types of properties make sense to reason about due to the shared structure and common vocabulary between categories. This may prove helpful in scaling recognition up to the long tail of fine-grained categories in the world, and may also be valuable as an output of its own, useful for interpretability and communication with humans.

**Subcategories.** As fine-grained recognition continues to improve, it makes sense to start reasoning about even finer-grained categories, effectively reasoning about *subcategories*. In the case of bird species, this may involve predicting whether a bird is the male or female of its species, which often look different, or even whether the bird is a juvenile or adult. While some initial work has been done along these lines [174], it is my belief that this topic will only become increasingly important in the near future.

# Bibliography


[1] American Community Survey 5 Year Data (2008-2012). http://www.census.gov/data/developers/data-sets/acs-survey-5-year-data.html#notes.

[2] Differences in new vehicle buyers' ethnicities predict where future sales will grow. http://www.strategicvision.com/auto_2014_ethnic_release.php.

[3] SFPD Crime Reporting Plots — Data — San Francisco. https://data.sfgov.org/Public-Safety/SFPD-Crime-Reporting-Plots-Zipped-Shapefile-Format/5aii-qc4e.

[4] Rapping 2 u lyrics. http://genius.com/77142/Das-racist-rapping-2-u/White-people-love-me-like-they-love-subarus.

[5] Relocation Essentials. http://www.relocationessentials.com/.

[6] State $CO_2$ Emissions. http://www.eia.gov/environment/emissions/state/.

[7] TIGER/Line - Geography - U.S. Census Bureau. https://www.census.gov/geo/maps-data/data/tiger-line.html.

[8] Patricia A Adler. *Wheeling and dealing: An ethnography of an upper-level drug dealing and smuggling community*. Columbia University Press, 1993.

[9] Anelia Angelova and Shenghuo Zhu. Efficient object detection and segmentation for fine-grained recognition. In *CVPR*, 2013.

[10] Anelia Angelova, Shenghuo Zhu, and Yuanqing Lin. Image segmentation for large-scale subcategory flower recognition. In *Workshop on Applications of Computer Vision (WACV)*, pages 39–45. IEEE, 2013.

[11] Stephen Ansolabehere, Maxwell Palmer, and Amanda Lee. Precinct-Level Election Data. `http://hdl.handle.net/1903.1/21919`, 2014.

[12] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[13] Sean M Arietta, Alexei A Efros, Ravi Ramamoorthi, and Maneesh Agrawala. City forensics: Using visual elements to predict non-visual city attributes. 2014.

[14] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *Learning Theory*, pages 35–50. Springer, 2007.

[15] Richard E Barlow, David J Bartholomew, JM Bremner, and H Daniel Brunk. *Statistical inference under order restrictions: the theory and application of isotonic regression*. Wiley New York, 1972.

[16] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[17] Thomas Berg and Peter N Belhumeur. How do you tell a blackbird from a crow? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9–16, 2013.

[18] Thomas Berg and Peter N Belhumeur. Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 955–962. IEEE, 2013.

[19] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[20] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[21] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. *NIPS*, 2010.

[22] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009.

[23] William Brangham. Running on renewable energy, Burlington, Vermont powers green movement forward. `http://www.pbs.org/newshour/bb/vermont-city-come-rely-100-percent-renewable-energy/`.

[24] Steve Branson, Oscar Beijbom, and Serge Belongie. Efficient large-scale structured learning. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1806–1813, 2013.

[25] Steve Branson, Grant Van Horn, Pietro Perona, and Serge Belongie. Improved bird species recognition using pose normalized deep convolutional nets. In *British Machine Vision Conference*, 2014.

[26] Steve Branson, Grant Van Horn, Catherine Wah, Pietro Perona, and Serge Belongie. The ignorant led by the blind: A hybrid human–machine vision system for fine-grained categorization. *International Journal of Computer Vision*, pages 1–27, 2014.

[27] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. *ECCV*, 2010.

[28] Timothy Cain. U.S. Auto Sales By Brand - 2011 Year End. `http://www.goodcarbadcar.net/2012/01/us-auto-sales-by-brand-2011-year-end.html/`.

[29] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. Bicos: A bi-level co-segmentation method for image classification. In *ICCV*, 2011.

[30] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *ICCV*, 2013.

[31] Yuning Chai, Esa Rahtu, Victor Lempitsky, Luc Van Gool, and Andrew Zisserman. Tricos: a tri-level class-discriminative co-segmentation method for image classification. In *ECCV*. 2012.

[32] William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. Listen, attend, and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*, 2016.

[33] Angel X Chang and Christopher D Manning. Tokensregex: Defining cascaded regular expressions over tokens. Technical report, CSTR 2014-02, Department of Computer Science, Stanford University, 2014.

[34] Guang Chen, Jianchao Yang, Hailin Jin, Jonathan Brandt, Eli Shechtman, Aseem Agarwala, and Tony X Han. Large-scale visual font recognition. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014.

[35] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *International Conference on Computer Vision (ICCV)*. IEEE, 2015.

[36] Xinlei Chen, Ashish Shrivastava, and Arpan Gupta. Neil: Extracting visual knowledge from web data. In *International Conference on Computer Vision (ICCV)*, pages 1409–1416. IEEE, 2013.

[37] Xinlei Chen and C Lawrence Zitnick. Minds eye: A recurrent visual representation for image caption generation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[38] Sangho Choo and Patricia L Mokhtarian. What type of vehicle do people drive? the role of attitude and lifestyle in influencing vehicle type choice. *Transportation Research Part A: Policy and Practice*, 38(3):201–222, 2004.

[39] David Cline, Stefan Jeschke, K White, Anshuman Razdan, and Peter Wonka. Dart throwing on surfaces. In *Eurographics*, 2009.

[40] Brendan Collins, Jia Deng, Kai Li, and Li Fei-Fei. Towards scalable dataset construction: An active learning approach. In *European Conference on Computer Vision (ECCV)*, pages 86–98. Springer, 2008.

[41] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cedric Bray. Visual categorization with bags of keypoints. In *ECCV*, 2004.

[42] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[43] Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of the 22nd international conference on World Wide Web*, pages 307–318. International World Wide Web Conferences Steering Committee, 2013.

[44] Hal Daumé III. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, 2007.

[45] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[46] Jia Deng, Jonathan Krause, and Li Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2013.

[47] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *EACL Workshop on Statistical Machine Translation*, 2014.

[48] Santosh K Divvala, Alireza Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3270–3277. IEEE, 2014.

[49] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[50] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[51] Kun Duan, Devi Parikh, David Crandall, and Kristen Grauman. Discovering localized attributes for fine-grained recognition. In *CVPR*, 2012.

[52] Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS*, 1995.

[53] Ayse Naz Erkan. *Semi-supervised learning via generalized maximum entropy.* PhD thesis, New York University, 2010.

[54] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *JMLR*, 2008.

[55] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In *ECCV*, 2010.

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

[57] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. *Pattern Analysis and Machine Intelligence*, 32(9), 2010.

[58] Rob Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. Learning object categories from internet image searches. *Proceedings of the IEEE*, 98(8):1453–1466, 2010.

[59] Richard Florida and Charlotta Mellander. Democrat vs. Republican: Whos Buying What Car? Technical report, Strategic Vision, 3 2012.

[60] Richard Florida and Charlotta Mellander. Segregated City: The Geography of Economic Segregation in Americas Metros. Technical report, Martin Prosperity Institute, Rotman School of Management, University of Toronto, 2 2015.

[61] Efstratios Gavves, Basura Fernando, Cees GM Snoek, Arnold WM Smeulders, and Tinne Tuytelaars. Local alignments for fine-grained categorization. *International Journal of Computer Vision*, pages 1–22, 2014.

[62] Efstratios Gavves, Basura Fernando, CGM Snoek, AWM Smeulders, and Tinne Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, 2013.

[63] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Visual census: Using cars to study people and society.

[64] Arthur Getis and J Keith Ord. The analysis of spatial association by use of distance statistics. *Geographical analysis*, 24(3):189–206, 1992.

[65] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[66] Christoph Goering, Erik Rodner, Alexander Freytag, and Joachim Denzler. Nonparametric part transfer for fine-grained recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2489–2496. IEEE, 2014.

[67] 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.

[68] Matthieu Guillaumin, Daniel Küttel, and Vittorio Ferrari. Imagenet auto-annotation with segmentation propagation. *International Journal of Computer Vision (IJCV)*, pages 1–21, 2014.

[69] Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV'10*.

[70] Paritosh Gupta, Sai Sankalp Arrabolu, Mathew Brown, and Silvio Savarese. Video scene categorization by 3d hierarchical histogram matching. In *ICCV*, 2009.

[71] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[73] Aharon Bar Hillel and Daphna Weinshall. Subordinate class recognition using relational object models. In *NIPS*, 2006.

[74] Cody E. Hinchliff, Stephen A. Smith, James F. Allman, J. Gordon Burleigh, Ruchi Chaudhary, Lyndon M. Coghill, Keith A. Crandall, Jiabin Deng, Bryan T. Drew, Romina Gazis, Karl Gude, David S. Hibbett, Laura A. Katz, H. Dail Laughinghouse, Emily Jane McTavish, Peter E. Midford, Christopher L. Owen, Richard H. Ree, Jonathan A. Rees, Douglas E. Soltis, Tiffani Williams, and Karen A. Cranston. Synthesis of phylogeny and taxonomy into a comprehensive tree of life. *Proceedings of the National Academy of Sciences*, 2015.

[75] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[76] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR*, 2013.

[77] David Hoiem, Alexei Efros, and Martial Hebert. Putting objects in perspective. *IJCV*, 2008.

[78] Gary B Huang, Honglak Lee, and Erik Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *CVPR*, 2012.

[79] Gary B Huang, Marwan Mattar, Honglak Lee, and Erik G Learned-Miller. Learning to align from scratch. In *NIPS*, 2012.

[80] Edison Media Research/Mitofsky International. Election results 2008. *The New York Times*, 11 2008.

[81] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[82] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *WS ACM SIGKDD*, 2010.

[83] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Neural Information Processing Systems (NIPS)*, 2015.

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

[85] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016.

[86] Armand Joulin, Francis Bach, and Jean Ponce. Discriminative clustering for image co-segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1943–1950. IEEE, 2010.

[87] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[88] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-fei Li. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, CVPR*. Citeseer, 2011.

[89] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[90] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork RNN. In *ICML*, 2014.

[91] Adam DI Kramer, Jamie E Guillory, and Jeffrey T Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24):8788–8790, 2014.

[92] Jonathan Krause, Timnit Gebru, Jia Deng, Li-Jia Li, and Li Fei-Fei. Learning features and parts for fine-grained recognition. In *International Conference on Pattern Recognition (ICPR)*, Stockholm, Sweden, August 2014.

[93] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

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

[95] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. IEEE, 2013.

[96] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma,

Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*, 2016.

[97] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[98] Daniel Kuettel, Matthieu Guillaumin, and Vittorio Ferrari. Segmentation propagation in imagenet. In *European Conference on Computer Vision (ECCV)*, pages 459–473. Springer, 2012.

[99] Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. Baby talk: Understanding and generating image descriptions. In *CVPR*, 2011.

[100] Neeraj Kumar, Peter N Belhumeur, Arijit Biswas, David W Jacobs, W John Kress, Ida C Lopez, and João VB Soares. Leafsnap: A computer vision system for automatic plant species identification. In *ECCV*. 2012.

[101] Frances E Kuo and William C Sullivan. Environment and crime in the inner city does vegetation reduce crime? *Environment and behavior*, 33(3):343–367, 2001.

[102] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[103] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[104] Jennifer Lees-Marshment, Brian Conley, and Kenneth Cosgrove. *Political Marketing in the United States*. Routledge, 2014.

[105] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *International Conference on Computer Vision (ICCV)*, pages 277–284. IEEE, 2009.

[106] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *International Conference on Machine Learning (ICML)*, pages 148–156, 1994.

[107] David Lanier Lewis and Laurence Goldstein. *The automobile and American culture*. University of Michigan Press, 1983.

[108] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*, 2015.

[109] Li-Jia Li and Li Fei-Fei. Optimol: automatic online picture collection via incremental model learning. *International Journal of Computer Vision (IJCV)*, 88(2):147–168, 2010.

[110] Joerg Liebelt and Cordelia Schmid. Multi-view object class detection with a 3d geometric model. In *CVPR*, 2010.

[111] Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. Hierarchical recurrent neural network for document modeling. In *EMNLP*, 2015.

[112] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[113] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *International Conference on Computer Vision (ICCV)*. IEEE.

[114] Yen-Liang Lin, Vlad I Morariu, Winston Hsu, and Larry S Davis. Jointly optimizing 3d model fitting and fine-grained classification. In *European Conference on Computer Vision (ECCV)*, pages 466–480. Springer, 2014.

[115] Jiongxin Liu, Angjoo Kanazawa, David Jacobs, and Peter Belhumeur. Dog breed classification using part localization. In *ECCV*. 2012.

[116] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[117] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[118] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[119] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford CoreNLP natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[120] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *International Conference on Learning Representations (ICLR)*, 2015.

[121] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[122] Clay McShane and Peter J Hugill. *Down the asphalt path: The automobile and the American city*, volume 134. Columbia University Press New York, 1994.

[123] Mary Meeker. Internet trends 2014-code conference. `http://www.kpcb.com/internet-trends`, 2014.

[124] Metropolitan Area Planning Council. The Massachusetts vehicle census. `http://www.37billionmilechallenge.org/`.

[125] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182, 2011.

[126] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *International Conference on Machine Learning (ICML)*, pages 567–574, 2012.

[127] Patrick AP Moran. Notes on continuous stochastic phenomena. *Biometrika*, pages 17–23, 1950.

[128] Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. Scaling up crowd-sourcing to very large datasets: a case for active learning. *Proceedings of the VLDB Endowment*, 8(2):125–136, 2014.

[129] Joann Muller. What The Rich People Really Drive. `http://www.forbes.com/sites/joannmuller/2011/12/30/what-the-rich-people-really-drive/`.

[130] Margaret Myers and Sharon G Dean. Cadillac flambé: Race and brand identity. In *Proceedings of the 23rd conference on historical analysis and research in marketing*, volume 13, pages 158–161, 2007.

[131] Nikhil Naik, Jade Philipoom, Ramesh Raskar, and Cesar Hidalgo. Streetscore—predicting the perceived safety of one million streetscapes. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 793–799. IEEE, 2014.

[132] Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006.

[133] Maria-Elena Nilsback and Andrew Zisserman. Delving into the whorl of flower segmentation. In *BMVC*, 2007.

[134] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.

[135] Department of Commerce U.S. Census Bureau. U.S. Census Bureau's Budget Estimates As Presented to Congress April 2013. Technical report, Department of Commerce U.S. Census Bureau, 4 2013.

[136] J Keith Ord and Arthur Getis. Local spatial autocorrelation statistics: distributional issues and an application. *Geographical analysis*, 27(4):286–306, 1995.

[137] Vicente Ordonez and Tamara L Berg. Learning high-level judgments of urban perception. In *Computer Vision–ECCV 2014*, pages 494–510. Springer, 2014.

[138] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. Association for Computational Linguistics, 2002.

[139] Devi Parikh and Kristen Grauman. Interactively building a discriminative vocabulary of nameable attributes. In *CVPR*, 2011.

[140] Omkar M Parkhi, Andrea Vedaldi, CV Jawahar, and Andrew Zisserman. The truth about cats and dogs. In *ICCV*, 2011.

[141] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012.

[142] Bojan Pepik, Peter Gehler, Michael Stark, and Bernt Schiele. $3d^2pm$ – 3d deformable part models. In *ECCV*, 2012.

[143] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[144] Jian Pu, Yu-Gang Jiang, Jun Wang, and Xiangyang Xue. Which looks like which: Exploring inter-class relationships in fine-grained visual categorization. In *European Conference on Computer Vision (ECCV)*, pages 425–440. Springer, 2014.

[145] Sean F Reardon and David O'Sullivan. Measures of spatial segregation. *Sociological methodology*, 34(1):121–162, 2004.

[146] Carolina Redondo-Cabrera, Roberto Javier Lopez-Sastre, Javier Acevedo-Rodríguez, and Saturnino Maldonado-Bascón. Surfing the point clouds: Selective 3d spatial pyramids for category-level object recognition. In *CVPR*, 2012.

[147] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[148] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[149] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[150] Michael Rubinstein, Armand Joulin, Johannes Kopf, and Ce Liu. Unsupervised joint object discovery and segmentation in internet images. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1939–1946. IEEE, 2013.

[151] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015.

[152] Silvio Savarese and Li Fei-Fei. 3D generic object categorization, localization and pose estimation. In *ICCV*, 2007.

[153] Wilbur Lang Schramm. *The story of human communication: Cave painting to microchip*. Harpercollins College Division, 1988.

[154] Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Harvesting image databases from the web. *Pattern Analysis and Machine Intelligence (PAMI)*, 33(4):754–766, 2011.

[155] Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

[156] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[157] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1289–1296, 2008.

[158] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008.

[159] Kevin J Shih, Arun Mallya, Saurabh Singh, and Derek Hoiem. Part localization using multi-proposal consensus for fine-grained categorization. In *British Machine Vision Conference (BMVC)*, 2015.

[160] Marcel Simon and Erik Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *ICCV*, 2015.

[161] Marcel Simon, Erik Rodner, and Joachim Denzler. Part detector discovery in deep convolutional neural networks. In *Asian Conference on Computer Vision (ACCV)*, volume 2, pages 162–177, 2014.

[162] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[163] Mary S Smith. *Crime prevention through environmental design in parking facilities*. US Department of Justice, Office of Justice Programs, National Institute of Justice, 1996.

[164] Michael Stark, Jonathan Krause, Bojan Pepik, David Meger, James J. Little, Bernt Schiele, and Daphne Koller. Fine-grained categorization for 3d scene understanding. In *BMVC*, 2012.

[165] Hao Su, Jia Deng, and Li Fei-Fei. Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[166] Sainbayar Sukhbaatar and Rob Fergus. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, 2014.

[167] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

[168] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[169] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[170] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[171] John Tierney. Your car: politics on wheels. *New York Times*, 1, 2005.

[172] Antonio Torralba, Alexei Efros, et al. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1521–1528. IEEE, 2011.

[173] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[174] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015.

[175] Andrea Vedaldi and Brian Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[176] Andrea Vedaldi, Siddharth Mahendran, Stavros Tsogkas, Subhransu Maji, Ross Girshick, Juho Kannala, Esa Rahtu, Iasonas Kokkinos, Matthew B Blaschko, David Weiss, Ben Taskar, Karen Simonyan, Naomi Saphra, and Sammy Mohamed. Understanding objects in detail with fine-grained attributes. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[177] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. CIDEr: Consensus-based image description evaluation. In *CVPR*, 2015.

[178] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

[179] Catherine Wah and Serge Belongie. Attribute-based detection of unfamiliar classes with humans in the loop. In *Computer Vision and Pattern Recognition (CVPR)*, pages 779–786. IEEE, 2013.

[180] Catherine Wah, Steve Branson, Pietro Perona, and Serge Belongie. Multiclass recognition and part localization with humans in the loop. In *ICCV'11*.

[181] Catherine Wah, Steve Branson, Pietro Perona, and Serge Belongie. Multiclass recognition and part localization with humans in the loop. In *ICCV*, 2011.

[182] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[183] Catherine Wah, GV Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. Similarity comparisons for interactive fine-grained categorization. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[184] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity

with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

[185] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR'10*.

[186] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

[187] James Q Wilson and George L Kelling. Broken windows. *Atlantic monthly*, 249(3):29–38, 1982.

[188] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M. Seitz. Multicore bundle adjustment. In *CVPR*, 2011.

[189] Yu Xiang and Silvio Savarese. Estimating the aspect layout of object categories. In *CVPR*, 2012.

[190] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[191] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[192] Saining Xie, Tianbao Yang, Xiaoyu Wang, and Yuanqing Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[193] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. 2015.

[194] Zhe Xu, Shaoli Huang, Ya Zhang, and Dacheng Tao. Augmenting strong supervision using web data for fine-grained categorization. In *International Conference on Computer Vision (ICCV)*, 2015.

[195] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[196] Shulin Yang, Liefeng Bo, Jue Wang, and Linda Shapiro. Unsupervised template learning for fine-grained object recognition. In *NIPS*, 2012.

[197] Bangpeng Yao, Gary Bradski, and Li Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, 2012.

[198] Bangpeng Yao, Aditya Khosla, and Li Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *CVPR'11*.

[199] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *ICCV*, 2015.

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

[201] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *CVPR*, 2016.

[202] Christoph Zauner. Implementation and benchmarking of perceptual image hash functions. *Master's thesis, Austria*.

[203] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision (ECCV)*, pages 834–849. Springer, 2014.

[204] Ning Zhang, Ryan Farrell, and Trever Darrell. Pose pooling kernels for sub-category recognition. In *CVPR*, 2012.

[205] Ning Zhang, Ryan Farrell, Forrest Iandola, and Trevor Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *ICCV*, 2013.

[206] Yu Zhang, Xiu-shen Wei, Jianxin Wu, Jianfei Cai, Jiangbo Lu, Viet-Anh Nguyen, and Minh N Do. Weakly supervised fine-grained image categorization. *arXiv preprint arXiv:1504.04943*, 2015.

[207] Bolei Zhou, Liu Liu, Aude Oliva, and Antonio Torralba. Recognizing city identity via attribute analysis of geo-tagged images. In *Computer Vision–ECCV 2014*, pages 519–534. Springer, 2014.

[208] M Zeeshan Zia, Michael Stark, Bernt Schiele, and Konrad Schindler. Revisiting 3d geometric models for accurate object shape and pose. In *3dRR11*.