

UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

Sharing information across object templates

Permalink

<https://escholarship.org/uc/item/0409b08s>

Author

Zhu, Xiangxin

Publication Date

2014

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,
IRVINE

Sharing Information Across Object Templates

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Xiangxin Zhu

Dissertation Committee:
Professor Deva Ramanan, Chair
Professor Charless Fowlkes
Professor Alexander Ihler

2014

DEDICATION

To my family

TABLE OF CONTENTS

	Page
LIST OF FIGURES	v
LIST OF TABLES	vii
ACKNOWLEDGMENTS	viii
CURRICULUM VITAE	ix
ABSTRACT OF THE DISSERTATION	x
1 Introduction	1
1.1 Summary of sharing approaches in this thesis	4
1.2 Thesis Overview and Contributions	5
2 Is big training data sufficient?	8
2.1 Introduction	8
2.2 Related Work	11
2.3 Big Detection Datasets	11
2.3.1 Collecting PASCAL-10X	13
2.3.2 Data Quality	14
2.4 Mixture models	16
2.5 Experiments	18
2.5.1 The importance of proper regularization	22
2.5.2 The importance of clean training data	22
2.5.3 Performance of mixture models	24
2.6 Conclusion	28
3 Capturing long-tail distributions of object subcategories	29
3.1 Long-tail and its challenges	31
3.2 Related work	33
3.3 Learning long-tail subcategory models	35
3.3.1 Initialization	36
3.3.2 Discriminative clustering with sharing	37
3.3.3 Greedy model selection	39
3.4 Experimental results	40

4	Sharing local appearance with parts	48
4.1	Deformable part model	49
4.1.1	Tree structured part model	49
4.1.2	Shape model	50
4.1.3	Inference	51
4.1.4	Learning	52
4.2	Revisit mixture models	53
4.2.1	Deformable Part Models (DPMs)	54
4.2.2	Exemplar Part Models (EPMs)	56
4.2.3	Exemplar DPMs (EDPMs)	57
4.2.4	Inference	58
4.3	Experiments	60
4.4	Conclusion	65
5	Face analysis in the wild: A case study	68
5.1	Introduction	69
5.2	Related Work	71
5.3	Model	72
5.4	Inference	74
5.5	Learning	74
5.6	Experimental Results	75
5.6.1	Datasets	75
5.6.2	Our models	77
5.6.3	Face detection	78
5.6.4	Pose estimation	83
5.6.5	Landmark localization	85
5.7	Diagnostic analysis and discussion	88
5.8	Conclusion	94
	Bibliography	96

LIST OF FIGURES

	Page
1.1 Long-tail distributions exist for object subcategories	2
1.2 Illustration of sharing	3
1.3 Explain different ways of sharing data	5
2.1 Best reported results on PASCAL VOC over years	9
2.2 Ideal curves of performance vs training data size and model complexity	10
2.3 MTurk user interfaces for image classification and object annotation	15
2.4 Average images of the supervised and k-means face clusters	19
2.5 Average images of the supervised and k-means bus clusters	20
2.6 Training and testing errors with and without cross validate on C	21
2.7 Performance curves and templates of the single mixture models trained with all and frontal only faces	23
2.8 Visualization of the clean and noisy bicycle mixture components	24
2.9 Performance curves of the supervised vs k-mean clustering on face and bus	25
2.10 Average precision vs the number mixtures and the number training examples on face and bus	26
2.11 Average precision at varying amount of training data on 11 PASCAL categories	27
3.1 Long-tail distributions for object categories and performance per subcategory	30
3.2 Long-tail distributions exist for both object categories and subcategories	32
3.3 Illustration of overlapping subcategories	35
3.4 Overall pipeline of learned subcategory models	35
3.5 Visualization of the exemplar templates with and without sharing	36
3.6 Visualization of the size of each subcategory clusters, the averages image and templates associated with the subcategories	41
3.7 Examples of the image popularity in training	42
3.8 Precision-recall curves of our DistDPM on two public dataset of cats and dogs	43
3.9 Precision-recall curves of 8 PASCAL VOC categories	43
3.10 Performance vs the number of subcategory mixtures on car and cat	45
3.11 Comparison of various methods on hard and easy classes	46
4.1 Illustration of local sharing via parts	49
4.2 Classic exemplar templates vs the EPM templates	57
4.3 Shape models for DPM, EPM and EDPM	59
4.4 Complete plot of the performance of various methods on face detection	61

4.5	Complete plot of the performance of various methods on bus detection	62
4.6	Visualization of detections using our EDPM	64
5.1	Example of joint face detection, pose estimation and landmark localization	70
5.2	Visualization of our face templates	70
5.3	Deformation patterns of our FaceDPM and AAMs	73
5.4	Example images from our annotated faces-in-the-wild (AFW) testing set.	75
5.5	Example images from MultiPIE with annotated landmarks.	75
5.6	ROC curves for face detection on the FDDB benchmark	79
5.7	Number of positive examples for training	80
5.8	Examples of missed faces on FDDB	81
5.9	Precision-recall curves for face detection on our AFW testset	82
5.10	Qualitative results of our FaceDPL on AFW images	83
5.11	Cumulative error distribution curves for pose estimation	85
5.12	Cumulative localization error curves on the frontal faces from MultiPIE	86
5.13	Cumulative error distribution curves for landmark localization	87
5.14	Example AFW image with large mouth deformations	88
5.15	Performance vs the number of positive examples	90
5.16	Performance vs the number of negative images	91
5.17	Performance vs the number of mixture components	92
5.18	Visualization of star structures and the learned tree structures	93
5.19	Effects on the performance when changing various model settings	94

LIST OF TABLES

	Page
2.1 Statistics of our dataset and PASCAL 2010 trainval	12
2.2 Attributes of horse in our dataset, PASCAL 2007 and 2010	14
3.1 Results on PASCAL 2007	44
5.1 Counts of the missed faces by our FaceDPL on the FDDB dataset	80

ACKNOWLEDGMENTS

I owe a great thanks to my advisor, Deva Ramanan. Anyone who has met him is struck by his intelligence, kindness, and generosity. It has been a tremendous inspiration to work with them. Without his tireless supports and guidance, this work would not be possible. It's a pleasure to have known him and I consider myself extremely lucky for having been his student.

Thanks to Charless Fowlkes and Max Welling for all the discussions through the years and providing key insights and suggestions into my work. Working with them has been a memorable experience.

I have been fortunate to work with Drago Anguelov at Google's visual search team. The way that he approaches and solves real world problems shows me an excellent example of what makes a great engineer.

Over the past five years it has been my honor to collaborate with my co-authors: Carl Vondrick, Anoop Korattikara, John Lowengrub and Fang Jin.

Thanks to my committee: Deva, Charless and Alex Ihler, for carefully reading my thesis and providing valuable suggestions on improving it.

Thanks to my wonderful colleagues and friends in the UCI vision lab, who made the PhD journey much more enjoyable than I thought would be: Chaitanya Desai, Yi Yang, Dennis Park, Hamed Pirsiavash, Sam Hallman, Mohsen Hejrati James Supancic, Bailey Kong, Raul Diaz and Golnaz Ghiasi. I also thank all my friends, near and far, for their support and care.

Very special thanks to my wife, Yang Yang, for her love and supports during bright and dark days. Debating on vision problems in the kitchen is actually fun. Thanks to my parents for everything.

CURRICULUM VITAE

Xiangxin Zhu

EDUCATION

Doctor of Philosophy in Computer Science

University of California Irvine

2014

Irvine, CA

Master of Science in Pattern Recognition

Chinese Academy of Sciences

2008

Beijing, China

Bachelor of Science in Automation

Tsinghua University

2005

Beijing, China

ABSTRACT OF THE DISSERTATION

Sharing Information Across Object Templates

By

Xiangxin Zhu

Doctor of Philosophy in Computer Science

University of California, Irvine, 2014

Professor Deva Ramanan, Chair

Object detection is a central and challenging task in computer vision. In this thesis, we first examine the “big data” hypothesis: object detection might be solved with simple models backed with massive training data. We empirically show that the performance of one of the state-of-the-art methods (discriminatively trained HoG templates) tends to saturate fast when fed with more data. The required training data may need to grow exponentially in order to produce a fixed improvement in accuracy. We also find that the key difficulties in detection are large variation in object appearance and more importantly, that the variation exhibits a “long tail” distribution: there are many rare cases with little training data, which makes those cases hard to model. This thesis addresses such challenges by proposing new representations that share information within and across object subcategories. Sharing allows one to learn models for rare subcategories in the long-tail where traditional approaches suffer from lack of training data. We investigate two methods for sharing. We first examine global models that share entire training examples across multiple subcategories. For example, an SUV image might be used to train both a car and truck subcategory model. We also examine local sharing that share subwindows of training examples through “parts”. For example, nearly all vehicles contain wheel parts. By mixing and matching (or composing) different parts together, one can implicitly encode an exponentially large set of subcategory models, which could even represent those subcategories not encountered in the training data.

We extensively experiment and evaluate our models on different benchmarks, and show superior

performance over the state-of-the-art. Finally, we conclude with a detailed analysis of local part sharing for face analysis, perhaps the most well studied of all object recognition problems. By using semantically-defined parts (such as eyes, nose, lips), one can simultaneously perform face detection, pose estimation, and landmark localization with state-of-the-art accuracy, with a single model.

Chapter 1

Introduction

Object detection is of central importance in computer vision, where the goal is to automatically localize and identify the extent of object instances within an image. Truly successful detection systems will be the cornerstone for many applications such as robotics, surveillance, human-computer interaction, biometrics, and image retrieval. Object detection is also one of the most challenging tasks. Despite all the advances in the past 30 years, the dream of having a computer interpret an image at the same level as a two-year old (such as find and count the animals and toys) remains elusive.

Why is object detection so hard? There exist many explanations, but one standard answer is that objects vary in appearance. In theory, if we collect a large enough dataset that cover all possible appearance variations, and build separate mixture components or subcategory models on different viewpoints, shape deformation, etc, that should address the problem (with possibly huge computational costs). This thought aligns with an emerging idea in our community that object detection might be solved with simple models backed with massive training data.

This leads us to consider a basic question: will continually increasing amounts of training data along with large mixture models be sufficient to drive continued progress in object detection?

In this thesis, we empirically found that the answer is “no”. Our experimental results show that the performance tends to saturate after a modest number of mixtures and a modest size of training data. Then what prevents this “nonparametric” paradigm from working? There are many possible reasons. One of the key difficulties is that the variation exhibits a “long tail” distribution (Fig. 1.1): there are a small number of common cases and large (probably infinite) number of rare cases which collectively make up a significant portion of the data. When collecting new data, sampling a long-tail distribution tends to rarely hit a particular subcategory in the tail. This suggests that even as one grows a training set, one is unlikely to encounter many examples of a particular rare subcategory in the tail. Empirically, we observed that the required training data may need to grow exponentially in order to produce a fixed improvement in accuracy. Given the size of existing datasets, a simple extrapolation shows that the current state-of-the-art will need unrealistically large amount of data to continue producing consistent improvements in performance.

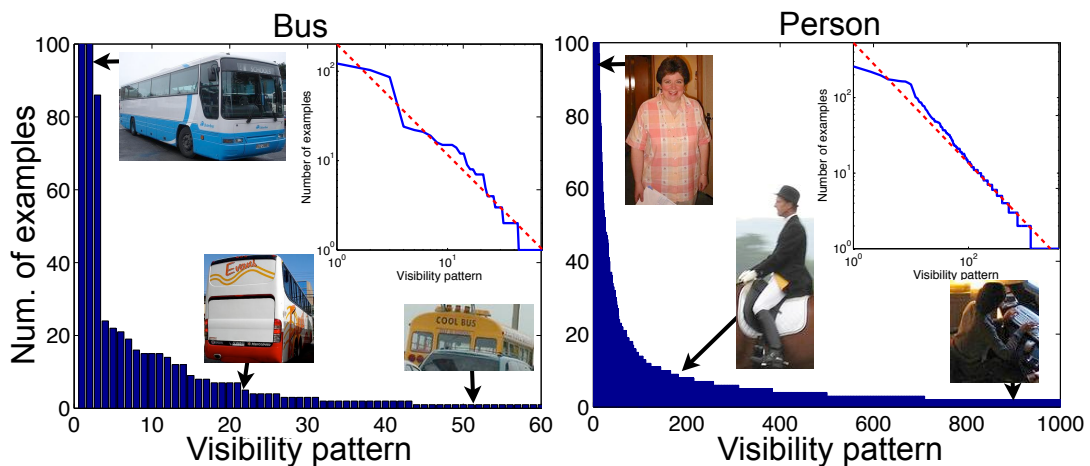


Figure 1.1: Long-tail distributions exist for object subcategories. This figure shows the distributions of the keypoint visibility patterns for bus and person from PASCAL (using the manual annotations of [13]). The blue curve in the inset show a log-log plot, along with a best-fit line in red. This suggests that the distribution follows a long-tail power law. How to strictly define “subcategory” and how to group examples into subcategories are still open questions. Visibility patterns of keypoints are used as a proxy for general appearance in generating this figure, as they represent the variations due to viewpoints and occlusions. In this thesis, we describe an approach to *discover* the subcategories, rather than pre-define it. We empirically show that the visual subcategory distributions follow a long-tail.

In this thesis, in order to address the lack of training data for the rare cases, we propose sharing training examples across subcategories. Two types of sharing are discussed: global sharing allows borrowing similar examples from other subcategories. This introduces a notion of overlapping subcategories, which we will explore in some detail. For example, a sport utility vehicle could be equally classified as a truck or a car. 45-degree viewpoint buses could be used to train both frontal and side-view bus models (Fig. 1.2a). However, global sharing may still be limited in the *one-shot learning regime*. Some rare subcategories – such as a lamborghini in profile with open wing doors – may only be observed once in a training set. In such scenarios, we introduce the notion of local sharing through “parts”. Parts are a representation that establish correspondence among certain spatial regions across training examples. For example, lamborghini may still have headlights or wheels that look similar to other common cars. Similarly, parts of the lip still look similar across different viewpoints and expressions (Fig. 1.2b).

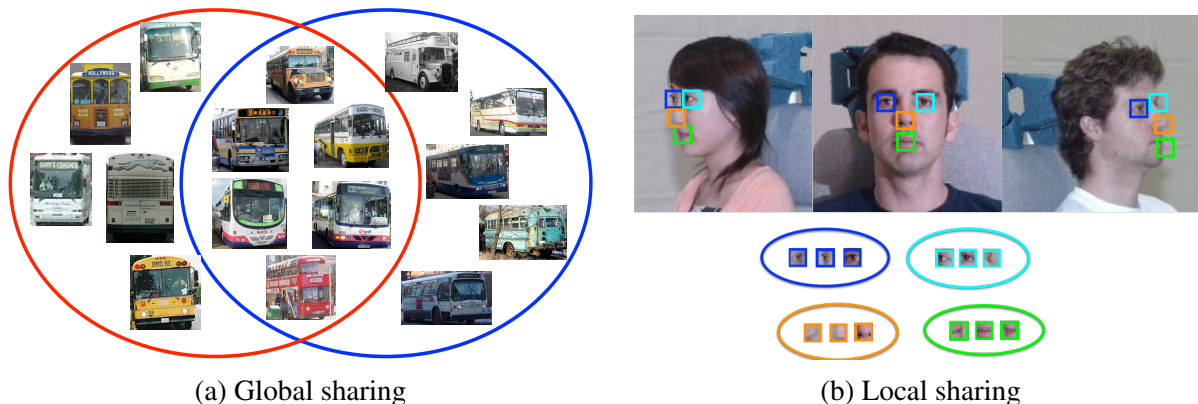


Figure 1.2: (a) Global sharing: frontal (red) and side-view (blue) buses may share a large number of $\frac{3}{4}$ -view examples. (b) A local region or “part” (cyan) is shared across faces in different viewpoints.

One important final difficulty remains. A long tail distribution suggests that there are many subcategories that are never observed in the training data. Such a learning problem is sometimes called *zero-shot learning*: how do we learn model for subcategories that have never been seen? Sharing does not appear to be quite the answer, since we do not even observe any data that can be shared. Rather, we argue that *synthesis* is a natural approach. By rearranging parts into new arrangements and combinations (not observed in the training data), one can model unseen

subcategories. Our analysis suggests that the “sharing and synthesizing” approach is crucial and effective for recognizing those unseen configurations.

From a broader perspective, an emerging idea in our community is that object detection might be solved with simple models backed with massive training sets. This work suggests a slightly refined view. Given the size of existing datasets, it appears that the current state-of-the-art will need significant additional data (perhaps exponentially larger sets) to continue producing consistent improvements in performance. We found that larger gains were possible by enforcing richer constraints within the model, often through appearance sharing and compositional representations that could make better use of data. In some sense, we need “better models” to make better use of “big data”.

Another common hypothesis is that we should focus on developing better features, not better learning algorithms. While HoG is certainly limited, we still see substantial performance gains without any change in the features themselves or the class of discriminant functions. Instead, the strategic issues appear to be parameter sharing, compositionality. Establishing and using accurate, clean correspondence among training examples (e.g., that specify that certain examples belong to the same subcategory, or that certain spatial regions correspond to the same part) and developing compositional approaches that implicitly make use of augmented training sets appear the most promising directions.

1.1 Summary of sharing approaches in this thesis

We summarize the different sharing approaches proposed and used in this thesis in Fig. 1.3 to avoid future confusion.

Exemplar models learn a template from each example, there is no sharing involved. Partitioned subcategory models as used in Chapter 2 divide training data into non-overlapping groups and

learn a template for each group. Overlapping subcategory models (introduced in Chapter 3) learn templates with overlapping groups. Part model (introduced in Chapter 4) associates local regions together as "parts" instead of sharing the entire example images.

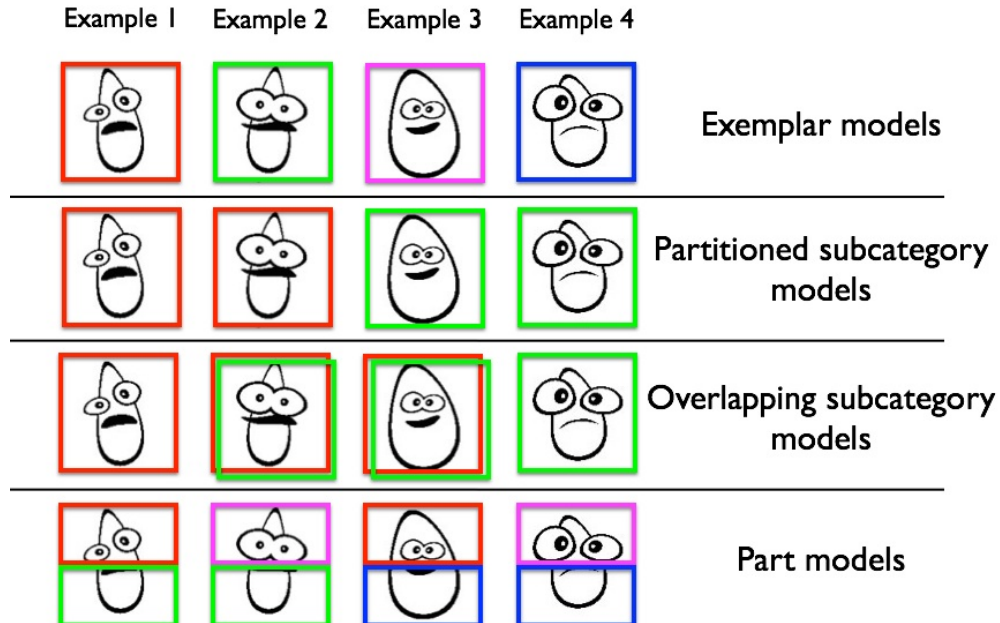


Figure 1.3: This figure explains how different models use the training data. We use a box to represent an example image, and use colors to indicate how examples are shared. Exemplar models learn a template from each example, there is no sharing involved. Partitioned subcategory models as used in Chapter 2 divide training data into non-overlapping groups and learn a template for each group. Overlapping subcategory models (introduced in Chapter 3) learn templates with overlapping groups. Part model associates local regions together as "parts" instead of sharing the entire example images. Cartoon faces are created by Matthias Dörfelt.

1.2 Thesis Overview and Contributions

The rest of this thesis is organized as follows:

Chapter 2 is an extension of our previous work [95]. It investigates the question: “Can we keep increasing object detection performance by collecting large amounts of training data while growing the complexity of mixtures/subcategory models?”

We collected and annotated a dataset that contains approximately 10 times as many training examples

per category as the well-known object detection benchmark PASCAL 2010 dataset [33] provided, allowing us to explore the potential gains of larger numbers of positive training instances.

We focus on the popular paradigm of discriminatively trained templates defined on oriented gradient features. We investigate the performance of mixtures of templates as the number of mixture components and the amount of training data grows. Surprisingly, the performance of classic mixture models appears to saturate quickly (10 templates and 100 positive training examples per template). It appears that the current state-of-the-art will need significant additional data (perhaps exponentially larger sets) to continue producing consistent improvements in performance.

Chapter 3 argues that object subcategories follow a long-tail distribution: a few subcategories are common, while many are rare. The long-tail distribution raises three challenges that complicate object detection: (1) The “right” criteria for grouping examples into subcategories is not clear. (2) Even given the optimal criteria, it is not clear how to algorithmically optimize for it. (3) Even given the optimal clustering, how does one learn models for rare subcategories (small clusters) with little training data?

We describe distributed algorithms for learning large-mixture models that capture long-tail distributions, which are hard to model with current approaches. We introduce a generalized notion of mixtures (or subcategories) that allow for examples to be shared across multiple subcategories. We optimize our models with a discriminative clustering algorithm that searches over mixtures in a distributed, brute-force fashion. We used our scalable system to train tens of thousands of deformable mixtures for VOC objects. We demonstrate significant performance improvements, particularly for object classes that are characterized by large appearance variation.

The ideas described in this chapter were first published in [93].

Chapter 4 introduces the notion of local sharing through parts. We propose part models that can represent an exponential number of subcategories by rearranging parts into new arrangements and combinations.

We also introduce several non-parametric models such as Exemplar Part Models (EPMs) and Exemplar Deformable Part Models (EDPMs) to diagnose the state of the art. We found that larger gains were possible by compositional mixtures that share template parameters via parts and that can synthesize new templates not encountered during training.

Chapter 5 is a case study of our discoveries. We build a strong model for face detection, pose estimation and landmark localization in real-world cluttered image (namely FaceDPL for the initials of the three tasks). It is based on our previous work [94], and is substantially extended with more in-depth analyses and extensive evaluation.

We group training examples into viewpoint specific subcategories, and associated local regions across subcategories together using facial landmark annotations. By assigning semantic meanings to the subcategory mixture components and parts, our model can detect faces, estimate face pose, and localize landmarks jointly.

We collected and labeled a in-the-wild face dataset with detailed annotations to benchmark our models and the state of the arts. We did extensive evaluations on both our new dataset and other two popular benchmarks: MultiPIE and FDDB. Our model compares favorably to the other methods from academia, and even beat commercial systems on all three tasks.

We also conducted in-depth analysis of the results, investigated how different factors in the model design affect the performance, which help to understand what is crucial for building good models. We also carefully scrutinize the errors, and point out the existing challenges and the next steps to move forward for research on this topic.

Related Work associated with each of these sub-problems has been incorporated inside the corresponding chapter that addresses it.

Chapter 2

Is big training data sufficient?

Datasets for training object recognition systems are steadily increasing in size. This chapter investigates the question of whether existing detectors will continue to improve as data grows, or saturate in performance due to limited model complexity and the Bayes risk associated with the feature spaces in which they operate. We focus on the popular paradigm of discriminatively trained templates defined on oriented gradient features. We investigate the performance of mixtures of templates as the number of mixture components and the amount of training data grows. Surprisingly, even with proper treatment of regularization and “outliers”, the performance of classic mixture models appears to saturate quickly (~ 10 templates and ~ 100 positive training examples per template).

2.1 Introduction

Much of the impressive progress in object detection is built on the methodologies of statistical machine learning, which make use of large training datasets to tune model parameters. Consider the benchmark results of the well-known PASCAL VOC object challenge (Fig. 2.1). There is a clear

trend of increased benchmark performance over the years as new methods have been developed. However, this improvement is also correlated with increasing amounts of training data. One might be tempted to simply view this trend as a another case of the so-called “effectiveness of big-data”, which posits that even very complex problems in artificial intelligence may be solved by simple statistical models trained on massive datasets [42]. This leads us to consider a basic question about the field: will continually increasing amounts of training data be sufficient to drive continued progress in object recognition?

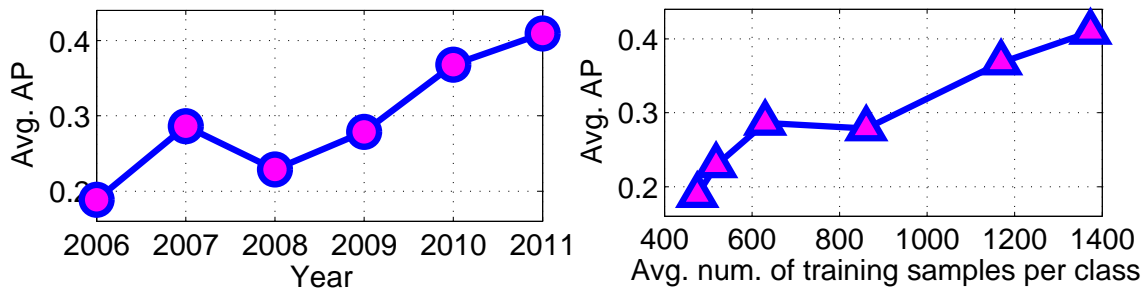


Figure 2.1: The best reported performance on PASCAL VOC challenge has shown marked increases since 2006 (left). This could be due to various factors: the dataset itself has evolved over time, the best-performing methods differ across years, etc. In the right, we plot a particular factor – training data size – which appears to correlate well with performance. This begs the question: has the increase been largely driven from the availability of larger training sets?

To tackle this question, we collected a massive training set that is an order of magnitude larger than existing collections such as PASCAL [33]. We follow the dominant paradigm of scanning-window templates trained with linear SVMs on HOG features [22, 35, 13, 56], and evaluate detection performance as a function of the amount of training data and the model complexity. We observe that the detection performance quickly encountered diminishing returns with only modest amounts of training data.

Challenges: We found there is a surprising amount of subtlety in scaling up training data sets in current systems. For a fixed model, one would expect performance to generally increase with the amount of data and eventually saturate (Fig. 2.2). Empirically, we often saw the bizarre result that off-the-shelf implementations show decreased performance with additional data! One would also expect that to take advantage of additional training data, it is necessary to grow the model

complexity, in this case by adding mixture components to capture different object sub-categories and viewpoints. However, even with growing number of mixture components, we still run into saturation quickly.

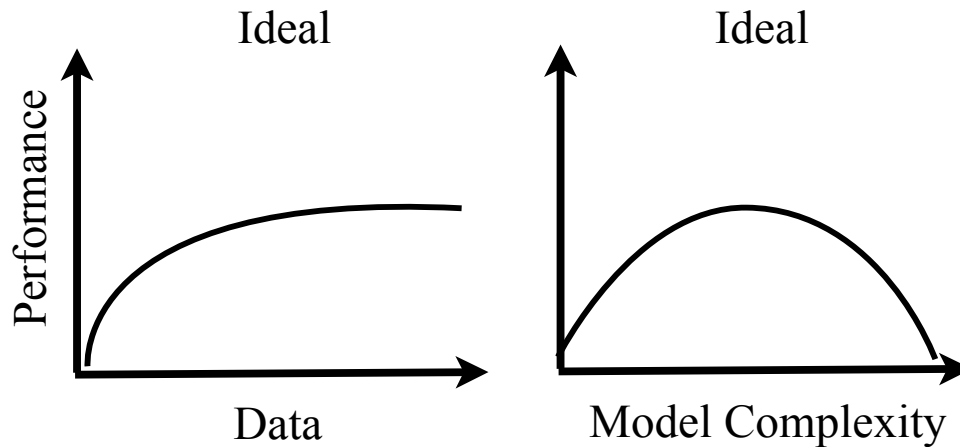


Figure 2.2: We plot idealized curves of performance versus training dataset size and model complexity. The effect of additional training examples is diminished as the training dataset grows (**left**), while we expect performance to grow with model complexity up to a point, after which an overly-flexible model overfits the training dataset (**right**). Both these notions can be made precise with learning theory bounds, see e.g. [59].

Proposed solutions: In this chapter, we offer explanations and solutions for many of these difficulties. First, we found it crucial to set model regularization as a function of training dataset using cross-validation, a standard technique which is often overlooked in current object detection systems. Second, existing strategies for discovering sub-category structure, such as clustering aspect ratios [35], appearance features [28], and keypoint labels [13] may not suffice. We found this was related to the inability of classifiers to deal with “polluted” data when mixture labels were improperly assigned. Increasing model complexity is thus only useful when mixture components capture the “right” sub-category structure.

We discuss related work in Sec. 2.2, introduce our large-scale dataset in Sec. 2.3, describe our mixture models in Sec. 2.4, present extensive experimental results in Sec. 2.5, and conclude with a discussion in Sec. 2.6.

2.2 Related Work

We view our study as complementary to other meta-analysis of the object recognition problem, such as studies of the dependence of performance on the number of object categories [26], visual properties [44], dataset collection bias [77], and component-specific analysis of recognition pipelines [63].

Our analysis is focused on template-based approaches to recognition, as such methods are currently competitive on challenging recognition problems such as PASCAL. However, it behooves us to recognize the large body of alternate approaches including hierarchical or “deep” feature learning [50], local feature analysis [79], kernel methods [80], and decision trees [12], to name a few. Such methods may produce different dependencies on performance as a function of dataset size due to inherent differences in model architectures. We hypothesize that our conclusions may generally hold for other architectures.

2.3 Big Detection Datasets

Throughout the chapter we carry out experiments using two datasets. We vary the number of positive training examples, but in all cases keep the number of negative training images fixed. We found that performance was relatively static with respect to the amount of negative training data, once a sufficiently large negative training set was used.

PASCAL-10X: Our first dataset is a newly collected data set that we refer to as PASCAL-10X and describe in detail in the following section ¹. This dataset covers the 11 PASCAL categories (see Tab. 2.1) and includes approximately 10 times as many training examples per category as the standard training data provided by the PASCAL detection challenge, allowing us to explore the

¹The dataset can be downloaded from <http://vision.ics.uci.edu/datasets/>

potential gains of larger numbers of positive training instances. We evaluate detection accuracy on the 11 PASCAL categories from the PASCAL 2010 trainval dataset (because test annotations are not public), which contains 10000+ images.

Category	PASCAL 2010		Our Data Set	
	Images	Objects	Images	Objects
Bicycle	471	614	5,027	7,401
Bus	353	498	3,405	4,919
Cat	1,005	1,132	12,204	13,998
Cow	248	464	3,194	6,909
Dining Table	415	468	3,905	5,651
Horse	425	621	4,086	6,488
Motorbike	453	611	5,674	8,666
Sheep	290	701	2,351	6,018
Sofa	406	451	4,018	5,569
Train	453	524	6,403	7,648
TV Monitor	490	683	5,053	7,808
Totals	4,609	6,167	50,772	81,075

Table 2.1: PASCAL 2010 trainval and our data set for select categories. Our data set is an order of magnitude larger.

Faces: In addition to examining performance on PASCAL object categories, we also trained models for face detection. We found faces to contain more structured appearance variation, which often allowed for more easily interpretable diagnostic experiments. Face models are trained using the CMU MultiPIE dataset[39], a well-known benchmark dataset of faces spanning multiple viewpoints, illumination conditions, and expressions. We use up to 900 faces across 13 view points. Each viewpoint was spaced 15° apart spanning 180° . 300 of the faces are frontal, while the remaining 600 are evenly distributed among the remaining viewpoints. For negatives, we use 1218 images from the INRIAPerson database [22]. Detection accuracy of face models are evaluated on the annotated face in-the-wild (AFW) [94], which contains images from real-world environments and tend to have cluttered backgrounds with large variations in both face viewpoint and appearance.

2.3.1 Collecting PASCAL-10X

In this section, we describe our procedure for building a large, annotated dataset that is as similar as possible to the PASCAL 2010 for object detection. We collected images from Flickr and annotations from Amazon Mechanical Turk (MTurk), resulting in the data set summarized in Tab. 2.1. We built training sets for 11 of the PASCAL VOC categories that are an order of magnitude larger than the VOC 2010 standard trainval set. We selected these classes as they contain the smallest amount of training examples, and so are most likely to improve from additional training data. We took care to ensure high-quality bounding box annotations and high-similarity to the PASCAL 2010 dataset. To our knowledge, this is the largest publicly available positive training set for these PASCAL categories.

Collection: We downloaded over one hundred thousand large images from Flickr to build our dataset. We took care to directly mimic the collection procedure used by the PASCAL organizers. We begin with a set of keywords (provided by the organizers) associated with each object class. For each class, we picked a random keyword, chose a random date since Flickr’s launch, selected a random page on the results, and finally took a random image from that page. We repeat this procedure until we had downloaded an order of magnitude larger number of images for each class.

Filtering: The downloaded images from Flickr did not necessarily contain objects for the category that we were targeting. We created MTurk tasks that asked workers to classify the downloaded images on whether they contained the category of interest. Our user interface in Fig. 2.3 gave workers instructions on how to handle special cases and this resulted in acceptable annotation quality without finding agreement between workers.

Annotation: After filtering the images, we created MTurk tasks instructing workers to draw bounding boxes around a specific class. Workers were only asked to annotate up to five objects per image using our interface as in Fig. 2.3, although many workers gave us more boxes. On average, our system received annotations at three images per second, allowing us to build bounding boxes

for 10,000 images in under an hour. As not every object is labeled, our data set cannot be used to perform detection benchmarking (it is not possible to distinguish false-positives from true-negatives). We experimented with additional validation steps, but found they were not necessary to obtain high-quality annotations.

2.3.2 Data Quality

To verify the quality of our annotations, we performed an in-depth diagnostic analysis of a particular category (horses). Overall, our analysis suggests that our collection and annotation pipeline produces high-quality training data that is similar to PASCAL.

Attribute distribution: We first compared various distributions of attributes of bounding boxes from PASCAL-10X to those from both PASCAL 2010 and 2007 trainval. Attribute annotations were provided by manual labeling. Our findings are summarized in Tab. 2.2. Interestingly, horses collected in 2010 and 2007 vary significantly, while 2010 and PASCAL-10X match fairly well. Our images were on average twice the resolution as those in PASCAL so we scaled our images down to construct our final dataset.

Attributes	Us	PASCAL	
		2010	2007
Truncated	30.8	31.5	15.8
Occluded	5.9	8.6	7.1
Jumping	4.0	4.3	15.8
Standing	69.9	68.8	54.6
Trotting	23.5	24.9	26.6
Sitting	2.0	1.4	0.7
Other	0.0	0.5	0
Person Top	24.8	29.1	57.5
Person Besides	8.8	10.0	8.6
No Person	66.0	59.8	33.8


Table 2.2: Frequencies of attributes (percent) across images in our 10x horse data set compared to the PASCAL 2010 train-val data set. Bolded entries highlight significant differences relative to our collected data. Our dataset has similar attribute distribution to the PASCAL 2010, but differs significantly from 2007, which has many more sporting events.

When to mark yes:	When to mark no:
<ul style="list-style-type: none"> • if the image contains at least one horse. • if the horses are clearly visible through glass. • if the horses are in a mirror. • if the horses are in poor lighting, but still visible. • if the image has a picture of horses as long as they are realistic. 	<ul style="list-style-type: none"> • if all the horses are toys or photoshopped. • if the image is about horses but does not contain a horse. • if every horse is very tiny. • if the image is taken inside a horse. • if the image is poor quality or has bad motion blur. • if the image is a collage or multiple images. • if you don't know what the image contains.

The images are below (they may take a second to load):

Yes, there is at least one horse.

No, there are no horses.



- Draw a box around each individual **horse** in this image.
- If there are more than 5 **horses**, then label the 5 largest.
- You must [read the instructions and examples](#) as we **hand review all work**.

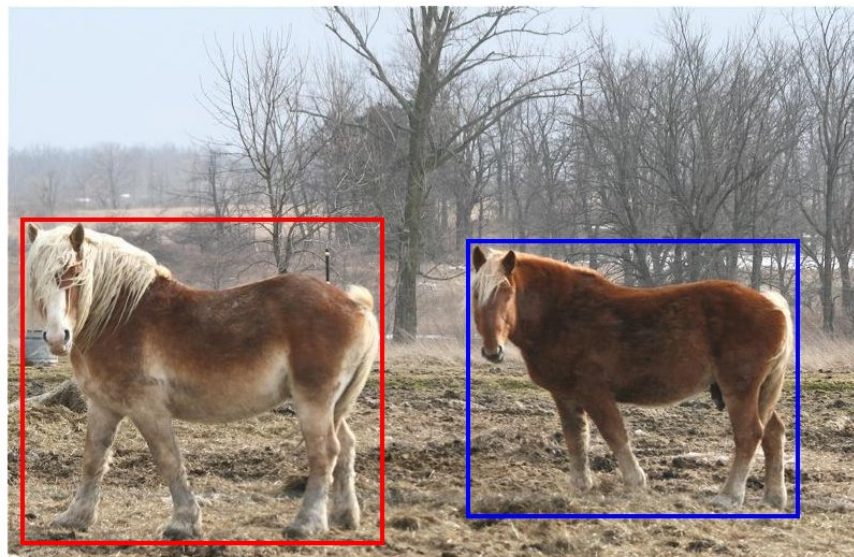


Image does not contain **horses**.

Your work will directly impact active research.

Figure 2.3: Our MTurk user interfaces for image classification and object annotation. We provided detailed instructions to workers, resulting in acceptable annotation quality.

User assessment: We also gauged the quality of our bounding boxes compared to PASCAL with a user study. We flashed a pair of horse bounding boxes, one from PASCAL-10X and one from PASCAL 2010, on a screen and instructed a subject to label which appeared to be better example. Our subject preferred the PASCAL 2010 data set 49% of the time and our data set 51% of the time. Since chance is 50%-50% and our subject operated close to chance, this further suggests PASCAL-10X matched well with PASCAL. Qualitatively, the biggest difference observed between the two datasets was that PASCAL-10X bounding boxes tend to be somewhat “looser” than the (hand curated) PASCAL 2010 data.

Redundant annotations: We tested the use of multiple annotations for removing poorly labeled positive examples. All horse images were labeled twice, and only those bounding boxes that agreed across the two annotation sessions were kept for training. We found that training on these cross-verified annotations did not significantly affect the performance of the learned detector.

2.4 Mixture models

To take full advantage of additional training data, it is vital to grow model complexity. We accomplish this by adding a mixture component to capture additional “sub-category” structure. Our basic building block will be a mixture of linear classifiers, or templates. Formally speaking, we compute the detection score of an image window I as:

$$S(I) = \max_m \left[w_m \cdot \phi(I) + b_m \right] \quad (2.1)$$

where m is a discrete mixture variable, $\Phi(I)$ is a HOG image descriptor [22], w_m is a linearly-scored template, and b_m is an (optional) bias parameter that acts as a prior that favors particular templates over others.

In this section, we describe approaches for learning mixture models by clustering positive examples

from our training set. We train independent linear classifiers (w_m, b_m) using positive examples from each cluster. One difficulty in evaluating mixture models is that fluctuations in the (non-convex) clustering results may mask variations in performance we wish to measure. We took care to devise a procedure for varying K (the number of clusters) and N (the amount of training data) in such a manner that would reduce stochastic effects of random sampling.

Unsupervised clustering: For our unsupervised baseline, we cluster the positive training images of each category into 16 clusters using hierarchical k-means, recursively splitting each cluster into $k = 2$ subclusters. For example, given a fixed training set, we would like the cluster partitions for $K = 8$ to respect the cluster partition of $K = 4$. To capture both appearance and shape when clustering, we warp an instance to a canonical aspect ratio, compute its HOG descriptor (reduce the dimensionality with PCA for computational efficiency), and append the aspect ratio to the resulting feature vector.

Partitioned sampling: Given a fixed training set of N_{max} positive images, we would like to construct a smaller sampled subset, say of $N = \frac{N_{max}}{2}$ images, whose cluster partitions respect those in the full dataset. This is similar in spirit to stratified sampling and attempts to reduce variance in our performance estimates due to “binning artifacts” of inconsistent cluster partitions across re-samplings of the data.

To do this, we first hierarchically-partition the full set of N_{max} images by recursively applying k-means. We then subsample the images in the leaf nodes of the hierarchy in order to generate a smaller hierarchically partitioned dataset by using the same hierarchical tree defined over the original leaf clusters. This sub-sampling procedure can be applied repeatedly to produce training datasets with fewer and fewer examples that still respects the original data distribution and clustering.

The sampling algorithm, shown in Algorithm 1, yields a set of partitioned training sets, indexed by (K, N) with two properties: (1) for a fixed number of clusters K , each smaller training set is a subset of the larger ones, and (2) given a fixed training set size N , small clusters are strict

<p>Input: $\{N_n\}; \{S^{(i)}\}$ Output: $\{C_n^{(i)}\}$</p> <pre> 1 $C_0^{(i)} = S^{(i)}, C_n^{(i)} = \emptyset \quad \forall i, \forall n > 1$ 2 for $n = 1 : end$ do // For each N_n 3 for $t = 1 : N_n$ do 4 $z \sim \frac{ C_{n-1}^{(z)} }{\sum_j C_{n-1}^{(j)} };$ // Pick a cluster randomly 5 $C_n^{(z)} \Leftarrow C_{n-1}^{(z)};$ // sample zth cluster without replacement 6 end 7 end </pre>
--

Algorithm 1: Partitioned sampling of the clusters. N_n is the number of samples to return for set n with $N_0 = N_{max}$; $N_n > N_{n+1}$. $S^{(i)}$ is the i^{th} cluster from the lowest level of the hierarchy (e.g., with $K = 16$ clusters) computed on the full dataset N_{max} . Steps 4-5 randomly sample N_n training samples from $\{C_{n-1}^{(i)}\}$ to construct K sub-sampled clusters $\{C_n^{(i)}\}$, each of which contain a subset of the training data while keeping the same distribution of the data over clusters.

refinements of larger clusters. We compute confidence intervals in our experiments by repeating this procedure multiple times to resample the dataset and produce multiple sets of (K, N) -consistent partitions.

Supervised clustering: To examine the effect of supervision, we cluster the training data by manually grouping visually similar samples. For CMU MultiPIE, we define clusters using viewpoint annotations provided with the dataset. We generate a hierarchical clustering by having a human operator merge similar viewpoints, following the partitioned sampling scheme above. Since PASCAL-10X does not have viewpoint labels, we generate an “over-clustering” with k-means with a large K , and have a human operator manually merge clusters. Fig. 2.4 and Fig. 2.5 show example clusters for faces and buses.

2.5 Experiments

Armed with our array of non-parametric mixture models and datasets, we now present an extensive diagnostic analysis on 11 PASCAL categories from the 2010 PASCAL trainval set and faces from



(a) Unsupervised

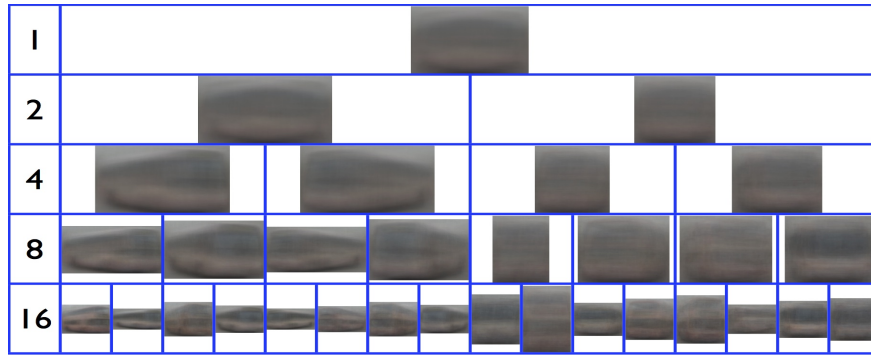


(b) Supervised

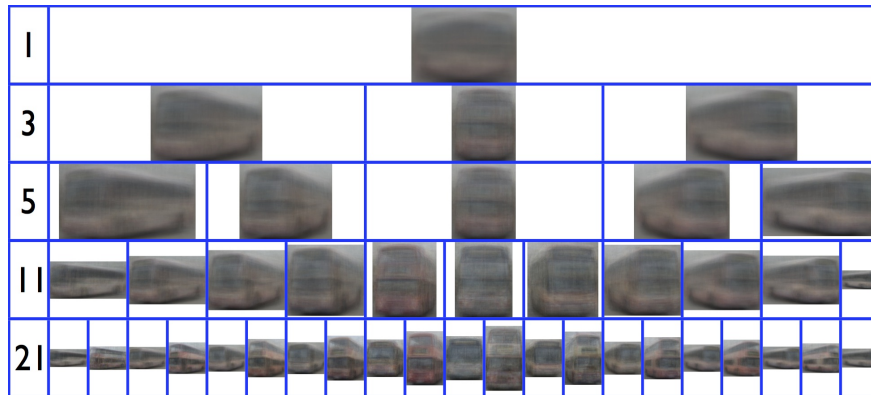
Figure 2.4: We compare supervised versus automatic (k-means) approaches for clustering by displaying the average RGB image of each cluster. The supervised methods use viewpoint labels to cluster the training data. Because our face data is relatively clean, both obtain reasonably good clusters. However, at some levels of the hierarchy, unsupervised clustering does seem to produce suboptimal partitions - for example, at $K = 2$. There is no natural way to group multi-view faces into two groups. Automatically selecting K is a key difficulty with unsupervised clustering algorithms.

the Annotated Faces in the Wild test set [94]. For each category, we train the model with varying number of samples (N) and mixtures (K). To train our mixture models, we learn rigid HOG templates [22] with linear SVMs [15]. We calibrated SVM scores using Platt scaling [66]. Since the goal is to calibrate scores of mixture components relative to each other, we found it sufficient to train scaling parameters using the original training set rather than using a held-out validation set.

To show the uncertainty of the performance with respect to different sets of training samples, we randomly re-sample the training data 5 times for each N and K following the partitioned sampling



(a) Unsupervised

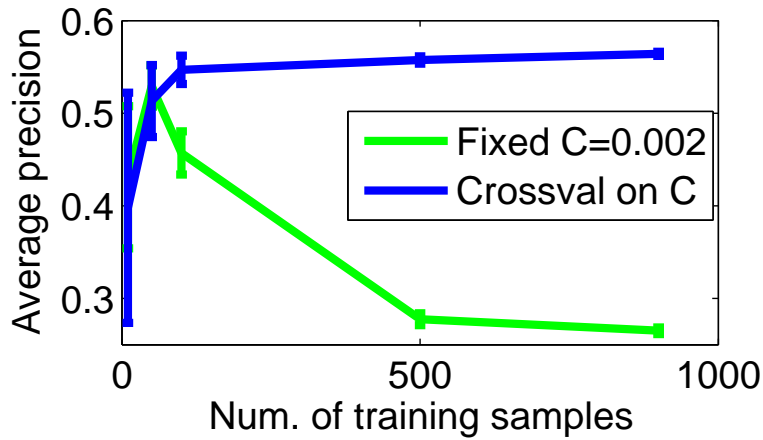


(b) Supervised

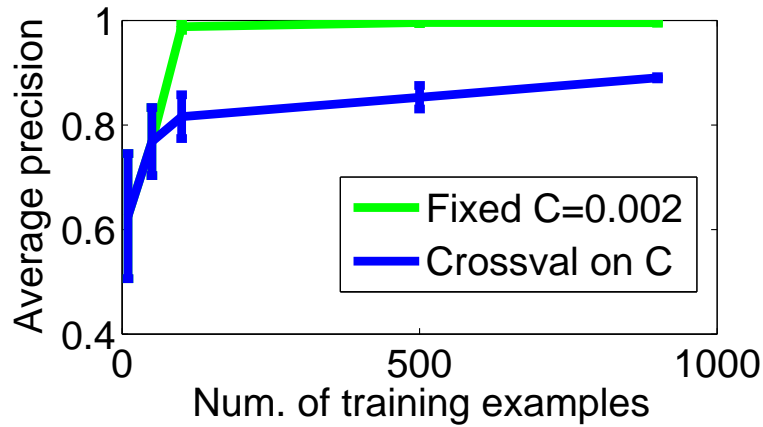
Figure 2.5: We compare supervised versus automatic (k-means) approaches for clustering images of PASCAL buses. Supervised clustering produces more clear clusters, e.g. the 21 supervised clusters correspond to viewpoints and object type (single vs double-decker). Supervised clusters perform better in practice, as we show in Fig. 2.9.

scheme described in Sec. 2.4. The best regularization parameter C for the SVM was selected by cross validation. For diagnostic analysis, we first focus on faces and buses.

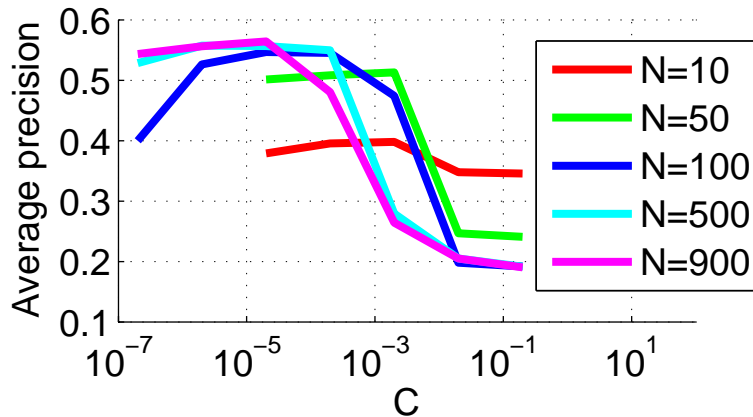
Evaluation: We adopt the PASCAL VOC precision-recall protocol for object detection (requiring 50% overlap), and report average precision (AP). While learning theory often focuses on analyzing 0-1 classification error rather than AP [59], we experimentally verified that AP typically tracks 0-1 classification error and so focus on the former in our experiments.



(a) Single template performance on the testing data



(b) Single template performance on the training data



(c) Single template performance on the testing data w/ varying C

Figure 2.6: (a) More training data could hurt if we did not cross-validate to select the optimal C. (b) Training error, when measured on a fixed training set of 900 faces and 1218 negative images, always decreases as we train with more of those images. This further suggests that overfitting is the culprit, and that proper regularization is the solution. (c) Test performance can change drastically with C. Importantly, the optimal setting of C depends on the amount of positive training examples N .

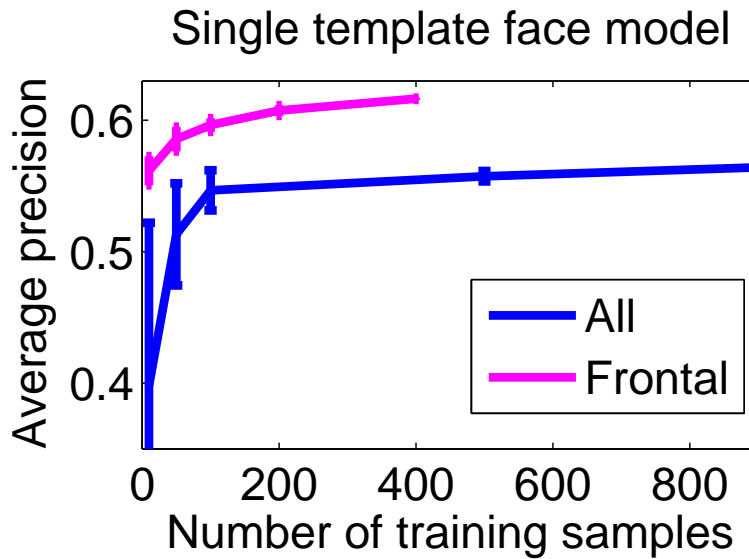
2.5.1 The importance of proper regularization

We begin with a rather simple experiment: how does a single rigid HOG template tuned for faces perform when we give it more training data N ? Fig. 2.6 shows the surprising result that additional training data can *decrease* performance! For imbalanced object detection datasets with many more negatives than positives, the hinge loss appears to grow linearly with the amount of positive training data; if one doubles the number of positives, the total hinge loss also doubles. This leads to overfitting. To address this problem, we found it *crucial to cross-validate C* across different N . By doing so, we do see better performance with more data (Fig. 2.6a). While cross-validating regularization parameters is a standard procedure when applying a classifier to a new dataset, most off-the-shelf detectors are trained using a fixed C across object categories with large variations in the number of positives. We suspect other systems based on standard detectors [35, 22] may also be suffering from suboptimal regularization and might show an improvement by proper cross-validation.

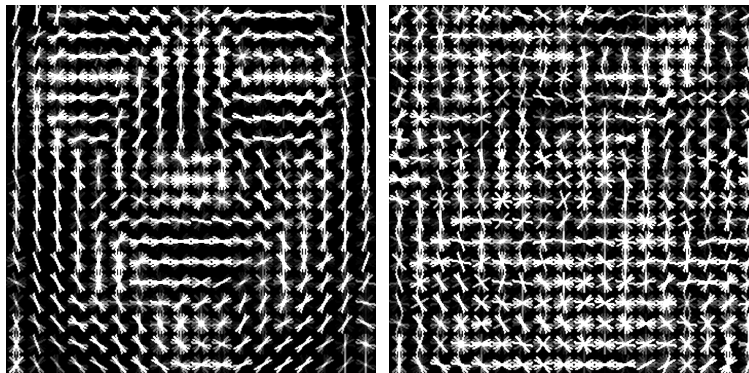
2.5.2 The importance of clean training data

Although proper regularization parameters proved to be crucial, we still discovered scenarios where additional training data hurt performance. Fig. 2.7 shows an experiment with a fixed set of N training examples where we train two detectors: (1) *All* is trained with with all N examples, while (2) *Frontal* is trained with a smaller, “clean” subset of examples containing frontal faces. We cross-validate C for each model for each N . Surprisingly, *Frontal* outperforms *All* even though it is trained with less data.

This outcome cannot be explained by a failure of the model to generalize from training to test data. We examined the training loss for both models, evaluated on the full training set. As expected, *All* has a lower SVM objective function than *Frontal* (1.29 vs 3.48). But in terms of 0-1 loss, *All* makes nearly twice as many classification errors on the same training images (900 vs 470).



(a)



(b) Frontal

(c) All

Figure 2.7: In (a), we compare the performance of a single HOG template trained with N multi-view face examples, versus a template trained with a subset of those N examples corresponding to frontal faces. The frontal-face template (b) looks “cleaner” and makes fewer classification errors on both testing and training data. The fully-trained template (c) looks noisy and performs worse, even though it produces a lower SVM objective value (when both (b) and (c) are evaluated on the full training set). This suggests that SVMs are sensitive to noise and benefit from training with “clean” data.

This observation suggests that *the hinge loss is a poor surrogate to the 0-1 loss* because “noisy” hard examples can wildly distort the decision boundary as they incur a large, unbounded hinge penalty. Interestingly, latent mixture models can mimic the behavior of non-convex bounded loss functions [84] by placing noisy examples into junk clusters that simply serve to explain outliers in the training set. In some cases, a single “clean” mixture component by itself explains most of the

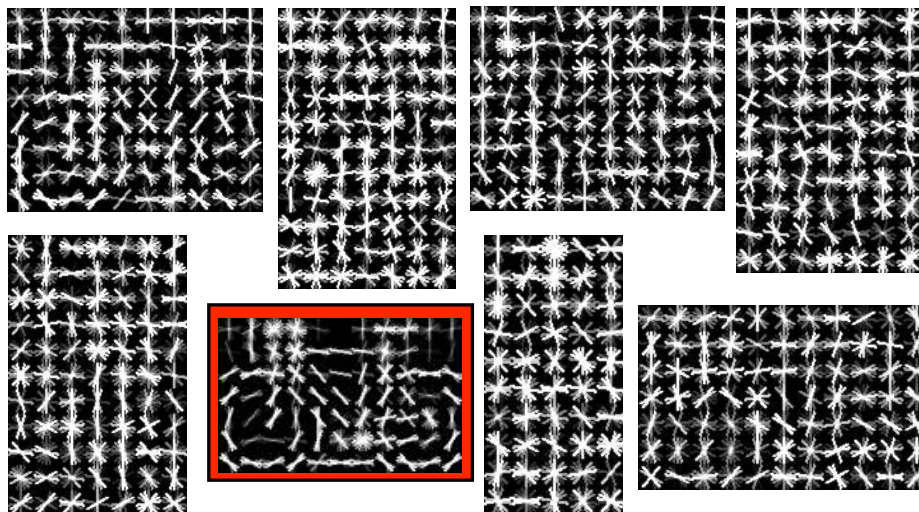


Figure 2.8: The single bicycle template (marked with red) alone achieves $ap=29.4\%$, which is almost equivalent to the performance of using all 8 mixtures ($ap=29.7\%$). Both models strongly outperform a single-mixture model trained on the full training set. This suggests that these additional mixtures are useful during training to capture outliers and prevent “noisy” data from polluting a “clean” template that does most of the work at test time.

test performance (Fig. 2.8).

The importance of “clean” training data suggests it could be fruitful to correctly cluster training data into mixture components where each component is “clean”. We evaluated the effectiveness of providing fully supervised clustering in producing clean mixtures. In Fig. 2.9, we see a small 2% to 5% increase for manual clustering. In general, we find that unsupervised clustering can work reasonably well but depends strongly on the category and features used. For example, the DPM implementation of [35] initializes mixtures based on aspect ratios. Since faces in different viewpoint share similar aspect ratios, this tends to produce “unclean” mixtures compared to our non-latent clustering.

2.5.3 Performance of mixture models

Given the right regularization and clean mixtures trained independently, we now evaluate whether performance asymptotes as the amount of training data and the model complexity increase.

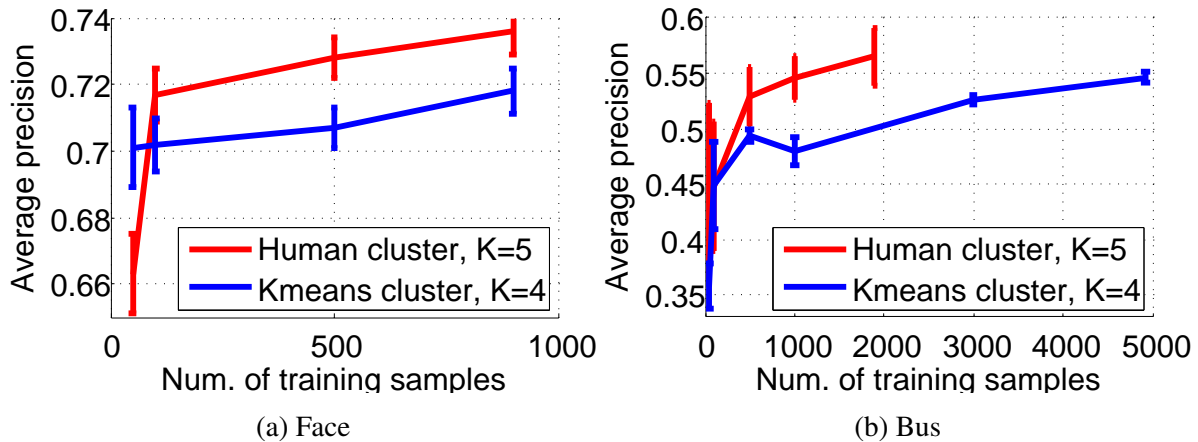


Figure 2.9: We compare the human clustering and automatic k-means clustering at near-identical K . We find that supervised clustering provides a small but noticeable improvement of 2-5%.

Fig. 2.10 shows performance as we vary K and N after cross-validating C and using supervised clustering. Fig. 2.10a demonstrates that increasing the amount of training data yields a clear improvement in performance at the beginning, and the gain quickly becomes smaller later. Larger models with more mixtures tend to perform worse with fewer examples due to over fitting, but eventually win with more data. Surprisingly, improvement tends to saturate at ~ 100 training examples per mixture and with ~ 10 mixtures. Fig. 2.10b shows performance as we vary model complexity for a fixed amount of training data. Particularly at small data regimes, we see the critical point one would expect from Fig. 2.2: a more complex model performs better up to a point, after which it overfits. We found similar behavior for the buses category which we manually clustered by viewpoint.

We performed similar experiments for all 11 PASCAL object categories in our PASCAL-10X dataset shown in Fig. 2.11. We evaluate performance on the PASCAL 2010 trainval set since the testset annotations are not public. We cluster the training data into $K=[1,2,4,8,16]$ mixture components, and $N=[50, 100, 500, 1000, 3000, N_{max}]$ training samples, where N_{max} is the number of training samples collected for the given category. For each N , we select the best C and K through cross-validation. Fig. 2.11a, appears to suggest that performance is saturating across all categories as we increase the amount of training data. However, if we plot performance on a log

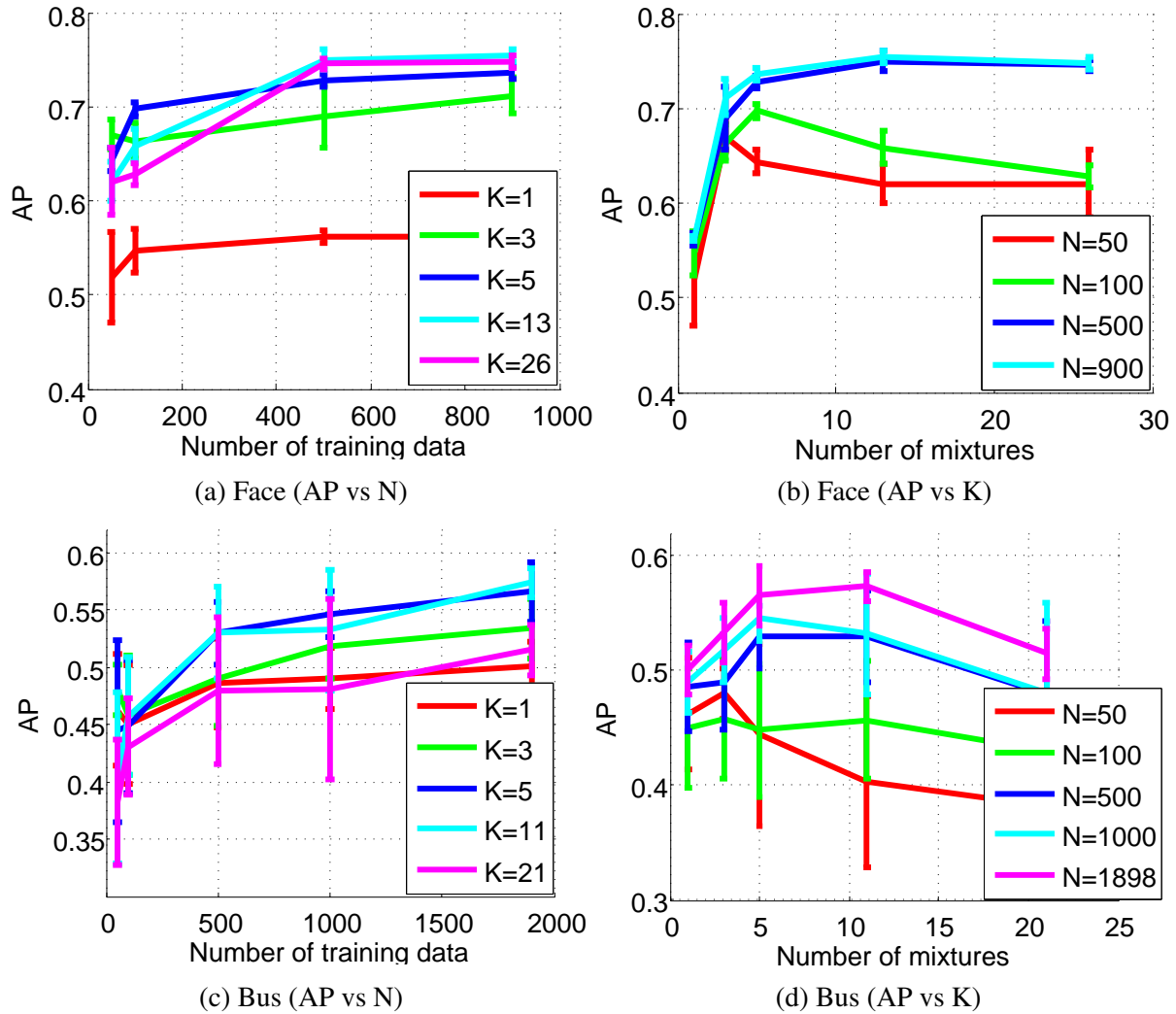
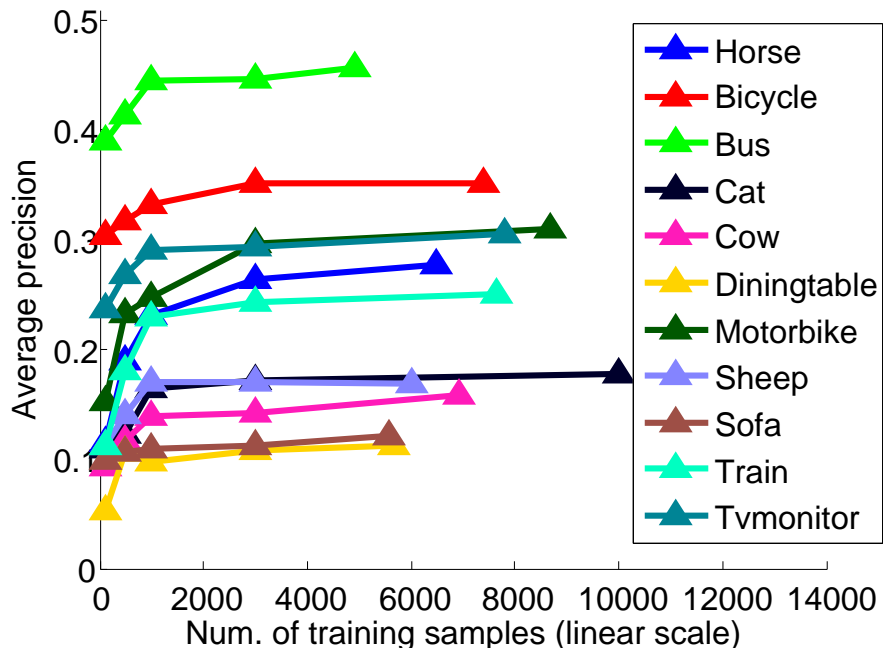
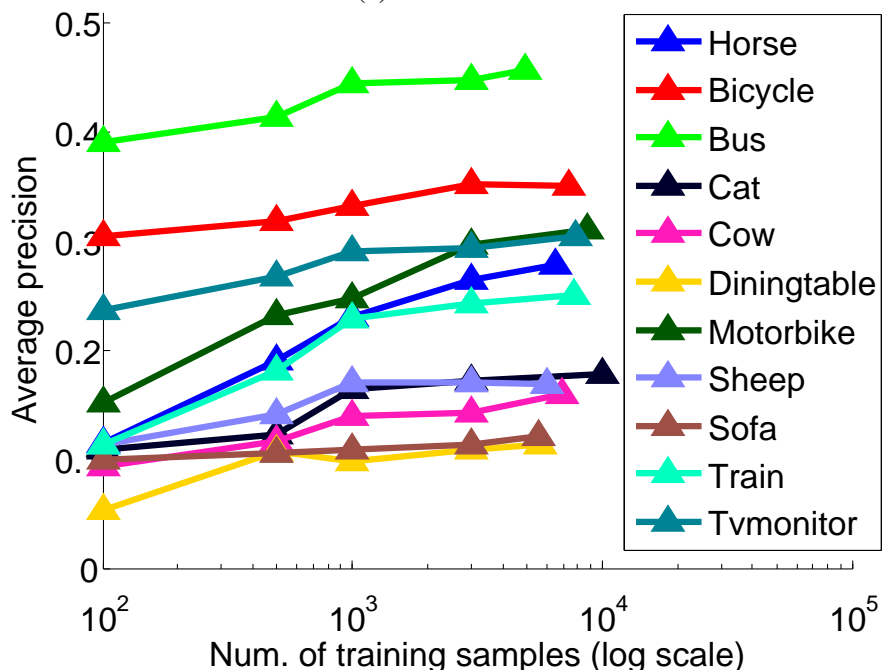


Figure 2.10: (a)(c) show the monotonic non-decreasing curves when we add more training data. The performance saturates quickly at a few hundred training samples. (b)(d) show how the performance changes with more mixtures K . Given a fixed number of training samples N , the performance increases at the beginning, and decreases when we split the training data too much so that each mixture only has few samples.

scale (Fig. 2.11b), it appears to increase roughly linearly. This suggests that the required training data may need to grow exponentially to produce a fixed improvement in accuracy. For example, if we extrapolate the steepest curve in Fig. 2.11b (motorbike), we will need 10^{12} motorbike samples to reach 90% AP!



(a) Linear scale



(b) Log scale

Figure 2.11: We plot the best performance at varying amount of training data for 11 PASCAL categories on PASCAL 2010 trainval set. (a) shows that all the curves look saturated with a relatively small amount of training data; but in log scale (b) suggests a diminishing return instead of true saturation. However the performance increases so slow that we will need more than 10^{12} examples per category to reach 90% AP if we keep growing at the same rate.

2.6 Conclusion

We have performed an extensive analysis of the current dominant paradigm for object detection using HOG feature templates. We specifically focused on performance as a function of the amount of training data.

To scale current systems to larger datasets, we find that one must get certain “details” correct. Specifically, (a) cross-validation of regularization parameters is mundane but crucial, (b) current discriminative classification machinery is overly sensitive to noisy data, suggesting that (c) manual cleanup and supervision or more clever latent optimization during learning may play an important role for designing high-performance detection systems. We also demonstrate that HOG templates have a relatively small effective capacity; one can train accurate HOG templates with 100-200 positive examples (rather than thousands of examples as is typically done [22]).

From a broader perspective, an emerging idea in our community is that object detection might be solved with simple models backed with massive training sets. Our experiments suggest a slightly refined view. Given the size of existing datasets, it appears that the current state-of-the-art will need significant additional data (perhaps exponentially larger sets) to continue producing consistent improvements in performance.

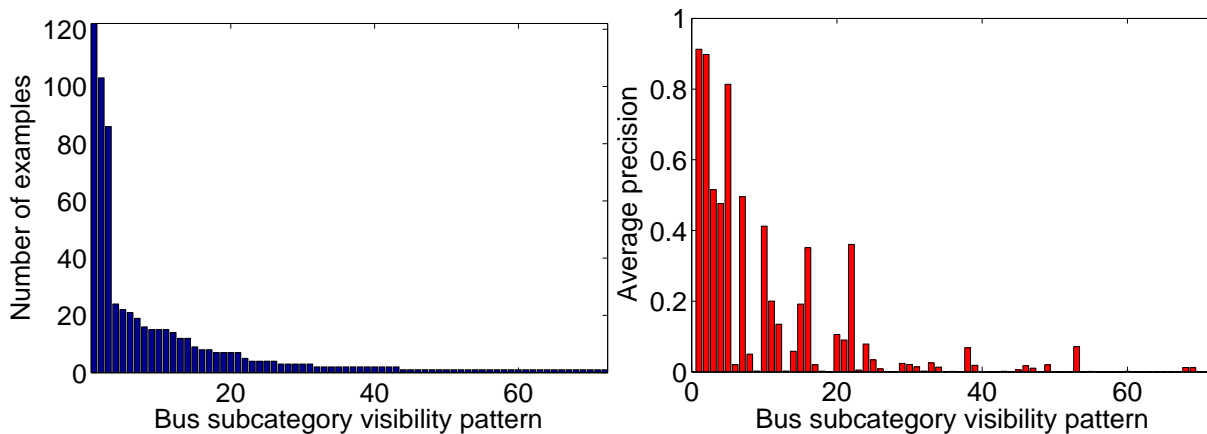
Chapter 3

Capturing long-tail distributions of object subcategories

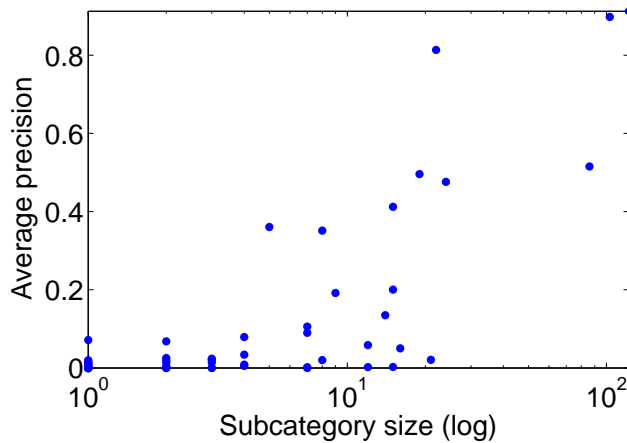
In the last chapter, we empirically examined the performance of subcategory models as one increases the amount of training data, and found that performance saturated fairly quickly. If the performance increases at the rate we show in Fig. 2.11, we will need more than 10^{12} training examples per category to reach 90% AP. Collecting such a large training set for each object category is clearly infeasible. This motivates us to ask two questions: (1) Why does the current model not work well with a modest amount of training data? (2) Is it possible to boost the detection performance without collecting an enormous amount of data?

We see the answer to the first question as a consequence of the “long-tail” distribution of object subcategories (Fig. 3.1a): instances within an object category exhibits large appearance variation, which follow long-tail distributions. There are a small number of common cases, and a large number of unusual ones which collectively make up a large portion of the data. A particular subcategory from the long tail may be difficult to observe in any finite training dataset. It is therefore hard to model due to lack of training data. Fig. 3.1b shows that the current detector actually does well on the

common subcategories, yielding average precision of above 90% for the two largest subcategories, but generally does much worse on rare cases.



(a) Distributions of the visibility patterns for bus (b) Detection performance on each subcategory



(c) Scatter plot of subcategory size vs average precision

Figure 3.1: (a) shows the distribution of the keypoint visibility patterns for bus from PASCAL (using the manual annotations of [13]), which follows a long-tail. How to strictly define subcategory and how to group examples into subcategories are still open questions. Visibility patterns of keypoints are used as a proxy for general appearance in generating this figure, as they represent the variations due to viewpoints and occlusions. In this chapter, we describe an approach to discover the subcategories, rather than pre-define it. We empirically show that the discovered visual subcategories follow a long-tail. (b) shows the detection performance for each subcategory. The current detector tends to do well on the common ones (reaching above 90% on the top two), and work poorly on the rare subcategories presumably due to lack to training data. (c) Merges (a)(b) into a scatter plot.

In this chapter, We propose the answer to the second question as to augment the data from the tail, as this will probably give us a boost on the performance on the rare subcategories. We describe distributed algorithms for learning mixture models that capture long-tail distributions, which are hard

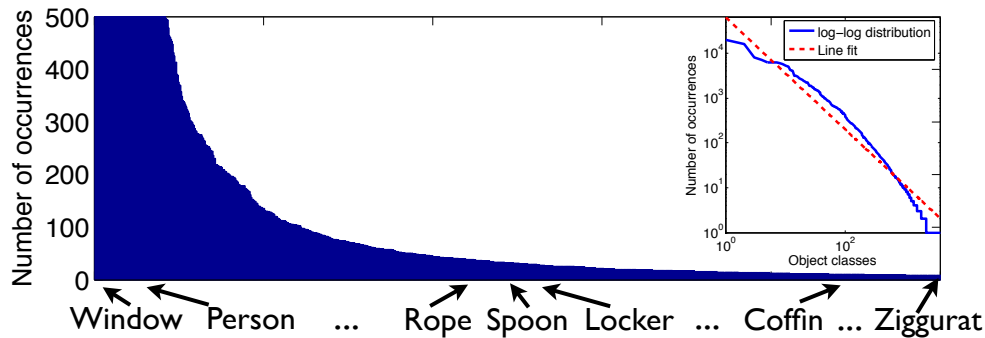
to model with current approaches. We introduce a generalized notion of mixtures (or subcategories) that allow for examples to be shared across multiple subcategories. We optimize our models with a discriminative clustering algorithm that searches over mixtures in a distributed, “brute-force” fashion. We used our scalable system to train tens of thousands of deformable mixtures for VOC objects. We demonstrate significant performance improvements, particularly for object classes that are characterized by large appearance variation.

3.1 Long-tail and its challenges

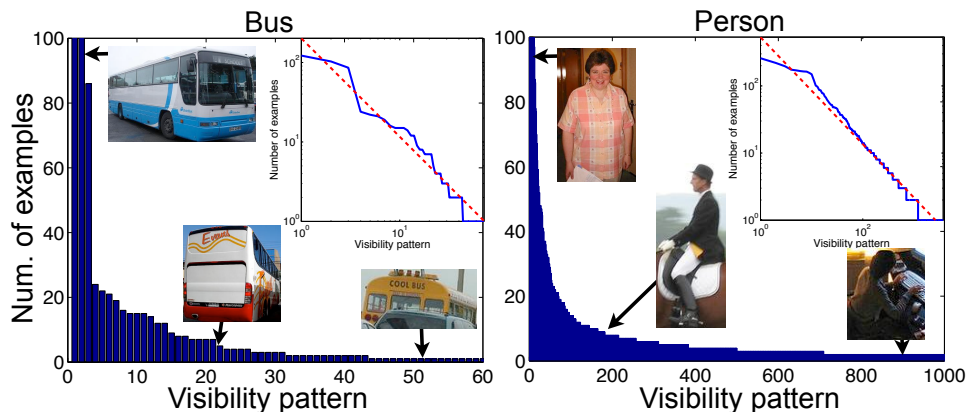
It is well-known that the frequency of object occurrence in natural scenes follows a long-tail distribution [71]: for example, people and windows are much more common than coffins and ziggurats (Fig. 3.2a). Long-tails complicate analysis because rare cases from the tail still collectively make up a significant portion of the data and so cannot be ignored. Many approaches try to minimize this phenomenon by working with balanced datasets of objects categories [27]. But long-tails still exist for object *subcategories*: most people tend to stand, but people can assume a large number of unusual poses (Fig. 3.2b). We believe that current approaches may capture iconic object appearances well, but are still limited due to inadequate modeling of the tail.

In theory, multi-mixture or subcategory models should address this, with possibly large computational costs: train a separate model for different viewpoints, shape deformation, etc. Empirically though, these approaches tend to saturate early in performance after a modest number of mixtures [95, 56, 29, 40, 35].

We argue that the long-tail raises three major challenges that current mixture models do not fully address: (1) The “right” criteria for grouping examples into subcategories is not clear. Various approaches have been suggested (including visual similarity [29], geometric similarity [9], semantic ontologies [27]), but the optimal criteria remains unknown. (2) Even given the optimal criteria, it is



(a) The number of examples by object class in SUN dataset



(b) Distributions of the visibility patterns for bus and person

Figure 3.2: Long tail distributions exist for both object categories and subcategories. (a) shows the number of examples by object class in the SUN dataset. The blue curve in the inset show a log-log plot, along with a best-fit line in red. This suggests that the distribution follows a long-tail power law. (b) shows the distributions of the keypoint visibility patterns for bus and person from PASCAL (using the manual annotations of [13]), which also follow a long-tail. We describe methods for automatically discovering long-tail distributions of subcategories with a distributed, “brute-force” search without using additional annotations.

not clear how to algorithmically optimize for it. Typical methods employ some form of clustering, but common algorithms (e.g., k-means) tend to report clusters of balanced sizes, while we hope to get long-tail distributions. (3) Even given the optimal clustering, how does one learn models for rare subcategories (small clusters) with little training data?

In our work, we address all three challenges: (1) We posit that the optimal grouping criteria is simply **recognition accuracy**. But this presumably requires a “brute-force” search over all possible clusterings, and an evaluation of the recognition accuracy of each grouping, which appears hopeless.

(2) We introduce a **discriminative clustering** algorithm that accomplishes this through distributed computation, making use of massively-parallel architectures. We show that long-tail cluster sizes *naturally emerge* from our algorithm. (3) To address the lack of training data for small clusters, we allow rare subcategories to share training examples with dominant ones, introducing a notion of a **overlapping subcategories**. Such fluid definitions of categories are common in psychology [69]. For example, a sport utility vehicle could be equally classified as a truck or a car. Overlapping subcategories allow for cluster label assignment to decouple across subcategories, crucial for our distributed optimization.

Noteably, our clustering algorithm does not explicitly enforce long-tail distributions as priors. Rather, our underlying hypothesis is that long-tail distributions are an *emergent property* of the “optimal” clustering, when measured with respect to recognition accuracy. We verify this hypothesis experimentally. It is possible that brute-force clustering of other data types with respect to other criteria may not produce long-tails. Rather, our experimental results reflect an empirical property of our visual world, much like the empirical analysis of Fig. 3.2.

We review related work in Sec. 3.2, introduce our generalized subcategory model and discriminative optimization algorithm in Sec. 3.3, and present results in Sec. 3.4. We demonstrate that our long-tail mixture-models significantly outperform prior work on benchmark detection datasets, in some cases achieving a factor of 2 improvement.

3.2 Related work

Subcategory discovery: Estimating subcategories is surprisingly hard; clustering based on keypoints [40, 9] and appearance [28, 7] have provided only modest performance increases [95]. [30] uses combined appearance, shape, and context information to discover a small number of common subcategories for object classification, however the rare cases are thrown away as “outliers”. One

attractive approach is to use a *discriminative* model to re-rank and identify other nearby training examples for clustering. This is often implemented through latent mixture assignment in a max-margin model [35] or discriminative k-means [85]. In practice, such methods are sensitive to initialization and can suffer from miscalibration [74]. We describe a discriminative clustering algorithm that searches over all initializations in a distributed fashion without ever comparing scores across different models during training. Finally, our models allow for overlapping clusters. This differs from soft assignment in that the total contribution of an example need not be 1; indeed, we show that certain examples are much more dominant than others (consistent with Rosch’s prototype theory [68]).

Sharing across categories: There has been much work on sharing information between object category models [71, 8, 54]. Most related is [54], which allows an object class to borrow examples from similar categories, e.g. some armchairs can be used to train sofa models. While this approach yields modest performance gains (1.4%AP), we produce larger gains presumably due to our brute-force optimization over subcategory labels and sharing. Another attractive formalism is that of attributes, or generic properties shared across object categories [52, 38, 82]. One could interpret binary subcategory labels as a binary attribute vector; indeed, we perform multi-dimensional scaling on such a vector to generate the visualization in Fig. 3.3. Our approach differs from much past work in that our “attributes” are latently inferred in a data-driven manner.

Sharing across subcategories: Various approaches have also explored sharing across subcategories. For example, it is common to share local features across view-based mixtures of an object [78, 94]. Typically, subcategory mixtures are supervised, but not always [62]. We share global examples rather than local parts, as the former is more amenable to brute-force distributed optimization.

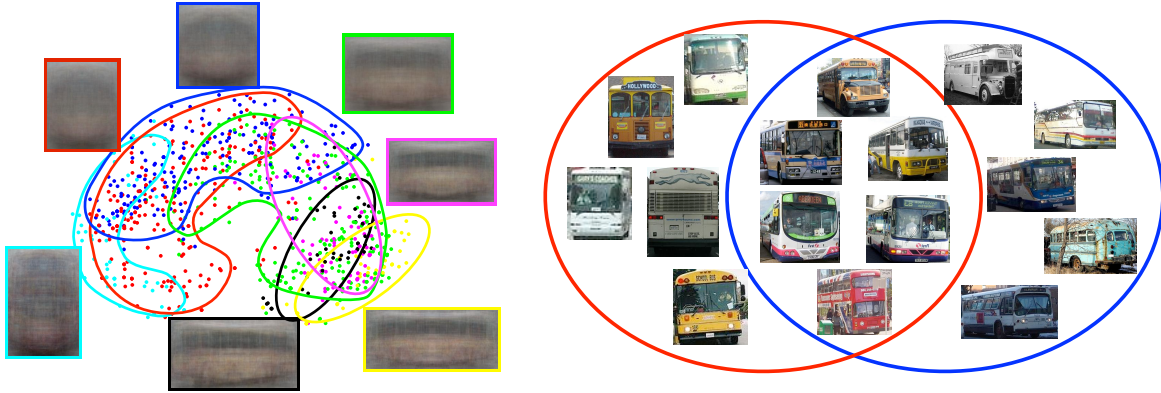


Figure 3.3: We describe overlapping subcategory models that allow for training data to belong to multiple clusters with a large variation in size. For example, frontal (red) and side-view (blue) buses may share a large number of $\frac{3}{4}$ -view examples, and both are much more common than multi-body articulated buses (yellow). We show that such models better characterize objects with “long-tail” appearance distributions.

3.3 Learning long-tail subcategory models

In this section, we describe our approach for learning long-tail subcategory models. We model each subcategory with a single-mixture deformable part model (DPM) [35]. Our overall pipeline is summarized in Fig. 3.4. We explain each step in detail in the following.

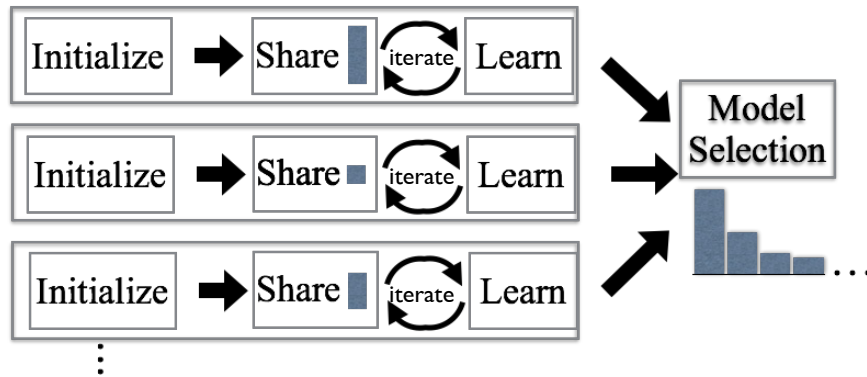


Figure 3.4: Our overall pipeline. We learn a massive number of candidate subcategory models in parallel, each *initialized* with its own training example (an exemplar) and particular cluster size. We train each subcategory with a discriminative clustering algorithm that iterates between selecting examples for *sharing* and *learning* detectors given those examples. Finally, we *select* a subset of candidate subcategory detectors for each object class as to maximize recognition accuracy. We show that this selection naturally produces subcategories with long-tail distribution of sizes.

3.3.1 Initialization

We begin by training a large “overcomplete” set of thousands to tens of thousands of candidate subcategory models in parallel. This large set of models will later be pruned. We initialize our subcategory models by learning a discriminative template for each positive example using exemplar SVMs [56]. We visualize exemplar root templates for cars in Fig. 3.5. In terms of category detection accuracy, they perform reasonably well (25% AP). But because it is easy to overfit to a single example, many templates include noisy features from the background.

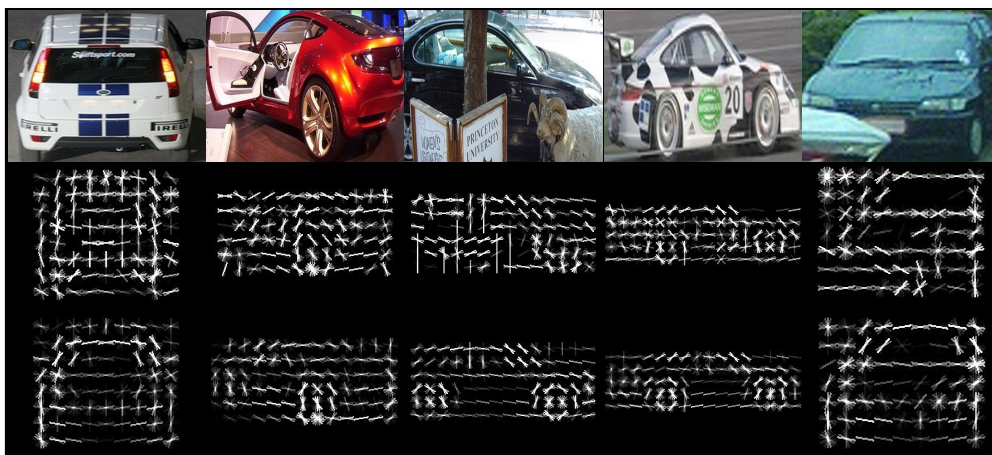


Figure 3.5: We visualize examples training images on the **top**. We show initial exemplar models trained with them in the **middle**. These templates perform well (25% AP on VOC2007), but sometimes emphasize incorrect gradients, such as the foreground tree in the center image. Retraining with the $n_m = 50$ highest-scoring examples (**bottom**) smooths out the template, de-emphasizing such noisy gradients (since they tend not be found in the n_m neighbors). This significantly improves performance to 42%. This suggests that optimal subcategory clusters may be overlapping, and maybe computed independently for each subcategory.

Sharing as regularization: To help learning more reliable templates for the rare examples, we retrain subcategory model m with the n_m highest-scoring positive examples under the exemplar model. We consider the sharing as a form of “regularization” that prevents overfitting to noisy gradients. To demonstrate the effect of sharing, we visualize the exemplar templates and the retrained templates for $n_m = 50$ in Fig. 3.5. The templates “regularized” by shared examples have less noisy gradients and almost double performance, producing an AP of 42%. Indeed, “averaging” across n_m similar training examples maybe more natural than penalizing the squared norm of

a template, as is typically done to prevent overfitting. This motivating example suggests that subcategory clusters need not be mutually exclusive and may overlap. In fact, we find that some positive examples are shared by many subcategories, a phenomenon that we will investigate later in Fig. 3.7.

Iteration: We make two further observations. First, one can iterate the procedure and find the n_m highest scoring examples with the retrained subcategory model and repeat. The optimal choice of neighbors for one cluster is independent of the choice of another cluster, suggesting these iterations can be performed independently and in parallel. We show in Sec. 3.3.2 that such a distributed, iterative algorithm is guaranteed to converge since it can be formalized as joint optimization of a well-defined (discriminative) objective function.

Cluster-size: Selecting the optimal cluster size n_m is tricky. We want large n_m for common cases. Rare clusters are particularly hard to model; from one perspective, they should use a small n_m so that learned detectors aren't polluted by visually dissimilar examples. On the other hand, models learned from very small clusters may tend to overfit because they are trained with less data. As argued above, we treat n_m as a subcategory-specific regularization parameter that is tuned on validation data. Specifically, we learn models for a log-linear range of $n_m \in N = \{50, 100, 200, 400, 800, 1600\}$ values. Given a dataset of positives P , we learn a large set of candidate subcategories mixtures M ($|M| = |N||P|$) *in parallel*, spanning both examples and cluster sizes. After training this large redundant set, we select a subset on validation data.

3.3.2 Discriminative clustering with sharing

We formalize the iterative algorithm introduced in the previous subsection. We do so by writing a objective function for jointly training *all* $|M|$ subcategory models, and describe a coordinate descent optimization that naturally decouples across subcategories.

Let us write a mixture of templates (can be part models or simply rigid templates) as

$$f(x) = \max_m w_m \cdot x \quad \text{where } m \in M \quad (3.1)$$

where m indicates a subcategory mixture component. M is the set of all mixture components. w_m is the template for m . x is an example. Given a training dataset (x_i, y_i) where $y_i \in \{-1, 1\}$ (for the detection problem), we explicitly write the non-convex learning objective as a function of both mixture models $\{w_m\}$ and binary latent variables $z_{im} \in \{0, 1\}$ that take on the value 1 if the i^{th} positive belongs to mixture component m . [67] refers to the $|P| \times |M|$ binary matrix of $Z = [z_{im}]$ as the “latent matrix”, where $|P|$ is the number of positive examples:

$$\begin{aligned} L(w, Z) = & \sum_{m \in M} \left[\frac{1}{2} \|w_m\|^2 \right. \\ & + C \sum_{i \in \text{pos}} z_{im} \max(0, 1 - w_m \cdot x_i) \\ & \left. + C \sum_{i \in \text{neg}} \max(0, 1 + w_m \cdot x_i) \right]. \end{aligned} \quad (3.2)$$

A standard latent SVM problem without sharing in [35] can be written as a joint minimization over (w, Z) subject to the constraints the rows of Z sum to 1 ($\sum_m z_{im} = 1$): each positive example i is assigned to exactly one mixture component.

We now replace the hard-assignment constraint $\sum_m z_{im} = 1$ with $\sum_i z_{im} = n_m$. Now we sum over i instead of m . This means that rather than forcing each positive to be assigned to exactly one mixture component, we force the mixture component m to consist of n_m examples. This constraint allows a single positive example i to be used to learn multiple mixtures, which provides a natural form of sharing between mixtures.

We describe a coordinate descent optimization algorithm for optimizing (3.2) subject to our new linear constraints:

1. **(Sharing)** $\min_Z L(w, Z)$: Compute $w_m \cdot x_i$ for $i \in \text{pos}$. Sort scores and set z_{im} for the n_m highest values to 1.
2. **(Learning)** $\min_w L(w, Z)$: Learn w_m with a convex program (SVM) using n_m positives and all negatives.

Step 1 optimizes latent assignment Z while keeping template weights w fixed. In detail, this step assigns n_m positive examples to each template w_m so as to minimize the hinge loss. The loss is exactly minimized by assigning the n_m highest scoring positives to m . Step 2 optimizes w while fixing Z . This step is a standard SVM problem. As both steps decrease the loss in (3.2), it is guaranteed to converge to a local optima.

Unlike other clustering methods such as k-means, both of the above steps can be performed independently and *in parallel* for all $|M|$ clusters.

3.3.3 Greedy model selection

For each object class, we generate a pool of $|M|$ candidate subcategory models. Typically, $|M|$ is in the thousands to tens of thousands. Many of these subcategory models will be redundant due to sharing. We want to compress the models by selecting a subset $S \subseteq M$.

We cast this as a combinatorial optimization problem: for a possible subset S , compute its average precision performance $AP(S)$ on a validation set, and select the subset that maximizes AP. To evaluate $AP(S)$, we run each subcategory model $m \in S$ on the validation set, eliminate overlapping detections from the subset S with non-maximum suppression (NMS), and compute a precision-recall curve.

The search over the powerset of M is clearly intractable, but we find a greedy strategy to work quite

well. Initialize $S = \{\}$ and repeatedly find the next-best mixture m to turn “on”:

1. $m^* := \underset{m}{\operatorname{argmax}} AP(S \cup m)$
2. $S := S \cup m^*$

A natural stopping condition is an insufficient increase in AP. In practice, we stop after instantiating a fixed number ($|S| = 50$) of subcategories. The first such instantiated subcategory tends to model a dominant cluster trained with a large n_m , while latter instantiations tend to consist of rare cases with small n_m .

To ensure that subcategory scores are comparable during NMS, we first calibrate the scores across models by mapping them to object class probabilities [66]. We use the same set of validation images for calibration and model selection. Calibration and selection is fast because the computationally-demanding portion (training a large pool of detectors and running them on validation images) is parallelized across all $|M|$ subcategory models. We visualize instantiated mixtures in Fig. 3.6 and Fig. 3.7.

3.4 Experimental results

Map-reduce: Our approach requires training and evaluating tens of thousands of DPMs across large numbers of object categories. In order to manage learning at this scale, we have implemented an in-house map-reduce version of the DPM codebase [1]. Map-reduce [18, 23] is an architecture for parallel computing. In our system, we use mappers to collect positive examples and negative images, and use mixture-specific reducers to learn the mixture models. This distributed architecture allows us to learn mixtures *in parallel* according to the formulation of (3.2).

Computation: Because our distributed training algorithm can be parallelized across $|M|$ cores,

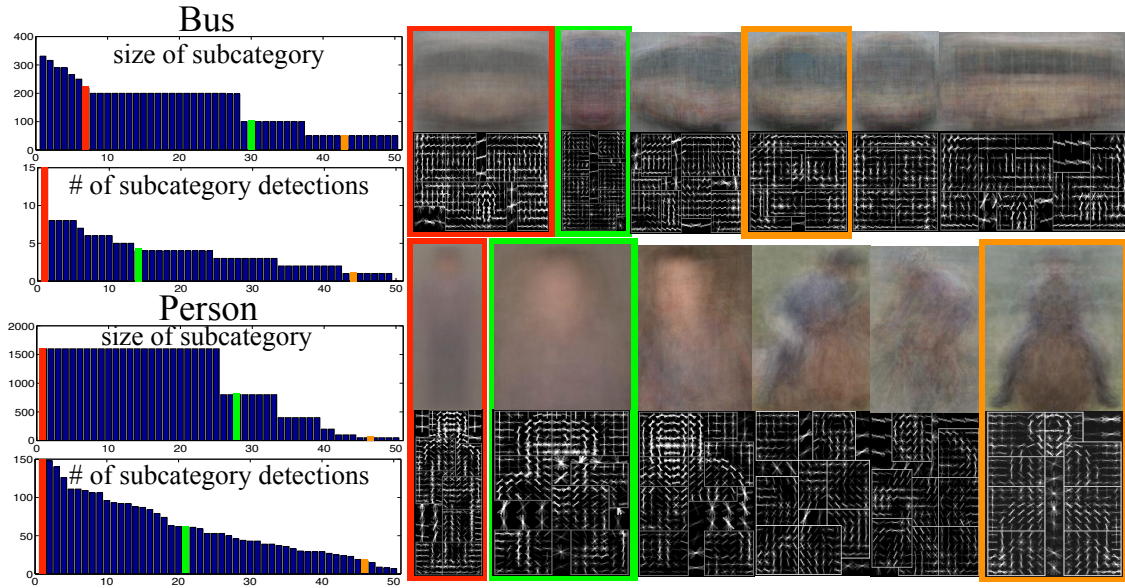


Figure 3.6: For two categories (bus and person), we plot the size of each subcategory cluster, as well as the number of true positive detections of that subcategory on test images (**left**). Both tend to follow a long-tail distribution; some subcategories are dominant, while many are rare. We visualize average images and templates associated with selected subcategories on the **right**. We find that bus subcategories correspond to bus viewpoint, while person subcategories correspond to different truncations, poses, and interactions with objects such as horses and bicycles.

each iteration of our learning algorithm is no slower than training a single-mixture DPM. We perform a fixed number (6) of discriminative clustering iterations, but find that cluster labels tend to stabilize after 3 iterations. At test-time, our long-tail subcategory models are equivalent to running $|S| = 50$ single-mixture DPMs in parallel, which takes about 1 second per image in our in-house implementation.

Benchmarks: Following much past work, we evaluate our detection system on PASCAL VOC 2007. Additionally, in order to test our implementation of DPMs (DistDPMs), we evaluate our DistDPMs on the Columbia Dog Breed Dataset [55] and the CUHK Cat Head Dataset [90]. The Columbia Dog dataset consist of 8000 dog images obtained from various repositories, while the CUHK Cat Dataset consists of 10000 Flickr images of cats. We used their standard training/test split: roughly 50%/50% for dogs and 70%/30% for cats.

Distributed DPMs: Before evaluating our proposed long-tail subcategory (LTS) models, we first

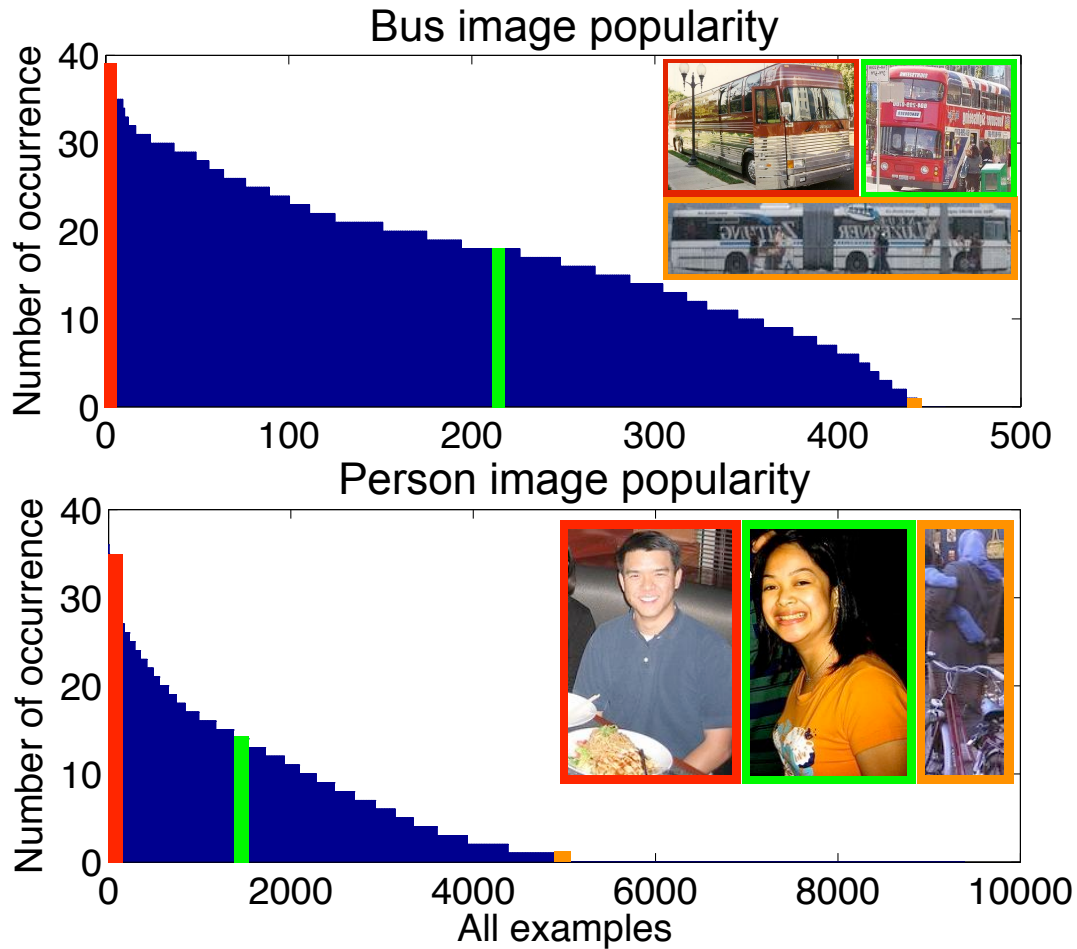


Figure 3.7: This plot reveals “popular” positive training images that are selected by many sub-categories. Popular training images (in red) are prototypes that are representative of the category. Unpopular or rare images (such as multi-body articulated bus, in orange) are used by less subcategories.

verify that our map-reduce DPM implementation can reproduce the state-of-the-art voc-release4 models of [1, 35], when we use the same number of mixtures and the same latent hard assignment strategy (no sharing) as in [1]. We compare to the raw detectors without any post-processing (bounding-box prediction or contextual rescoring) in Table 3.1. Our implementation produces an average AP of 28.1%, compared to 31.8% from [1]. The 3% drop is mainly due to the fact that [1] uses max-regularization over all mixtures and learns 3 pairs of mirrored mixtures rather than the 6 separate mixtures in our implementation. Both of these are hard to decouple and parallelize, but known to help the empirical performance.

Even given this shortcoming, we will show our codebase can be used to construct state-of-the-art sub-category models. We also verify that our re-implementation produces state-of-the-art performance on other benchmark datasets in Fig. 3.8.

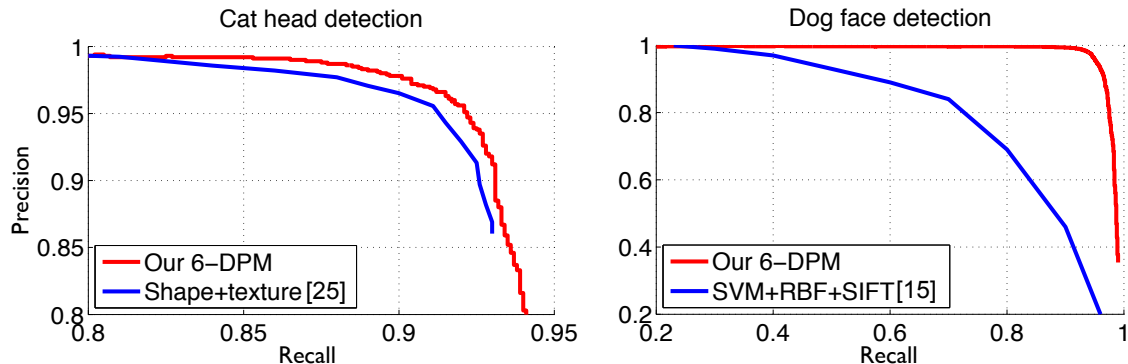


Figure 3.8: We run our implementation of 6-DistDPM on two public datasets of dog[55] and cat[90] face detection. Our 6-DistDPM outperforms the previously reported best methods on both datasets. Our improvement is particular large on the dog dataset, which arguably is more challenging due to the larger appearance variations of a dog face.

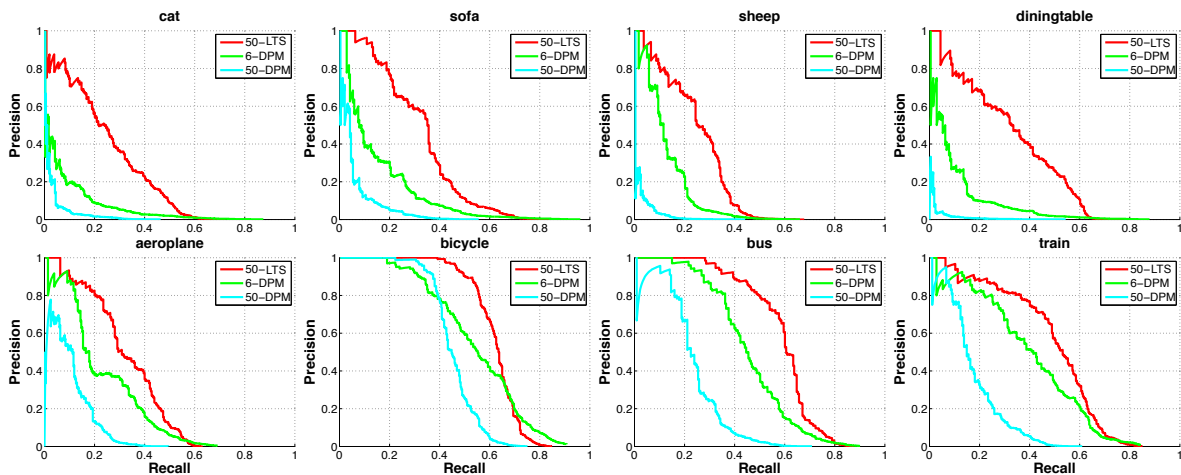


Figure 3.9: We plot precision-recall curves for PASCAL VOC 2007 categories. We show “hard” classes on the top row, where the baseline 6-mixture DPM (in green) performs poorly. These classes tend to have rich appearance variation (due to viewpoint, subclass structure, and occlusion from other objects). Scaling the baseline to 50-mixture DPM (in cyan) yields worse performance due to lack of training examples for each mixture component. Our 50-mixture long-tail subcategory model (LTS, in red) provides a significant improvement in such cases. The “easy” classes in the bottom row show similar trend. We further examine this performance increase in Fig. 3.11.

Long-tail subcategories (LTS). We use a subset of the VOC2011 training set (that does not overlap with the 2007 dataset) as validation data to greedily select our long-tail subcategory models

Category	plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	
voc-rel4[1]	29.6	57.3	10.1	17.1	25.2	47.8	55.0	18.4	21.6	24.7	
voc-rel5[2]	32.4	57.7	10.7	15.7	25.3	51.3	54.2	17.9	21.0	24.0	
LEO[92]	29.4	55.8	9.4	14.3	28.6	44.0	51.3	21.3	20.0	19.3	
DPM-WTA[24]	19	48	03	10	16	41	44	9	15	19	
ESVM[56]	20.8	48.0	7.7	14.3	13.1	39.7	41.1	5.2	11.6	18.6	
MCI+MCL[7]	33.3	53.6	9.6	15.6	22.9	48.8	51.5	16.3	16.3	20.0	
Our 6-DistDPM	26.6	54.3	9.4	14.4	24.8	46.5	50.5	12.8	14.1	26.3	
Our 50-LTS	34.1	61.2	10.1	18.0	28.9	58.3	58.9	27.4	21.0	32.3	
Category	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	total
voc-rel4[1]	23.3	11.2	57.6	46.5	42.1	12.2	18.6	31.9	44.5	40.9	31.8
voc-rel5[2]	25.7	11.6	55.6	47.5	43.5	14.5	22.6	34.2	44.2	41.3	32.5
LEO[92]	25.2	12.5	50.4	38.4	36.6	15.1	19.7	25.1	36.8	39.3	29.6
DPM-WTA[24]	23	10	52	34	20	10	16	28	34	34	24
ESVM[56]	11.1	3.1	44.7	39.4	16.9	11.2	22.6	17.0	36.9	30.0	22.7
MCI+MCL[7]	23.8	11.0	55.3	43.8	36.9	10.7	22.7	23.5	38.6	41.0	29.8
Our 6-DistDPM	14.0	3.2	57.7	43.2	35.2	10.1	16.9	18.2	41.6	41.2	28.1
Our 50-LTS	34.6	15.7	54.1	47.2	41.2	18.1	27.2	34.6	49.3	42.2	35.7

Table 3.1: Average precision for different object categories in PASCAL 2007 dataset. We compare our approach to the existing methods on the same benchmark. The first two rows contain the results reported in the released code of [1, 2] without any post-processing. Our parallel implementation with the same number of mixtures (6-DistDPM) is shown in the second last row. The last row is our long-tail subcategory model (LTS), which significantly improves performance, particularly on difficult categories with large appearance variation (Fig. 3.9).

(Sec. 3.3.3). Interestingly, we find that greedy selection on the original training data works well, producing only a 2% drop from the 50-LTS numbers in Table 3.1. We compare to previously published results that use the same feature set (HOG) without contextual post-processing, as the choice of features/post-processing is orthogonal to our work. We obtain the highest performance in 15/20 categories and the best overall performance. When compared to our in-house DPM implementation, our long tail models increase performance from 28.1% to 35.7%. We diagnose this performance increase (relative to our in-house implementation) in the following.

Hard vs easy classes: We split up the set of classes into hard vs easy, depending on baseline performance. We hypothesize that low baseline performance is indicative of a long tail (more

varied appearances) that are not well modeled by the 6 mixtures of a standard DPM. We more than **double** the average precision of the baseline for such categories (Fig. 3.11). On “easier” classes, 50-LTS still perform best, but 6-DPM is a close second. We posit that a smaller number of mixtures sufficiently captures the limited appearance variation of such classes. We plot performance versus the number of mixtures on the validation set in Fig. 3.10. For “easy” classes such as car, we find that a few mixtures do well. For “hard” classes with more appearance variation (such as cats), we find that adding additional mixtures, even beyond 50, will likely improve performance.

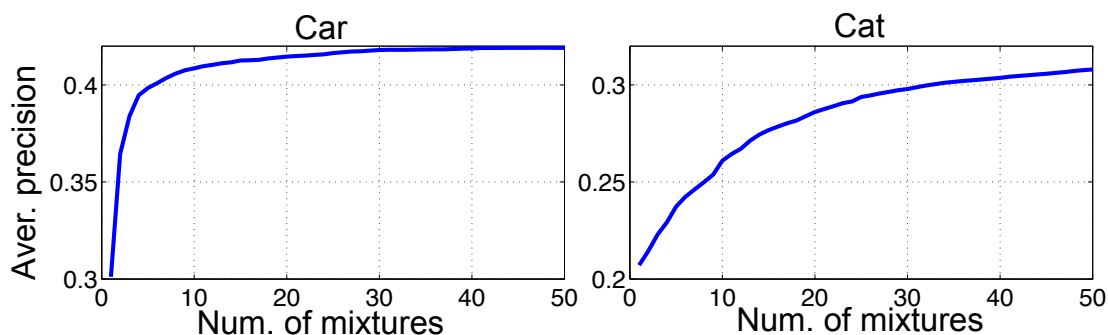


Figure 3.10: The performance as a function of the number of mixtures. For “easy” classes such as car, we find that a few mixtures do well. For “hard” classes with more appearance variation (such as cats), we find that adding additional mixtures, even beyond 50, improves performance.

50-LTS vs exemplars: We first compare to existing non-parametric models based on exemplar SVMs [56, 28]. One may hypothesize that rare subcategories (in the tail) are well modeled with a single exemplar. Our long-tail models with sharing considerably outperform such methods in Table 3.1, suggesting that sharing is still crucial for the rare cases.

50-LTS vs 50-DPM: We also compare to the widely-used latent mixture-assignment algorithm of [35] for large number of mixtures. At $|S| = 50$, although it seems that 50-LTS and 50-DPM have the same capacity, our approach consistently performs significantly better. The improvement may arise from two factors: (1) Our mixtures are trained with a brute-force search that avoids the local minima of latent reassignment. (2) Our mixtures share training examples across clusters. We now analyze the improvement due to (1) vs (2). We focus on examining the improvement due to “hard” classes. The similar observations apply to the “easier” classes too.

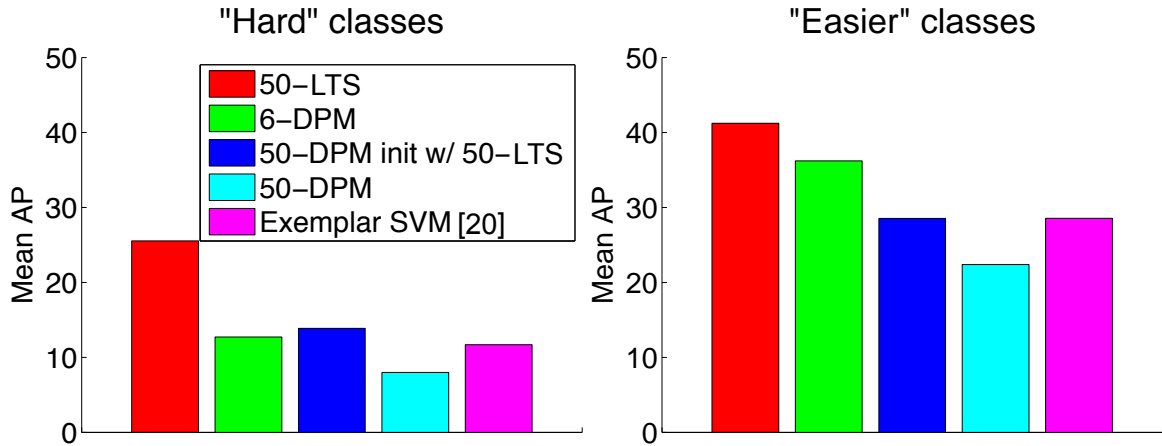


Figure 3.11: We separately analyze “hard” PASCAL classes with large appearance and viewpoint variations $\{bird, cat, dog, sheep, plant, table, sofa \text{ and } chair\}$; these typically have lower average precision (AP) than other classes. On hard classes, our LTS **doubles** the AP of DPMs, increasing accuracy from 12% to 25%. On easier categories, our LTS model still outperforms 6-mixture DPMs by 5%. To analyze these improvements, we first scale DPMs to large (50) mixtures (50-DPM), which consistently underperform the DPMs with 6 mixtures (6-DPM). When 50-DPMs are initialized with our 50-LTS models, performance increases (suggesting that poor latent variable initialization is one reason to blame). When these models are allowed to share training examples, we see the largest performance increase, suggesting our **overlapping subcategory models are crucial for large-mixture representations**.

Effect of brute-force search: One might argue that discriminative latent-reassignment may also benefit from a brute-force search over initializations of the latent variables. To help answer this, we initialize latent reassignment of 50-DPM using our 50-LTS detectors. This ensures that latent reassignment is initialized with a set of 50 good models that produce an AP of 25% (on the “hard” classes). This better initialization increases performance of 50-DPM from 7% to 13% AP, suggesting that latent-reassignment in [35] does suffer from local-minima.

Effect of overlapping subcategories: The above experiment also points out a conspicuous *drop* in performance; when initializing with 50 good detectors (25%AP), hard latent reassignment (that eliminates sharing by forcing each positive example to belong to a single cluster) dramatically drops performance to 13%AP. This suggests that the gains due to *sharing* are even more important than the gains due to better initialization. In our algorithm, initialization and sharing are intimately intertwined.

Conclusion: Many current approaches to object recognition successfully model the dominant iconic

appearances of objects, but struggle to model the long tail of rare appearances. We show that distributed optimization and example sharing partially address this issue. We introduce a discriminative clustering algorithm that naturally allows for example sharing and distributed learning. We use this algorithm to perform a “brute-force” search over subcategory initialization and subcategory size, and demonstrate that the resulting models significantly outperform past work on difficult objects with varied appearance. We posit that our performance is now limited by the lack of training data for the rare subcategories in the tail. We may need large training datasets to fully encounter the set of rare cases. Our analysis suggests that “big-rare-data”, which focuses on rare examples not already seen, may be more beneficial than traditional “big-data”.

Chapter 4

Sharing local appearance with parts

In the last chapter, in order to address the lack of training data for the rare cases, we propose a notion of overlapping subcategories: examples are shared across subcategories. We call this type of sharing “global appearance sharing”, as illustrated in Fig.3.3. However, such global sharing may still be limited as a long tail distribution suggests that there are many subcategories that are never observed in the training set. How do we learn model for subcategories that have never been seen? Global sharing does not appear to be quite the answer, since we do not even observe any data that can be shared.

In this chapter, we introduce the notion of “local appearance sharing”. Local correspondences among certain spatial regions are established via “parts” across training examples. For example, a rarely seen lamborghini may still have headlights or wheels that look similar to other common cars. Similarly, parts of the lip or eyes still look similar across different viewpoints and expressions (Fig. 4.1). We argue that by rearranging parts into new arrangements and combinations (we call this “synthesizing”) one can model an exponential number of subcategories, even those not observed in the training data. Our analysis suggests that the “sharing and synthesizing” approach is crucial and effective for recognizing those unseen configurations.

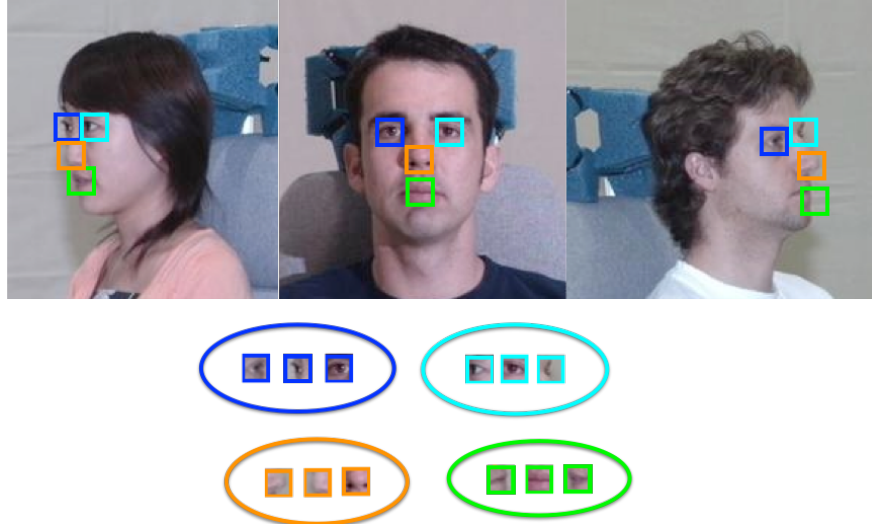


Figure 4.1: Local regions can be associated together to learn local “parts”.

In this chapter, we first introduce a deformable part model in Section 4.1. Section 4.2 shows that it can be viewed as an compositional model to efficiently represent and score an exponential number of rigid templates. We later describe other architectures of compositional models with local appearance sharing and derive a unified formulation of all of them as mixture models. We show and discuss our experimental results in Section 4.3.

4.1 Deformable part model

4.1.1 Tree structured part model

Our model is based on mixture of trees with a shared pool of parts V . We write each tree $T_m = (V_m, E_m)$ as a linearly-parameterized, tree-structured pictorial structure [89], where m indicates a mixture and $V_m \subseteq V$. Let us write I for an image, and $l_i = (x_i, y_i)$ for the pixel location of part i . We score a configuration of parts $L = \{l_i : i \in V\}$ as:

$$S(I, L, m) = \text{App}_m(I, L) + \text{Shape}_m(L) + \alpha^m \quad (4.1)$$

$$\text{App}_m(I, L) = \sum_{i \in V_m} w_i^m \cdot \phi(I, l_i) \quad (4.2)$$

$$\text{Shape}_m(L) = \sum_{ij \in E_m} a_{ij}^m dx^2 + b_{ij}^m dx + c_{ij}^m dy^2 + d_{ij}^m dy \quad (4.3)$$

Eqn.4.2 sums the appearance evidence for placing a template w_i^m for part i , tuned for mixture m , at location l_i . We write $\phi(I, l_i)$ for the feature vector (e.g., HoG descriptor) extracted from pixel location l_i in image I . Eqn.4.3 scores the mixture-specific spatial arrangement of parts L , where $dx = x_i - x_j$ and $dy = y_i - y_j$ are the displacement of the i th part relative to the j th part. Each term in the sum can be interpreted as a spring that introduces spatial constraints between a pair of parts, where the parameters (a, b, c, d) specify the rest location and rigidity of each spring. Finally, the last term α^m is a scalar bias or “prior” associated with viewpoint mixture m .

4.1.2 Shape model

Because the location variables l_i in Eqn.4.3 only appear in linear and quadratic terms, the shape model can be rewritten as:

$$\text{Shape}_m(L) = -(L - \mu_m)^T \Lambda_m (L - \mu_m) + \text{constant} \quad (4.4)$$

where (μ, Λ) are re-parameterizations of the shape model (a, b, c, d) ; this is akin to a canonical versus natural parameterization of a Gaussian. In our case, Λ_m is a block sparse precision matrix, with non-zero entries corresponding to pairs of parts i, j connected in E_m . One can show Λ_m is positive semidefinite if and only if the quadratic spring terms a and c are negative. We prove this in Appendix. 4.4. This corresponds to a shape score that penalizes configurations of L that deform from the ideal shape μ_m . The eigenvectors of Λ_m associated with the smallest eigenvalues represent

modes of deformation associated with small penalties. Notably, we discriminatively train (a, b, c, d) (and hence μ and Λ) in a max-margin framework.

4.1.3 Inference

Inference corresponds to maximizing $S(I, L, m)$ in Eqn.4.1 over L and m :

$$S^*(I) = \max_m [\max_L S(I, L, m)] \quad (4.5)$$

Simply enumerate all mixtures, and for each mixture, find the best configuration of parts. Since each mixture $T_m = (V_m, E_m)$ is a tree, the inner maximization can be done efficiently with dynamic programming(DP) [37].

We let $\text{child}(j)$ be the set of children of j in V_m . We compute the message that part j passes to its parent part i by the following:

$$\text{score}_j(l_j) = w_j \cdot \phi(I, l_j) + \sum_{k \in \text{child}(j)} m_{k \rightarrow j}(l_j) \quad (4.6)$$

$$m_{j \rightarrow i}(l_i) = \max_{l_j} \text{score}_j(l_j) + w_{i,j} \cdot \phi(l_i - l_j) \quad (4.7)$$

The DP process is the same for all the mixtures. We ignore the superscript m that indicates mixture in Eqn.4.6, 4.7 for the simplicity of notation.

Eqn.4.6 computes the local score of part j at all pixel locations l_j , by collecting messages from the children of j . Eqn.4.7 computes for every locations of part i , the best scoring location of its child part j . Once messages are passed to the root part ($i = 1$), $\text{score}_1(l_1)$ represents the best scoring configuration for each root position. One can use the root scores to generate multiple detections in images I by thresholding them and applying non-maximum suppression (NMS). By keeping

track of the argmax indices, one can backtrack to find the locations of each part in each maximal configuration.

4.1.4 Learning

We first learn the tree-structure, then learn both shape and appearance parameters discriminatively using a structured prediction framework.

Learning tree structure: When local part correspondence is given (this can be obtained with human supervision), we use the Chow-Liu algorithm [17] to find the maximum likelihood tree structure T^* that best explains the part locations for a given mixture, assuming parts are Gaussian distributed.

$$T^* = \arg \max_E \prod_n \left[\prod_i p(l_{i,n}) \prod_{i,j \in E} \frac{p(l_{i,n}, l_{j,n})}{p(l_{i,n})p(l_{j,n})} \right] \quad (4.8)$$

When E is restricted to a tree, the above is equivalent to computing the minimum spanning tree (MST) of a undirected complete graph, where the weight of each edge is assigned to be the mutual information between the location of part i and j that are connected by this edge[17]. Under a joint Gaussian assumption of part locations, the mutual information (or edge weight) for a pair of parts is:

$$e_{i,j} = \frac{1}{2} (\log |\Sigma_{l_i}| + \log |\Sigma_{l_j}| - \log |\Sigma_{l_i, l_j}|) \quad (4.9)$$

where Σ_{l_i} is the covariance matrix of l_i , Σ_{l_i, l_j} is the covariance matrix of l_i and l_j . We use sample estimates of these parameters learned from labeled training data.

Learning model parameters: Given labeled positive examples $\{I_n, L_n, m_n\}$ and negative examples $\{I_n\}$, we will define a structured prediction objective function similar to one proposed in [89]. To do so, let's write $z_n = \{L_n, m_n\}$. Note that the scoring function Eqn.4.1 is linear in the part

templates w , spring parameters (a, b, c, d) , and mixture biases α . and mixture biases $\{m\}$. [35], Concatenating these parameters into a single vector β , we can write the score as:

$$S(I, z) = \beta \cdot \Phi(I, z) \quad (4.10)$$

The vector $\Phi(I, z)$ is sparse, with nonzero entries in a single interval corresponding to mixture m .

Now we can learn a model of the form:

$$\begin{aligned} \arg \min_{\beta, \xi_n \geq 0} \quad & \frac{1}{2} \beta \cdot \beta + C \sum_n \xi_n & (4.11) \\ \text{s.t.} \quad & \forall n \in \text{pos} \quad \beta \cdot \Phi(I_n, z_n) \geq 1 - \xi_n \\ & \forall n \in \text{neg}, \forall z \quad \beta \cdot \Phi(I_n, z) \leq -1 + \xi_n \\ & \forall k \in K, \quad \beta_k \leq 0 \end{aligned}$$

The above constraint states that positive examples should score better than 1 (the margin), while negative examples, for all configurations of part positions and mixtures, should score less than -1. The objective function penalizes violations of these constraints using slack variables ξ_n . We write K for the indices of the quadratic spring terms (a, c) in parameter vector β . The associated negative constraints ensure that the shape matrix Λ is positive semidefinite (Sec.4.1.2). We solve this problem using the dual coordinate-descent solver in [89], which accepts negativity constraints.

4.2 Revisit mixture models

The two models we proposed in previous chapters: partitioned subcategory models (Chapter 2) and overlapping subcategory models (Chapter 3) can both be considered as a mixture of linear classifiers, or templates. Formally speaking, we compute the detection score of an image window I

as:

$$S(I) = \max_m \left[w_m \cdot \phi(I) + b_m \right] \quad (4.12)$$

where m is a discrete mixture variable, $\Phi(I)$ is a HOG image descriptor [22], w_m is a linearly-scored template, and b_m is an (optional) bias parameter that acts as a prior that favors particular templates over others.

In this section, we describe various architectures for compositional models that share local spatial regions of templates, or parts. We formulate them in a unified view, and show how they can be interpreted and extended as high-capacity mixture models.

4.2.1 Deformable Part Models (DPMs)

We begin with an analysis that shows that DPMs are equivalent to an exponentially-large mixture of rigid templates Eqn. (4.12). This allows us to analyze (both theoretically and empirically) under what conditions a classic mixture model will approach the behavior of a DPM. Let the location of part i be (x_i, y_i) . Given an image I , a DPM scores a configuration of P parts $(x, y) = \{(x_i, y_i) : i = 1..P\}$ as:

$$\begin{aligned} S_{DPM}(I) &= \max_{x,y} S(I, x, y) \quad \text{where} \\ S(I, x, y) &= \sum_{i=1}^P \sum_{(u,v) \in W_i} \alpha_i[u, v] \cdot \phi(I, x_i + u, y_i + v) \\ &\quad + \sum_{ij \in E} \beta_{ij} \cdot \psi(x_i - x_j - a_{ij}^{(x)}, y_i - y_j - a_{ij}^{(y)}) \end{aligned} \quad (4.13)$$

where W_i defines the spatial extent (length and width) of part i . The first term defines a local appearance score, where α_i is the appearance template for part i and $\phi(I, x_i, y_i)$ is the appearance feature vector extracted from location (x_i, y_i) . The second term defines a pairwise deformation

model that scores the relative placement of a pair of parts with respect to an anchor position $(a_{ij}^{(x)}, a_{ij}^{(y)})$. For simplicity, we have assumed all filters are defined at the same scale, though the above can be extended to the multi-scale case. When the associated relational graph $G = (V, E)$ is tree-structured, one can compute the best-scoring part configuration $\max_{(x,y) \in \Omega} S(I, x, y)$ with dynamic programming, where Ω is the space of possible part placements. Given that each of P parts can be placed at one of L locations, $|\Omega| = L^P \approx 10^{20}$ for our models.

By defining index variables in image coordinates $u' = x_i + u$ and $v' = y_i + v$, we can rewrite Eqn. (4.13) as:

$$\begin{aligned}
S(I, x, y) &= \sum_{u', v'} \sum_{i=1}^P \alpha_i [u' - x_i, v' - y_i] \cdot \phi(I, u', v') \\
&\quad + \sum_{ij \in E} \beta_{ij} \cdot \psi_{ij}(x_i - x_j - a_{ij}^{(x)}, y_i - y_j - a_{ij}^{(y)}) \\
&= \left(\sum_{u', v'} w(x, y)[u', v'] \cdot \phi(I, u', v') \right) + b(x, y) \\
&= w(x, y) \cdot \phi(I) + b(x, y)
\end{aligned} \tag{4.14}$$

where $w(x, y)[u', v'] = \sum_{i=1}^P \alpha_i [u' - x_i, v' - y_i]$. For notational convenience, we assume parts templates are padded with zeros outside of their default spatial extent.

From the above expression it is easy to see that the DPM scoring function is formally equivalent to an exponentially-large mixture model where each mixture component m is indexed by a particular configuration of parts (x, y) . The template corresponding to each mixture component $w(x, y)$ is constructed by adding together parts at shifted locations. The bias corresponding to each mixture component $b(x, y)$ is equivalent to the spatial deformation score for that configuration of parts.

DPMs differ from classic mixture models previously defined in that they (1) share parameters across a large number of mixtures or rigid templates, (2) extrapolate by “synthesizing” new templates not encountered during training, and finally, (3) use dynamic programming to efficiently search over a

large number of templates.

4.2.2 Exemplar Part Models (EPMs)

To analyze the relative importance of part parameter sharing and extrapolation to new part placements, we define a part model that limits the possible configurations of parts to those seen in the N training images, written as

$$S_{EPM}(I) = \max_{(x,y) \in \Omega_N} S(I, x, y) \quad \text{where} \quad \Omega_N \subseteq \Omega. \quad (4.15)$$

We call such a model an Exemplar Part Model (EPM), since it can also be interpreted as set of N rigid exemplars with shared parameters. EPMs are not to be confused with exemplar DPMs (EDPMs), which we will shortly introduce as their deformable counterpart. EPMs can be optimized with a discrete enumeration over N rigid templates rather than dynamic programming. However, by caching scores of the local parts, this enumeration can be made quite efficient even for large N . EPMs have the benefit of sharing, but cannot synthesize new templates that were not present in the training data. We visualize example EPM templates in Fig. 4.2.

To take advantage of additional training data, we would like to explore non-parametric mixtures of DPMs. One practical issue is that of computation. We show that with a particular form of sharing, one can construct non-parametric DPMs that are no more computationally complex than standard DPMs or EPMs, but considerably more flexible in that they extrapolate multi-modal shape models to unseen configurations.

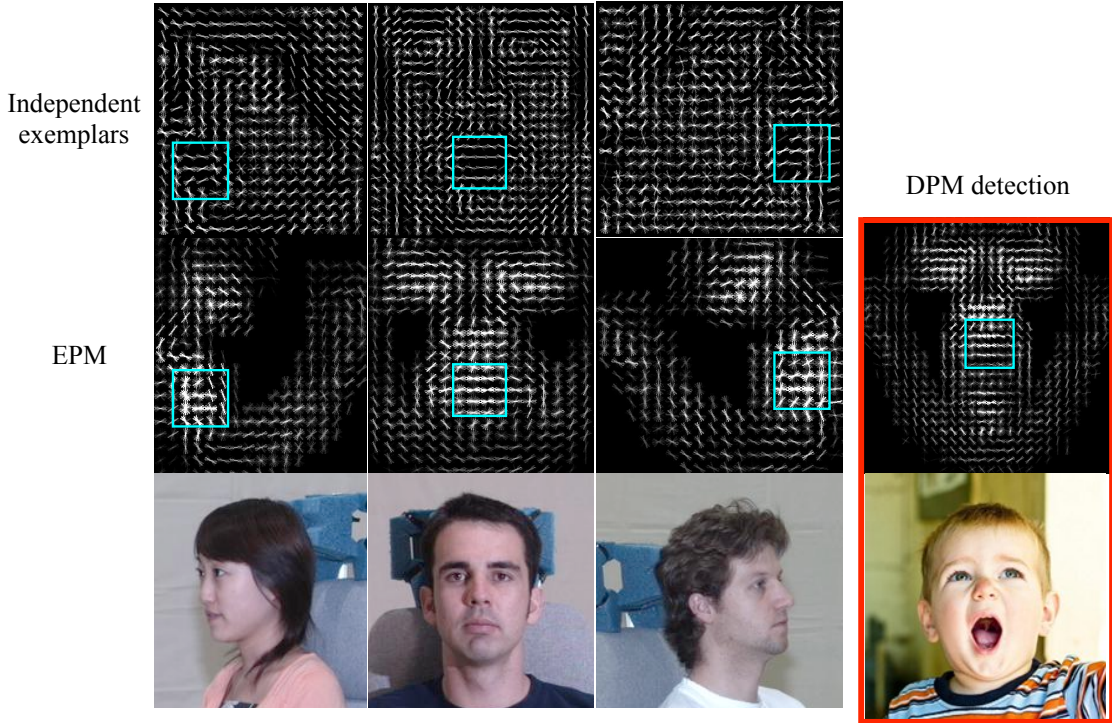


Figure 4.2: Classic exemplars vs EPMS. On the top row, we show three rigid templates trained as independent exemplar templates. Below them, we show their counterparts from an exemplar part model (EPM), along with their corresponding training images. EPMS share spatially-localized regions (or “parts”) between mixtures. Each rigid mixture is a superposition of overlapping parts. A single part is drawn in blue. We show parts on the top row to emphasize that these template regions are trained independently. On the [right], we show a template which is implicitly synthesized by a DPM for a novel test image on-the-fly. In Fig. 4.4, we show that both sharing of parameters between mixture components and implicit generation of mixture components corresponding to unseen part configurations contribute to the strong performance of a DPM.

4.2.3 Exemplar DPMs (EDPMs)

To describe our model, we first define a mixture of DPMs with a shared appearance model, but mixture-specific shape models. In the extreme case, each mixture will consist of a single training exemplar. We describe an approach that shares both the part filter computations *and* dynamic programming messages across all mixtures, allowing us to eliminate almost all of the mixture-dependant computation. Specifically, we consider mixture of DPMs of the form:

$$S(I) = \max_{m \in \{1 \dots M\}} \max_{z \in \Omega} \left[w(z) \cdot \phi(I) + b_m(z) \right] \quad (4.16)$$

where $z = (x, y)$ and we write a DPM as an inner maximization over an exponentially-large set of templates indexed by $z \in \Omega$, as in Eqn. (4.14). Because the appearance model does not depend on m , we can write:

$$S(I) = \max_{z \in \Omega} \left[w(z) \cdot \phi(I) + b(z) \right] \quad (4.17)$$

where $b(z) = \max_m b_m(z)$. Interestingly, we can write the DPM, EPM, and EDPM in the form of Eqn. (4.17) by simply changing the shape model $b(z)$:

$$b_{DPM}(z) = \sum_{ij \in E} \beta_{ij} \cdot \psi(z_i - z_j - a_{ij}) \quad (4.18)$$

$$b_{EDPM}(z) = \max_{m \in \{1 \dots M\}} \sum_{ij \in E} \beta_{ij} \cdot \psi(z_i - z_j - a_{ij}^m) \quad (4.19)$$

$$b_{EPM}(z) = b_{DPM}(z) + b_{EDPM}^*(z) \quad (4.20)$$

where a_{ij}^m is the anchor position for part i and j in mixture m . We write $b_{EDPM}^*(z)$ to denote a limiting case of $b_{EDPM}(z)$ with $\beta_{ij} = -\infty$ and thus takes on a value of 0 when z has the same relative part locations as some exemplar m and $-\infty$ otherwise.

While the EPM only considers M different part configurations to occur at test time, the EDPM extrapolates away from these shape exemplars. The spring parameters β in the EDPM thus play a role similar to the kernel width in kernel density estimation. We show a visualization of these shape models as probabilistic priors in Fig. 4.3.

4.2.4 Inference

We now show that inference on EDPMs (Eqn. 4.19) can be quite efficient. Specifically, inference on a star-structured EDPM is no more expensive than a EPM built from the same training examples. Recall that EPMs can be efficiently optimized with a discrete enumeration of N rigid templates

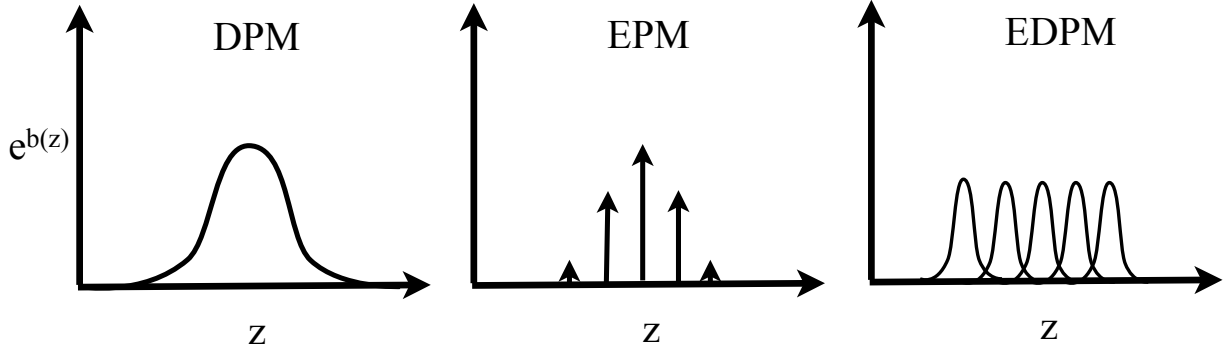


Figure 4.3: We visualize exponentiated shape models $e^{b(z)}$ corresponding to different part models. A DPM uses a unimodal Gaussian-like model (**left**), while a EPM allows for only a discrete set of shape configurations encountered at training (**middle**). An EDPM non-parametrically models an arbitrary shape function using a small set of basis functions. From this perspective, one can view EPMs as special cases of EDPMs using scaled delta functions as basis functions.

with “intelligent caching” of part scores.

Intuitively, one computes a response map for each part, and then scores a rigid template by looking up shifted locations in the response maps. EDPMs operate in a similar same manner, but one convolves a “min-filter” with each response map before looking up shifted locations.

To be precise, we explicitly write out the message-passing equations for a star-structured EDPM below, where we assume part $i = 1$ is the root without loss of generality:

$$S_{EDPM}(I) = \max_{z_1, m} \left[\alpha_1 \cdot \phi(I, z_1) + \sum_{j>1} m_j(z_1 + a_{1j}^m) \right] \quad (4.21)$$

$$m_j(z_1) = \max_{z_j} \left[\alpha_j \cdot \phi(I, z_j) + \beta_{1j} \cdot \psi(z_1 - z_j) \right] \quad (4.22)$$

The maximization in Eqn. (4.22) needs only be performed once across mixtures, and can be computed efficiently with a single min-convolution or distance transform [36]. The resulting message is then shifted by mixture-specific anchor positions a_{1j}^m in Eqn. (4.21). Such mixture-independent messages can be computed only for leaf parts, because internal parts in a tree will receive mixture-specific messages from downstream children. Hence star EDPMs are essentially no

more expensive than a EPM (because a single min-convolution per part adds a negligible amount of computation). In our experiments, running a 2000-mixture EDPM is almost as fast as a standard 6-mixture DPM. Other topologies beyond stars might provide greater flexibility. However, since EDPMs encode shape non-parametrically using many mixtures, each individual mixture may need not deform too much, making a star-structured deformation model a reasonable approximation (Fig. 4.3).

4.3 Experiments

We now perform a detailed analysis of compositional mixture models, including DPMs, EPMs, and EDPMs. We focus on face detection and Pascal buses. We consider the latent star-structured DPM of [35] as our primary baseline. For face detection, we also compare to the supervised tree-structured DPM of [94], which uses facial landmark annotations in training images as supervised part locations. Each of these DPMs makes use of different parts, and so can be used to define different EPMs and EDPMs. We plot performance of faces in Fig. 4.4 and buses in Fig.4.5.

Supervised DPMs: For face detection, we first note that a supervised DPM can perform quite well (91% AP) with less than 200 example faces. This represents a lower bound on the maximum achievable performance with a mixture of linear templates given a fixed training set. This performance is noticeably higher than that of our partitioned subcategory model, which maxes out at an AP of 76% with 900 training examples. By extrapolation, we predict that one would need $N = 10^{10}$ training examples to achieve the DPM performance. Both performs significant better than the exemplar models who do not share examples across templates. To analyze where this performance gap is coming from, we now evaluate the performance of various compositional mixtures models.

Latent parts: We begin by analyzing the performance of compositional mixtures defined by latent parts, as they can be constructed for both faces and Pascal buses. Recall that EPMs have the

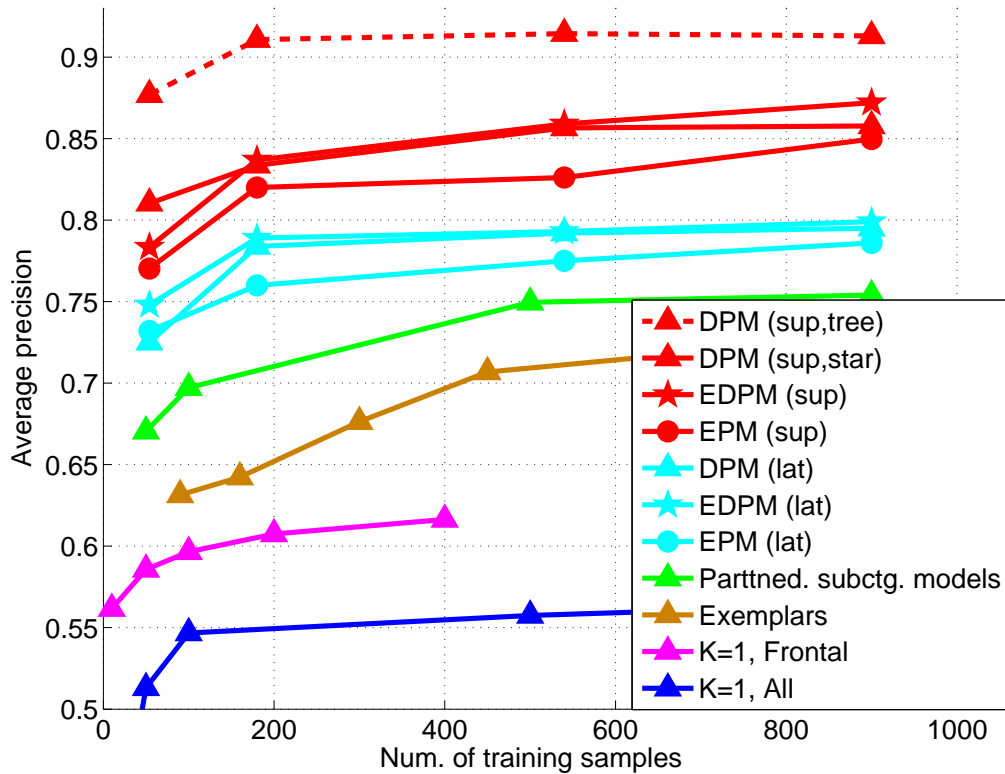


Figure 4.4: We compare the performance of mixtures models with EPMs and latent/supervised DPMs for the task of face detection. A single rigid template ($K = 1$) tuned for frontal faces outperforms the one tuned for all faces (as shown in Fig. 2.7). Mixture models boost performance to 76%, approaching the performance of a latent DPM (79%). The EPM shares supervised part parameters across rigid templates, boosting performance to 85%. The supervised DPM (91%) shares parameters but also implicitly scores additional templates not seen during training.

benefit of sharing parameters between rigid templates, but they cannot extrapolate to new shape configurations not seen among the N training examples. EPMs noticeably improve performance over partitioned subcategory models, improving AP from 76% to 78.5% for faces and improving AP from 56% to 64% for buses. In fact, for large N , they approach the performance of latent DPMs, which is 79% for faces and 63% for buses. For small N , EPMs somewhat underperform DPMs. This makes sense: with very few observed shape configurations, exemplar-based methods are limited.

Supervised parts: Here, supervised EPMs outperform partitioned subcategory mixtures 85% to 76%. Perhaps surprisingly, EPMs even outperform latent DPMs. However, supervised EPMs still

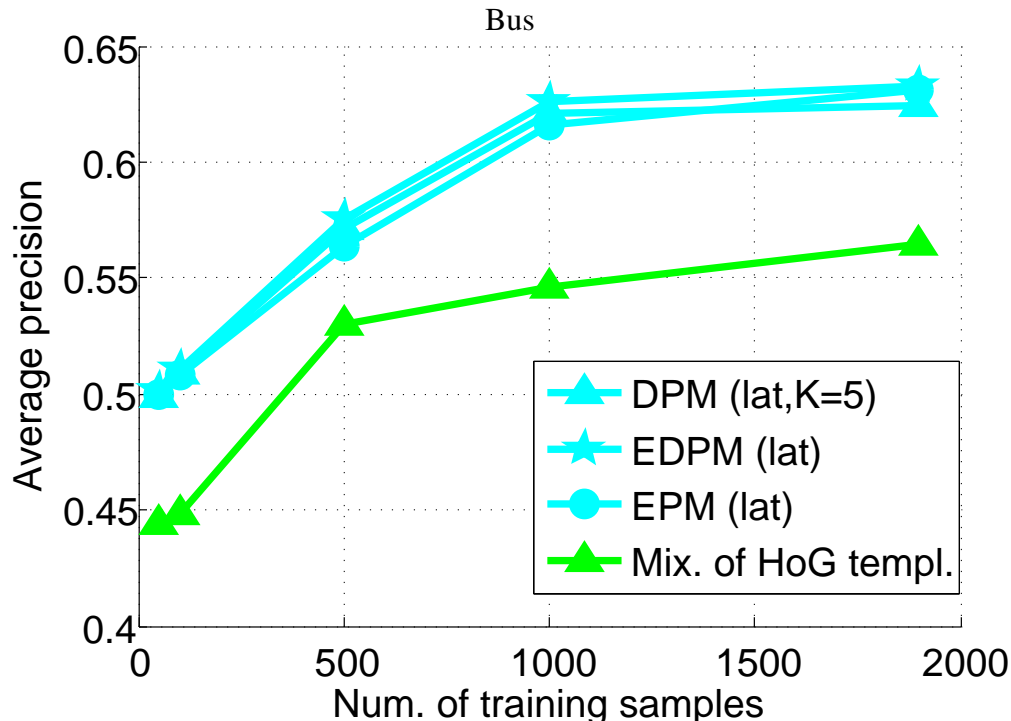


Figure 4.5: We compare the performance of mixture models with latent EPMs, EDPMs, and DPMs for bus detection. In the latent setting, EPMs significantly outperform the rigid mixtures of template and match the performance of the standard latent DPMs.

underperform a supervised DPM. This suggests that, in the supervised case, *the performance gap (85% vs 91%) stems from the ability of DPMs to synthesize configurations that are not seen during training*. Moreover, the reduction in relative error due to extrapolation is more significant than the reduction due to part sharing. [94] point out that a tree-structured DPM significantly outperforms a star-structured DPM, even when both are trained with the same supervised parts. One argument is that trees better capture nature spatial constraints of the model, such as the contour-like continuity of small parts. Indeed, we also find that a star-structured DPM does a “poorer” job of extrapolation. In fact, we show that an EDPM does as well a supervised star model, but not quite up to the performance of a tree DPM.

Analysis: Our results suggest that part models can be seen as a mechanism for performing intelligent parameter sharing across observed mixture components and extrapolation to implicit, unseen mixture components. Both these aspects contribute to the strong performance of DPMs. However, with the

“right” set of (supervised) parts and the “right” geometric (tree-structured) constraints, extrapolation to unseen templates has the potential to be much more significant. Once the representation for sharing and extrapolation is accurately specified, fairly little training data is needed. Indeed, our analysis shows that one can train a state-of-the-art face detector [94] with 50 face images.

Relation to Exemplar SVMs: In the setting of object detection, we were not able to see significant performance improvements of EDPMs over DPMs. However, EDPMs may be useful for other tasks. Specifically, they share an attractive property of exemplar SVMs (ESVMs) [56]: each detection can be affiliated with its closest matching training example (given by the mixture index), allowing us to transfer annotations from a training example to the test instance. [56] argue that non-parametric label transfer is an effective way of transferring associative knowledge, such as 3D pose, segmentation masks, attribute labels, etc. However, unlike ESVMs, EDPMs share computation among the exemplars (and so are faster), can generalize to unseen configurations (since they can extrapolate to new shapes), and also report a part deformation field associated with each detection (which maybe useful to warp training labels to better match the detected instance). We show example detections (and their matching exemplars) in Fig. 4.6.

Progression of performance: Fig. 4.4 shows an interesting progression of performance when models become more sophisticatedly designed and data are carefully shared. Note that all the methods in the figure are trained with the same set of 900 faces.

A single template model even with carefully data cleanup and model regularization described in Chapter 2, performs the worst with an average precision (AP) of 61.6%. Exemplars, although are prone to overfitting, can model appearance variations using multiple templates. It improves the performance to 72.9%. The partitioned subcategory models, with cross validated example grouping and parameter setting, achieve noticeable improvement over previous two, reaching 75.4%. This improvement is probably due to the sharing of examples which act as regularization for learning model templates. EPMs share local appearances across mixture templates. They are essentially exemplar templates with local part sharing. It achieves 85.0% ap, significantly higher

Best Matching Exemplar	Test Example
	
	
	
	

Figure 4.6: We visualize detections using our exemplar DPM (EDPM) model. As opposed to existing exemplar-based methods [56], our model shared parameters between exemplars (and so is faster to evaluate) and can generalize to unseen shape configurations. Moreover, EDPMs returns corresponding landmarks between an exemplar and a detected instance (and hence an associated set of landmark deformation vectors), visualized on the top row of faces.

than the previous three models. Finally, DPMs have the ability to synthesize unseen new shapes by deforming the parts. It achieves 91.3%. The 6% improvement over EPMs shows the power of synthesizing unseen cases.

Our results suggest that we could gain substantial performance improvement without any change in the features or the class of discriminant functions. Establishing and using accurate, clean correspondence among training examples (e.g., that specify that certain examples belong to the same sub-category, or that certain spatial regions correspond to the same part) and developing parameter sharing and compositional approaches appear to be a promising direction.

4.4 Conclusion

This chapter proposes various compositional models that share local appearance using parts. We show that all those models can be considered as mixture models of a set of rigid templates.

We show that the performance ceiling in Chapter 2 is not a consequence of HOG+linear classifiers. We provide an analysis of the popular deformable part model (DPM), showing that it can be viewed as an efficient way to implicitly encode and score an exponentially-large set of rigid mixture components with shared parameters. With the appropriate sharing, DPMs produce substantial performance gains over standard mixture models without local sharing. This suggests that larger gains were possible by enforcing richer constraints within the model, often through parameter sharing and compositional representations that that implicitly make use of augmented training sets appear the most promising directions.

Appendices

Proof Λ_m is positive semi-definite (PSD) if and only if the quadratic spring terms are negative

In our main submission, we state that “ Λ_m is positive semidefinite if and only if the quadratic spring terms a and c are negative”. The proof is as follows, ignoring the mixture index m for simplicity:

$$\text{Shape}(L) = \sum_{ij \in E} a_{ij} dx^2 + b_{ij} dx + c_{ij} dy^2 + d_{ij} dy \quad (4.23)$$

$$= - \sum_{ij \in E} \begin{pmatrix} l_i - \mu_i \\ l_j - \mu_j \end{pmatrix}^T \Lambda_{i,j} \begin{pmatrix} l_i - \mu_i \\ l_j - \mu_j \end{pmatrix} + \text{constant} \quad (4.24)$$

$$\text{where } \Lambda_{i,j} = - \begin{bmatrix} a_{ij} & 0 & -a_{ij} & 0 \\ 0 & c_{ij} & 0 & -c_{ij} \\ -a_{ij} & 0 & a_{ij} & 0 \\ 0 & -c_{ij} & 0 & c_{ij} \end{bmatrix} \quad (4.25)$$

where $L = \{l_i : i \in V\}$ and $l_i = (x_i, y_i)$ is the pixel location of part i . $dx = x_i - x_j$ and $dy = y_i - y_j$ are the displacement of the i th part relative to the j th part. $\Lambda_{i,j}$ can be estimated by expanding Eqn.4.25 and matching up the quadratic terms with Eqn.4.23. The linear terms are absorbed into μ , and don't appear in $\Lambda_{i,j}$. We rewrite Eqn.4.25 as a single quadratic expression:

$$\text{Shape}(L) = -(L - \mu)^T \Lambda (L - \mu) + \text{constant} \quad (4.26)$$

The matrix Λ is PSD if and only if $w^T \Lambda w \geq 0, \forall w$:

$$w^T \Lambda w = \sum_{ij \in E} -a_{ij}(w_1^{ij} - w_3^{ij})^2 - c_{ij}(w_2^{ij} - w_4^{ij})^2 \quad (4.27)$$

where $[w_1^{ij}, w_2^{ij}, w_3^{ij}, w_4^{ij}]$ are the elements of w from the slots corresponding to the i th and j th parts. For a tree-structured E , every part i has only one parent j . This means that each offset $(w_1^{ij} - w_3^{ij})$, $(w_2^{ij} - w_4^{ij})$ can be set to any real number, independent of any other offset. Note that if E contained loopy constraints, offsets are not independent (e.g., triplets of connected parts must obey the triangle inequality). In order to ensure the right-hand-side of Eqn.4.27 is nonnegative for any collection of (arbitrary) offsets, it is necessary and sufficient to set $a_{ij} \leq 0$ and $c_{ij} \leq 0$.

Chapter 5

Face analysis in the wild: A case study

In the last chapter, we introduced part models that share local information with parts. In this chapter, we examine the idea of local sharing by building a part model for face detection, which is perhaps the most well studied of all object detection problems. We propose a unified model for face detection, pose estimation, and landmark estimation (namely FaceDPL for the initials of the three tasks) in real-world, cluttered images. Our model is based on a mixtures of trees with a shared pool of parts; we model every facial landmark as a part and use global mixtures to capture topological changes due to viewpoint. We show that tree-structured models are surprisingly effective at capturing global elastic deformation, while being easy to optimize unlike dense graph structures. We present extensive results on standard face benchmarks, as well as a new “in the wild” annotated dataset, that suggests our system advances the state-of-the-art, sometimes considerably, for all three tasks. Though our model is modestly trained with hundreds of faces, it compares favorably to commercial systems trained with billions of examples (such as Google Picasa and face.com).

5.1 Introduction

The problem of finding and analyzing faces is a foundational task in computer vision. Though great strides have been made in face detection, it is still challenging to obtain reliable estimates of head pose and facial landmarks, particularly in unconstrained “in the wild” images. Ambiguities due to the latter are known to be confounding factors for face recognition [91]. Indeed, even face detection is arguably still difficult for extreme poses.

These three tasks (detection, pose estimation, and landmark localization) have traditionally been approached as separate problems with a disparate set of techniques, such as scanning window classifiers, view-based eigenspace methods, and elastic graph models. In this work, we present a single model that simultaneously advances the state-of-the-art, sometimes considerably, for all three. We argue that a unified approach may make the problem easier; for example, much work on landmark localization assumes images are pre-filtered by a face detector, and so suffers from a near-frontal bias.

Our model is a novel but simple approach to encoding elastic deformation and three-dimensional structure; we use mixtures of trees with a shared pool of parts (see Figure 5.1). We define a “part” at each facial landmark and use global mixtures to model topological changes due to viewpoint; a part will only be visible in certain mixtures/views. We allow different mixtures to share part templates. This allows us to model a large number of views with low complexity. Finally, all parameters of our model, including part templates, modes of elastic deformation, and view-based topology, are discriminatively trained in a max-margin framework.

Notably, most previous work on landmark estimation use densely-connected elastic graphs [83, 19] which are difficult to optimize. Consequently, much effort in the area has focused on optimization algorithms for escaping local minima. We show that multi-view trees are an effective alternative because (1) they can be globally optimized with dynamic programming and (2) surprisingly, they still capture much relevant global elastic structure.



Figure 5.1: We present a unified approach to face detection, pose estimation, and landmark estimation. Our model is based on a mixture of tree-structured part models. To evaluate all aspects of our model, we also present a new, annotated dataset of “in the wild” images obtained from Flickr.



Figure 5.2: Our mixture-of-trees model encodes topological changes due to viewpoint. Red lines denote springs between pairs of parts; note there are no closed loops, maintaining the tree property. All trees make use of a common, shared pool of part templates, which makes learning and inference efficient.

We present an extensive evaluation of our model for face detection, pose estimation, and landmark estimation. We compare to the state-of-the-art from both the academic community and commercial systems such as Google Picasa [3], Face++[4], and face.com [5] (acquired by Facebook).

We first show results in controlled lab settings, using the well-known MultiPIE benchmark [39]. We

definitively outperform past work in all tasks, particularly so for extreme viewpoints. As our results saturate this benchmark, we introduce a new “in the wild” dataset of Flickr images annotated with faces, poses, and landmarks. In all three tasks, our model is on par the best performers, substantially outperforms most of the baselines including even the commercial systems. In order to compare with broader set of existing methods, we also evaluated our model on the popular FDDB face detection benchmark[46]. Our model outperforms all previous methods. It is particularly impressive since our model is trained with hundreds of faces, while other top performers use 20X training data and commercial systems use up to billions of examples [75].

5.2 Related Work

As far as we know, no previous work jointly addresses the tasks of face detection, pose estimation, and landmark estimation. However, there is a rich history of all three in vision. We refer the reader to the recent surveys [91, 60, 88] for a full review. We focus on methods most related to ours.

Face detection is dominated by discriminatively-trained scanning window classifiers [70, 48, 61, 43], most ubiquitous of which is the Viola Jones detector [81] due its open-source implementation in the OpenCV library. Our system is also trained discriminatively, but with much less training data, particularly when compared to commercial systems.

Pose estimation tends to be addressed in a video scenario [91], or a controlled lab setting that assumes the detection problem is solved, such as the MultiPIE [39] or FERET [65] benchmarks. Most methods use explicit 3D models [11, 41] or 2D view-based models [64, 20, 47]. We use view-based models that share a central pool of parts. From this perspective, our approach is similar to aspect-graphs that reason about topological changes between 2D views of an object [14].

Facial landmark estimation dates back to the classic approaches of Active Appearance Models (AAMs) [19, 58] and elastic graph matching [53, 83]. Recent work has focused on global spatial

models built on top of local part detectors, sometimes known as Constrained Local Models (CLMs) [21, 72, 10]. Notably, all such work assumes a densely connected spatial model, requiring the need for approximate matching algorithms. By using a tree model, we can use efficient dynamic programming algorithms to find globally optimal solutions.

From a **modeling perspective**, our approach is similar to those that reason about mixtures of deformable part models [35, 89]. In particular [45] use mixtures of trees for face detection and [32] use mixtures of trees for landmark estimation. Our model simultaneously addresses both with state-of-the-art results, in part because it is aggressively trained to do so in a discriminative, max-margin framework. We also explore part sharing for reducing model size and computation, as in [78, 62].

Subsequent work: [87] [16] [57] [73] are proposed after our model was first published in [94], all of which reported improved performance over the first version of our model in [94]. [87] adopts a model with similar structure to [89]. In addition, it explores contextual information from a person body detector to boost the accuracy of face detection. [73] casts detection as an information retrieval problem. It searches a large pool of faces for the best matching of testing example. [57] learns a set of rigid mixtures using boosted integral channel features. [16] is the only one that does joint detection and landmark localization. It iteratively refines its predictions using cascade regressors. We will compare with these new methods in our experiments.

5.3 Model

Our model is based on mixture of trees with a shared pool of parts V . The model formulation can be found in Section 4.1. We model every facial landmark as a part and use global mixtures to capture topological changes due to viewpoint. We show such mixtures for viewpoint in Fig. 5.2. We will later show that global mixtures can also be used to capture gross deformation changes for a single

viewpoint, such as changes in expression.

Part sharing: Eqn.4.1 requires a separate template w_i^m for each mixture/viewpoint m of part i . However, parts may look consistent across some changes in viewpoint. In the extreme cases, a “fully shared” model would use a single template for a particular part across all viewpoints, $w_i^m = w_i$. We explore a continuum between these two extremes, written as $w_i^{f(m)}$, where $f(m)$ is a function that maps a mixture index (from 1 to M) to a smaller template index (from 1 to M'). We explore various values of M' : no sharing ($M' = M$), sharing across neighboring views, and sharing across all views ($M' = 1$).

Shape model: We compare our learned shape models with standard joint Gaussian models trained generatively with maximum likelihood in Fig.5.3. Those models are commonly used in AAMs and CLMs [21, 72]

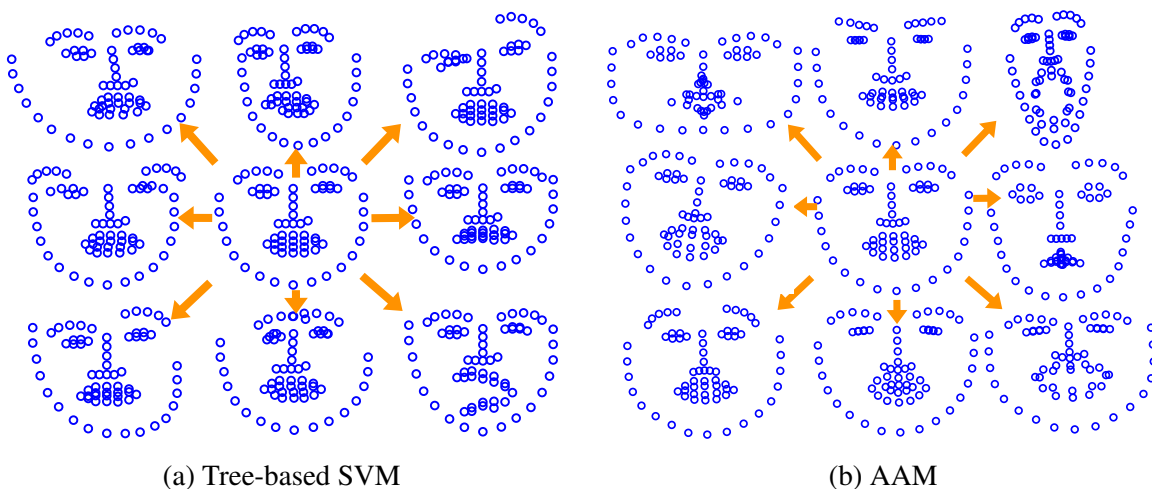


Figure 5.3: In (a), we show the mean shape μ_m and deformation modes (eigenvectors of Λ_m) learned in our tree-structured, max-margin model. In (b), we show the mean shape and deformation modes of the full-covariance Gaussian shape model used by AAMs. Note we exaggerate the deformations for visualization purposes. Model (a) captures much of the relevant elastic deformation, but produces some unnatural deformations because it lacks loopy spatial constraints (e.g., the left corner of the mouth in the lower right plot). Even so, it still outperforms model (b), presumably because it is easier to optimize and allows for joint, discriminative training of part appearance models.

5.4 Inference

Inference corresponds to maximizing $S(I, L, m)$ in Eqn.4.1 (we reproduce it below) over L and m . The best scoring location L^* indicates the landmark locations. The best scoring view-specific mixture m^* tells us the pose.

$$(L^*, m^*) = \operatorname{argmax}_{L, m} S(I, L, m) \quad (5.1)$$

Computation: The total number of distinct part templates in our vocabulary is $M'|V|$. Assuming each part is of dimension D and assuming there exist N candidate part locations, the total cost of evaluating all parts at all locations is $O(DNM'|V|)$. Using distance transforms [35], the cost of message passing is $O(NM|V|)$. This makes our overall model linear in the number of parts and the size of the image, similar to other models such as AAMs and CLMs.

Because the distance transform is rather efficient and D is large, the first term (local part score computation) is the computational bottleneck. A fully independent model uses $M' = M$, while a fully-shared model uses $M' = 1$, roughly an order of magnitude difference. In our experimental results, we show that the shared model may still be practically useful as it sacrifices some performance for speed. This means our *multiview* model can run as fast as a single-view model. Moreover, since single-view CLMs often pre-process their images to compute dense local part scores [72], our multiview model is similar in speed to such popular approaches but globally-optimizable.

5.5 Learning

To learn our model, we assume a fully-supervised scenario, where we are provided positive images with landmark and mixture labels, as well as negative images without faces. We learn both shape and appearance parameters discriminatively using a structured prediction framework. The details



Figure 5.4: Example images from our annotated faces-in-the-wild (AFW) testing set.



Figure 5.5: Example images from MultiPIE with annotated landmarks.

are described in Section 4.1.4.

5.6 Experimental Results

5.6.1 Datasets

CMU MultiPIE: CMU MultiPIE face dataset [39] contains around 750,000 images of 337 people under multiple viewpoints, different expressions and illumination conditions. Facial landmark annotations (68 landmarks for frontal faces (-45° to 45°), and 39 landmarks for profile faces) are

available from the benchmark curators for a relatively small subset of images. In our experiments, we use 900 faces from 13 viewpoints spanning over 180° spacing at 15° for training, and another 900 faces for testing. 300 of those faces are frontal, while the remaining 600 are evenly distributed among the remaining viewpoints. Hence our training set is considerably smaller than those typically used for training face detectors. Fig. 5.5 shows example images from all the 13 viewpoints with the annotated landmarks and face viewpoints. Faces in MultiPIE are almost of the same size around 160×160 .

Fddb: Fddb[46] is a popular benchmark for unconstrained face detection. This dataset contains 2845 images with a total of 5171 faces. It includes a wide range of difficulties including occlusions, difficult poses, , low resolution and out-of-focus faces. The results of a large set of published methods are available on the Fddb website, which makes comparisons to a broader set of baselines possible without having to implement all of them.

Our annotated face in-the-wild (AFW) testset: Fddb only contains the elliptical annotations for face detection. To further evaluate our model, especially its ability to estimate poses and localize landmarks in real world settings, we built an annotated faces in-the-wild (AFW) dataset from Flickr images (Fig. 5.4). Images tend to contain cluttered backgrounds with large variations in both face viewpoint and appearance (aging, sunglasses, make-ups, skin color, expression etc.). Each face is labeled with a bounding box, 6 landmarks (the center of eyes, tip of nose, the two corners and center of mouth) and a discretized viewpoint (-90° to 90° every 15°) along pitch and yaw directions and (left, center, right) viewpoints along the roll direction. We randomly sampled images, keeping each that contained at least one large face. This produced a 205-image dataset with 468 faces. Our dataset differs from similar “in-the-wild” collections [46, 49, 10] in its detailed annotation of multiple, non-frontal faces in a single image.

5.6.2 Our models

Model designs: Our FaceDPL is trained using 900 positive examples from MultiPIE, and 6000 negatives images from the PASCAL VOC 2010 trainval set that do not contain “person”. We model a subset of landmarks defined in MultiPIE as parts. There are 26 parts in near frontal viewpoints and 14 parts in profile viewpoints. Each part is represented as a 5×5 HoG cells with a spatial bin size of 4. We use 13 viewpoints, and build three in-plane rotated mixtures for each viewpoints, yielding a total of $13 \times 3 = 39$ mixtures. The parts are shared across neighboring viewpoints, ending up with a total of 318 parts. For simplicity, we do not enforce symmetry between left/right views.

Computation: On a commodity desktop, our FaceDPL model takes roughly 10 seconds to run on a VGA quality image. We will show in our diagnostic analysis that we can train our model using only 5 viewpoints instead of 13 without sacrificing the accuracy, which would reduce FaceDPL’s runtime to 6 seconds per image. With more sharing, we can future reduce the number of parts by half, and cut the runtime to ~ 3 seconds, with only a small drop in accuracy. Eventually with parallel, cascaded[34] implementations and other smart speedups[31, 25, 86], we believe our models could be real-time.

Sharing: We explore two levels of sharing of our models. They includes: share templates across neighboring viewpoints; share templates across only viewpoints with identical topology. We observe a trade-off between the amount of sharing (thus the number of part templates and running speed) and the performance. We will show the results in our diagnostic analysis in Sec. 5.7.

In-house baselines: In addition to comparing with numerous other systems, we evaluate restricted versions of our approach. We define *Multi.HoG* to be rigid, multiview HoG template detectors, trained on the same data as our models. We define *Star Model* to be the same with our models but defined using a “star” connectivity graph, where all parts are directly connected to a root nose part. This is similar to the popular star-based model of [35], but trained in a supervised manner given landmark locations.

5.6.3 Face detection

FDDDB

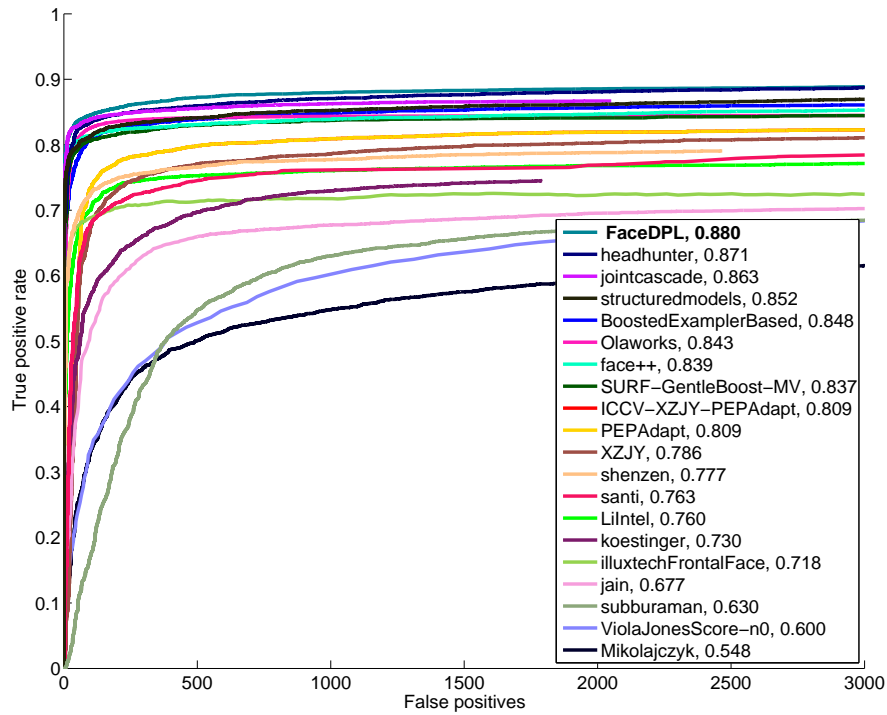
We first show our detection results on the FDDDB benchmark, as MultiPIE consists of centered faces thus is not suitable for evaluating detection.

FDDDB has two evaluation protocols. The discrete (Disc) setting considers a detection to be correct if it overlaps more than 50%, measured by the intersection-over-union ratio, to an annotated elliptical face region. This is similar to the popular PASCAL VOC protocol. In the continuous (Cont) setting, the detections are weighted by their overlap ratios, which encourages stricter overlapping to the labeled groundtruth. We convert our detection to ellipse based on the dataset description. This adjustment improves the overlap and has positive impact on the result curves.

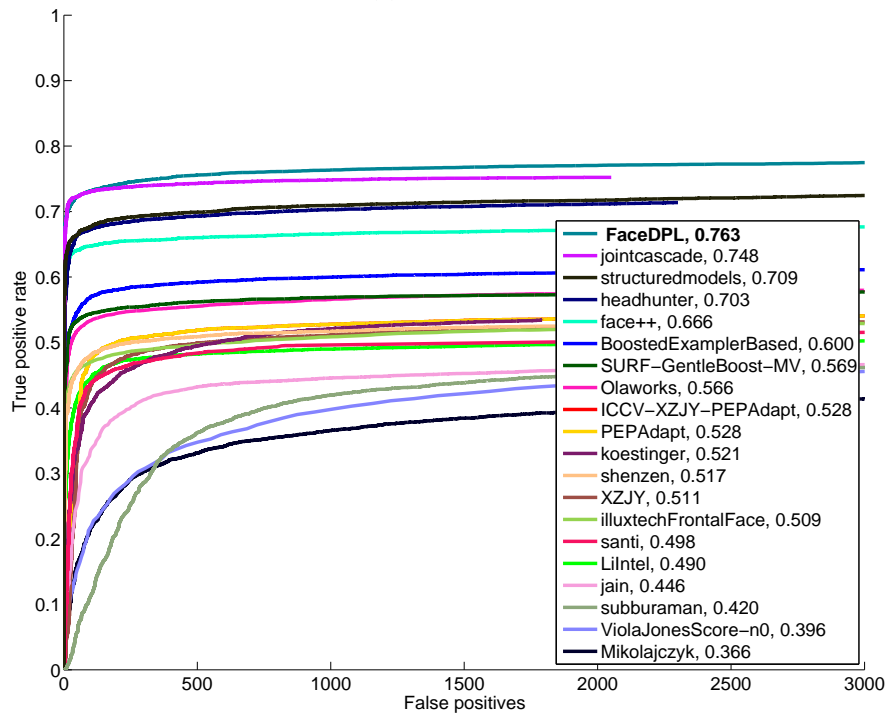
We compare our FaceDPL with a large set of previously published methods whose results can be downloaded from the FDDDB result website[6]. We show the ROC curves of our FaceDPL model and the other methods in Fig. 5.6. Our FaceDPL achieves 88.0% true positive rate at 1000 false positives, compares favorably to all the previous arts and outperforms the commercial systems such as Olaworks (acquired by Intel), Face++ and Illuxtech.

In the discrete curves (Fig. 5.6a), the top competitors such as HeadHunter and Joint Cascade perform very closely to our methods, but they are trained with $\sim 20X$ more faces than our FaceDPL model which is learned with only 900 faces (Fig. 5.7). This demonstrates that our deformable nature of our model allows it to learn the structural variations much more efficiently with a significantly smaller training set. This property will be extremely desirable/beneficial for applications where collecting data is expensive.

In the continuous setting (Fig. 5.6b), our FaceDPL also outperforms the other methods. We observe that the gap between the top two performers (our FaceDPL, JointCascade) and the rest curves is



(a) Disc ROC



(b) Cont ROC

Figure 5.6: ROC curves for face detection on the FDDB benchmark. Our FaceDPL outperforms other methods including commercial systems (the baseline curves are downloaded from [6]) on both the discrete and continuous score evaluation.

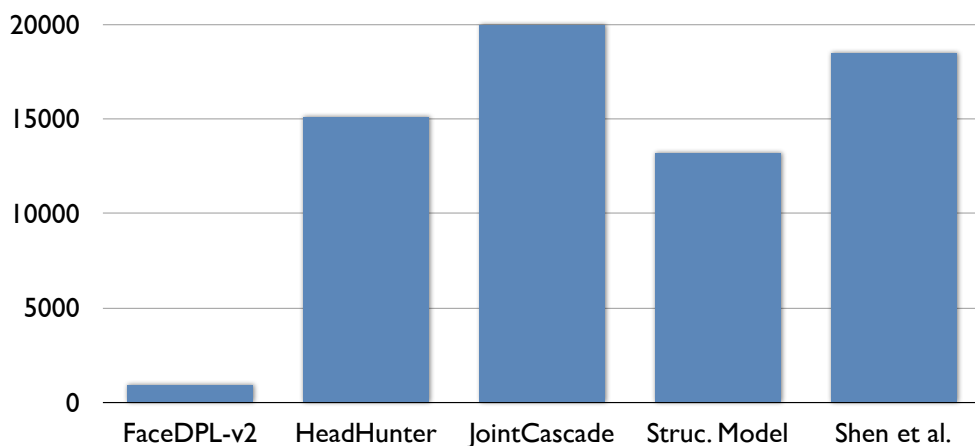


Figure 5.7: We show the number of positive examples used for training by the top performers on FDDB. Our FaceDPL is learned with 900 faces while the other methods use $\sim 20X$ more training examples.

	# missed faces	percentage
Heavily occluded	322	51.7%
Low resolution	266	42.7%
Other	35	5.6%
Total	623	100%

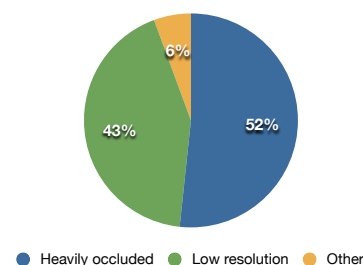


Table 5.1: Categorize the missed faces by our FaceDPL on the FDDB dataset. We show examples of missed faces in Fig. 5.8.

larger than that in the discrete setting. Note that these two methods both jointly detect faces and localize the landmarks. We conjecture that joint detection and landmark localization might help improve the overlap to the groundtruth.

We also noticed that in Fig. 5.6a, the top performers all saturated at almost the same true positive rate (85% \sim 88%), despite the fact that they use very different models and features. This may suggest that all these methods are suffering from common types of mistakes. In order to better understand the errors, we categorized the missed faces of FaceDPL across all 2845 FDDB images by manual inspection. The distribution of missed faces is shown in Tab. 5.1.



Figure 5.8: Examples of missed faces by our model on FDDB. Note that most of them are either heavily occluded faces or have very low resolution due to out-of-focus or simply being too far from the camera.

Our results suggest that 95% of the missed faces are due to two causes: **largely occluded faces** (more than 50% of the face is not visible) and **extremely low resolution** faces that are out-of-focus or too far from the camera. We show some examples in Fig. 5.8. To our surprise, commonly-thought-of difficulties in the literature (illumination / expression / viewpoint variation) collectively contribute to only 5% of the missed detections. Assuming that FDDB represents a realistic scenario for face detection in the wild, our results suggest that occlusion and low-resolution are the true remaining hurdles for face detection. That said, illumination, expression and viewpoint are likely still challenging for other related tasks, such as pose estimation, landmark localization, and face recognition.

AFW

The AFW benchmark adopts the PASCAL VOC precision-recall protocol for object detection (requiring 50% overlap). We compare our approach and baselines with the following: (1) OpenCV frontal + profile Viola-Jones, (2) Boosted frontal + profile face detector of [48], (3) Exemplar SVM

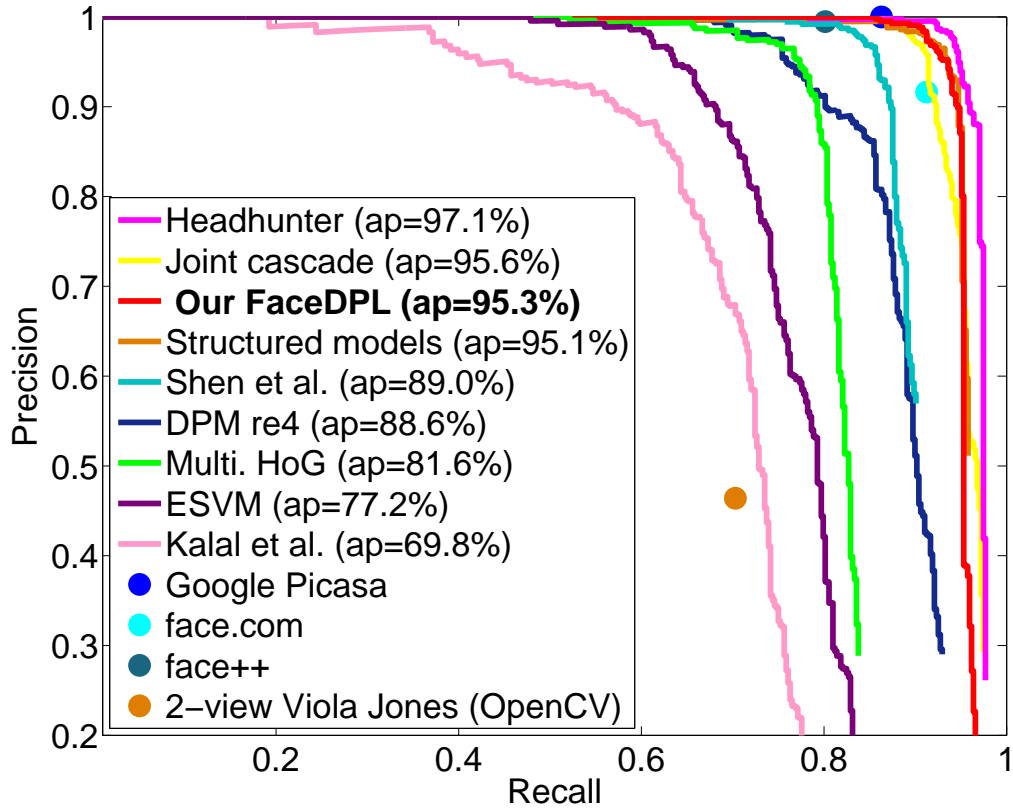


Figure 5.9: Precision-recall curves for face detection on our AFW testset. Our FaceDPL is on par with the best performers and outperforms most of the baselines including commercial systems, such as Google Picasa, Face++ and face.com. Note that the two leading methods, HeadHunter and Joint Cascade are trained with $\sim 20x$ more examples than our models and commercial systems used billions of examples[76].

[56], (4) Multiple rigid HoG templates, (5) Exemplar retrieval by Shen et al. [73], (6) Deformable part model (DPM) [35, 1] trained on same data as our models, (7) Structural model [87], (8) Joint cascade detector [16], (9) Head Hunter [57], (10) Google Picasa’s face detector, manually scored by inspection, (11) Face++’s face detector, (12) face.com’s face detector, which reports detections, viewpoints, and landmarks.

To generate an overly-optimistic multiview detection baseline for (1) and (2), we calibrated the frontal and side detectors *on the test set* and applied non-maximum suppression (NMS) to generate a final set of detections. We also improved (4) and (6) over what’s reported in [94] by optimizing the template resolution and NMS overlapping threshold as suggested in [57].



Figure 5.10: Qualitative results of our model on AFW images, tuned for an equal error rate of false positives and missed detections. We accurately detect faces, estimate pose, and estimate deformations in cluttered, real-world scenes.

Results are summarized in Fig. 5.9. Our FaceDPL outperforms 2-view Viola-Jones and most of the other baselines significantly, and is only slightly below the best performers HeadHunter and Joint Cascade, which are trained with an order of magnitude more examples than we do (Fig. 5.7). Our model also outperforms commercial systems, such as Google Picasa, Face++ and face.com.

Fig. 5.9 reveals an interesting progression of performance. Surprisingly, our rigid multiview HoG baseline outperforms popular face detectors currently in use, achieving an average precision (AP) of 81.6%. Adding latent star-structured parts, making them supervised with tree-structured relations each contributes to performance, with APs of 88.6%, and 95.3% respectively.

5.6.4 Pose estimation

We evaluate pose and landmarks on large faces such that landmarks are visible as higher resolution allows us to ask for more than detection. We plot the curves for all baselines on faces larger than 150 pixels in height (a total of 329, or 70% of the faces in AFW). We argue that high-resolution

images are rather common given HD video and megapixel cameras.

We compare our approach and baselines with the following: (1) Multiview AAMs: we train an AAM for each viewpoint using the code from [51], and report the view-specific model with the smallest reconstruction error on a test image. (2) face.com.

Fig.5.11 shows the cumulative error distribution curves on both datasets. We report the fraction of faces for which the estimated pose is within some error tolerance. Our FaceDPL model, Multi. AAMs and Multi. HoG all score above 99% when allowing $\pm 15^\circ$ error tolerance on MultiPIE, significantly higher than Face.com.

In general, we find that many methods saturate in performance on MultiPIE, originally motivating us to collect AFW.

Unlike on MultiPIE where we assume detections are given (as faces are well centered in image), we evaluate the performance on AFW in a more realistic manner: we evaluate results on faces found by a given algorithm and count missed detections as having an infinite error in pose estimation. Because AAMs do not have an associated detector, we given them the best-possible initialization with the ground-truth bounding box *on the test set* (denoted with an * in Fig.5.11b). assign the ground truth bounding boxes to it to give it best advantage (we mark the result of multiview AAMs with an “*” in Fig.5.11b to indicate this).

All curves decrease in performance in AFW (indicating the difficulty of the dataset), especially multiview AAMs, which suggests AAMs generalize poorly to new data. Our FaceDPL model achieves the best performance, correctly labeling 89.4% of the faces within $\pm 15^\circ$ error tolerance, outperforms face.com and Multiview AAMs by a large margin. Note that we don’t penalize false positives for pose estimation; the Multiview-HoG baseline would perform worse if we penalized false positives as incorrect pose estimates (because they are worse detectors). Our results are impressive given the difficulty of this unconstrained data.

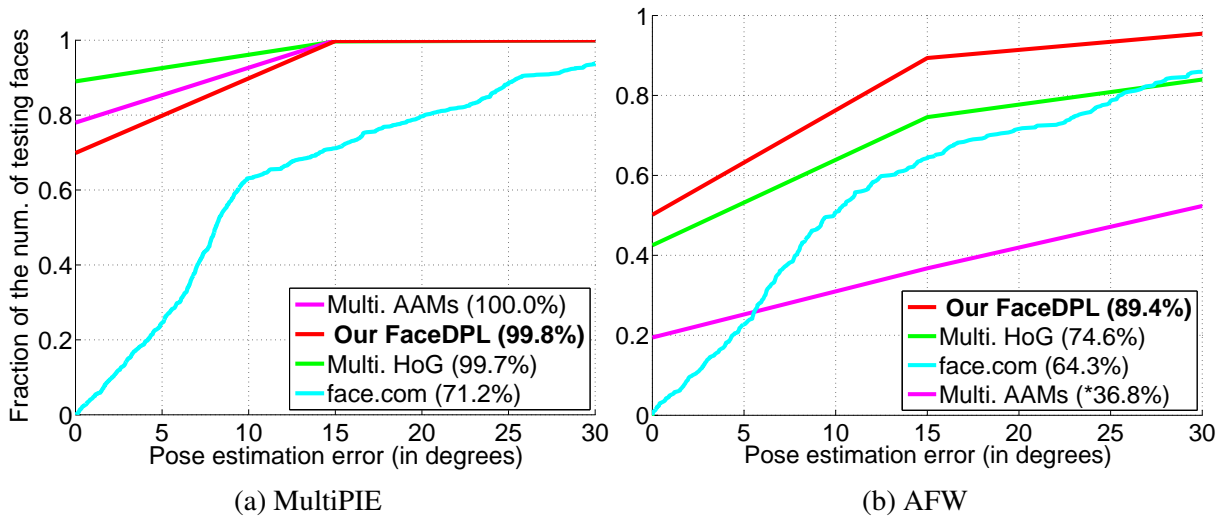


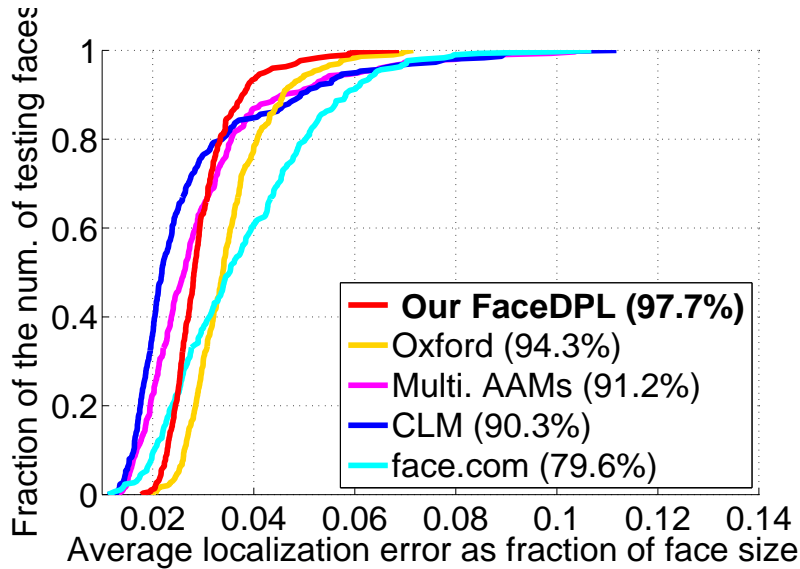
Figure 5.11: Cumulative error distribution curves for pose estimation. The numbers in the legend are the percentage of faces that are correctly labeled within $\pm 15^\circ$ error tolerance. AAMs are initialized with ground-truth bounding boxes (denoted by *). Even so, our FaceDPL models work best on both MultiPIE and AFW.

5.6.5 Landmark localization

We compare our approach and baselines with the following: (1) Multiview AAMs (2) Constrained local model (CLM): we use the off-the-shelf code from [72]. This work represents the current state-of-the-art results on landmark estimation in MultiPIE. (3) face.com reports the location of a few landmarks, we use 6 as output: eye centers, nose tip, mouth corners and center. (4) Oxford facial landmark detector [32] reports 9 facial landmarks: corners of eyes, nostrils, nose tip and mouth corners. (5) Joint Cascade: we re-plot the results on AFW from [16]. Both CLM and multiview AAMs are carefully initialized using the ground truth bounding boxes *on the test set*.

Landmark localization error is often normalized with respect to the inter-ocular distance [10]; this however, presumes both eyes are visible. This is not always true, and reveals the bias of current approaches for frontal faces! Rather, we normalize pixel error with respect to the face size, computed as the mean of height and width.

Various algorithms assume different landmark sets; we train linear regressors to map between these



(a) Localization results on frontal faces from MultiPIE

Figure 5.12: Cumulative localization error curves on the frontal faces from MultiPIE. The numbers in the legend are the percentage of faces whose localization error is less than .05 (5%) of the face size. Our FaceDPL model produces such a small error for 97.7% faces in the testset.

sets. On AFW, we evaluate algorithms using a set of 6 landmarks common to all formats. On MultiPIE, we use the original 68 landmarks when possible, but evaluate face.com and Oxford using a subset of landmarks they report. Note this gives them an extra advantage because their localization error tends to be smaller since they output fewer degrees of freedom.

We first evaluate performance on only frontal faces from MultiPIE in Fig.5.12a. All baselines perform well, but our FaceDPL model still outperforms the state-of-the-art CLM model. When evaluated on all view points, we see a performance drop across most baselines, particularly CLMs (Fig.5.13a). It is worth noting that, since CLM and Oxford are trained to work on near-frontal faces, we only evaluate them on faces between -45° and 45° where all frontal landmarks are visible (marked as a * in Fig.5.13a). Even given this advantage, our model outperforms all baselines by a large margin.

On AFW (Fig.5.13b), we again realistically count missed faces as having a localization error of infinity. We report results on large faces where landmarks are clearly visible (which includes 329

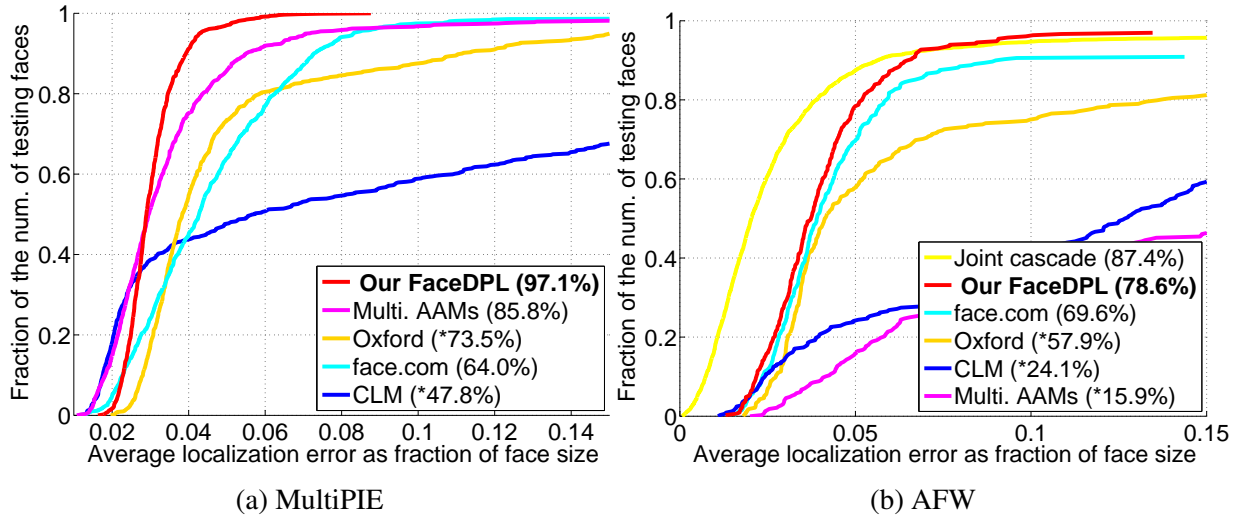


Figure 5.13: Cumulative error distribution curves for landmark localization. The numbers in legend are the percentage of testing faces that have average error below 0.05(5%) of the face size. (*) denote models which are given an “unfair” advantage, such as hand-initialization or a restriction to near-frontal faces (described further in the text). Even so, our FaceDPL models outperform popular baselines on both MultiPIE and our AFW testset.

face instances in AFW testset). Joint cascade achieves the best result with 87.4% of faces having landmark localization error below 5% of face size, followed by our models where FaceDPL gets 78.6%. AAMs and CLM’s accuracy plunges, which suggests these popular methods don’t generalize well to in-the-wild images. We gave an advantage to AAM, CLM, and Oxford by initializing them with ground truth bounding boxes *on the test set* (marked with “*” in Fig.5.13b).

Our models outperform the popular methods in use such as AAM and CLMs on both datasets. We outperform all methods by a large margin on MultiPIE. The large performance gap between two datasets suggest our models maybe overfitting to the lab conditions of MultiPIE; this in turn suggests they may do even better if trained on “in-the-wild” training data similar to AFW as Joint Cascade does. Our model even outperforms commercial systems such as face.com. This result is surprising since our model is only trained with 900 faces, while the latter appears to be trained using billions of faces [76].

We show an example AFW image with large mouth deformations in Fig. 5.14. AAMs mis-estimate

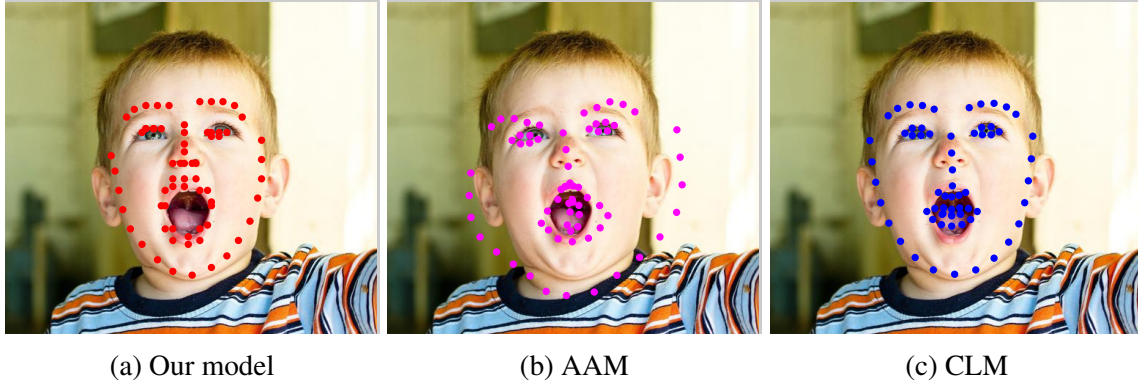


Figure 5.14: An example AFW image with large mouth deformations. AAMs mis-estimate the overall scale in order to match the mouth correctly. CLM matches the face contour correctly, but sacrifices accuracy at the nose and mouth. Our tree-structured model is flexible enough to capture large face deformation and yields the lowest localization error.

the overall scale in order to match the mouth correctly. CLM matches the face contour correctly, but sacrifices accuracy at the nose and mouth. Our tree-structured model is flexible enough to capture large face deformation and yields the lowest localization error.

5.7 Diagnostic analysis and discussion

In this section, we perform diagnostic analysis of our FaceDPL model, examining the impact of various hyper-parameters on detection, pose estimation and landmark localization.

Positive data size: We argue in Chapter 2 that after reaching a modest size, additional increase in the size of the training data yields diminishing returns (similar to a logarithmic curve). We vary the number of positive examples in the range of $\{90, 180, 540, 900\}$ and plot the performance of three tasks in Fig. 5.15. Consistent with our observation in Chapter 2, the detection accuracy does not improve much after 180 examples. Learning with 900 examples improves the final accuracy by only 2% (Fig. 5.15a).

The other two tasks seem to benefit from more training data. In particular, landmark localization

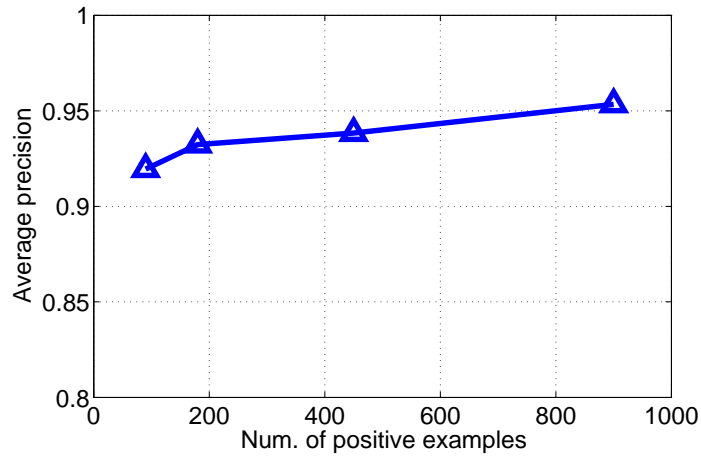
improves by 12% (Fig. 5.15c) when increasing the amount of data from 90 to 900 positives. Accurately localizing landmarks may require capturing subtleties in the appearance of a target landmark, and so may benefit from more training data. This suggests that FaceDPL’s pose estimation and landmark localization accuracy of our FaceDPL models may further improve with more training data (say, of the size used by the majority of related work).

Negative data size: The experiments in Chapter 2 vary the number of positive examples while keeping the negative set fixed. A natural question is to examine the impact of negative training data. Here we fix the number of positive to be 900, and change the size of negatives starting from 250 up to 6000. As Fig. 5.16 shows, although it seems that adding more negatives in general doesn’t hurt, we did not observe a clear trend of improvement either.

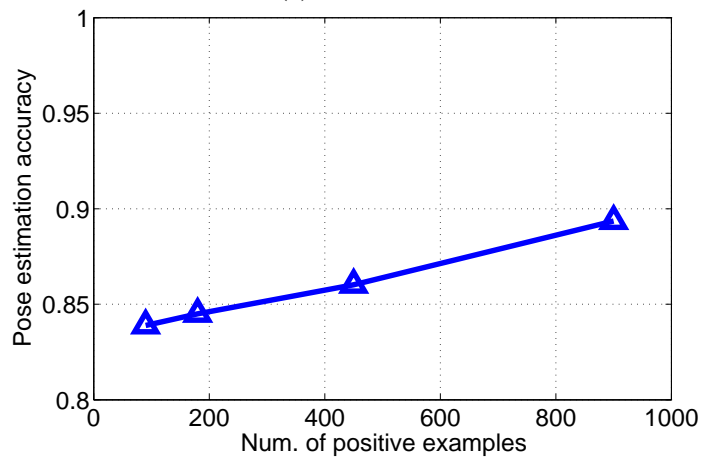
Number of viewpoint-mixtures: We pick subsets of mixtures from our FaceDPL model by uniformly downsampling the 13 viewpoints. This produces mixture sizes of [1, 3, 5, 7, 13] (we ignore the in-plane rotated mixtures here for simplicity), where the single mixture model only contains the frontal view mixture component, all the others contain mixtures from equally spacing viewpoints with varying step size. The results are summarized in Fig. 5.17. Fig. 5.17a shows that we do not lose any detection accuracy if the number of mixtures is reduced from 13 to 5. This behavior is expected as our FaceDPL model can deform itself to capture modest viewpoint changes.

However the accuracy of the other two tasks suffers a noticeable drop. For pose estimation, fewer mixtures imply that the model can only estimate poses in a coarser scale, hurting performance. For landmark localization, the remaining mixtures have to struggle to stretch themselves harder to match the landmarks from the absent viewpoint-specific mixtures.

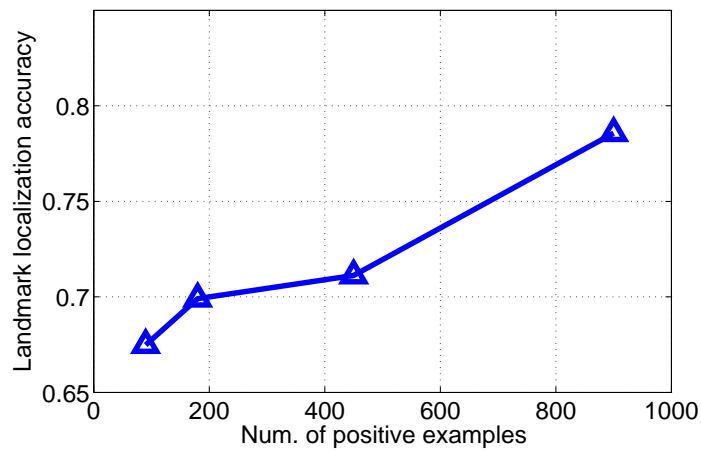
Note that as the parts are shared in our FaceDPL models, the 5-mixture model and the 13-mixture model have exactly the same number of part templates. Because distance transform is rather efficient and the computational bottleneck is scoring part templates (by convolution), adding more mixture components with shared parts is a cheap way to better capture shape and topological variations.



(a) Detection

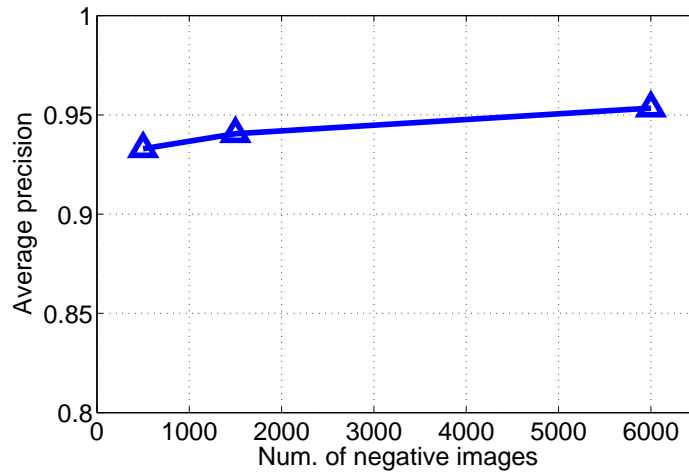


(b) Pose

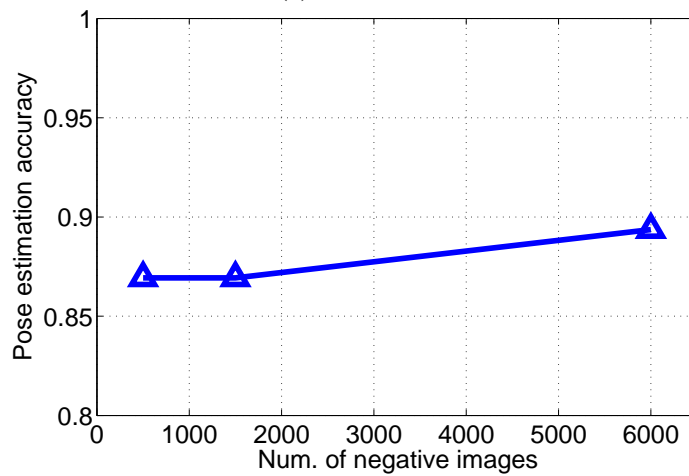


(c) Landmark

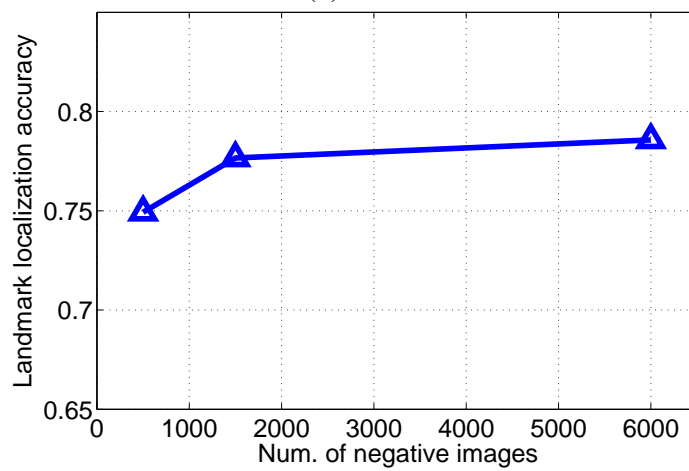
Figure 5.15: Performance with varying number of positive examples. The detection performance saturates early, while the pose estimation and landmark localization accuracy still seem to be growing with more positives in the range we exam.



(a) Detection

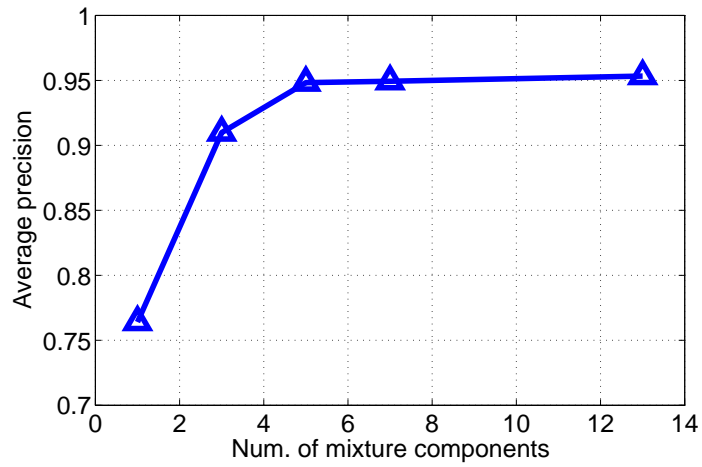


(b) Pose

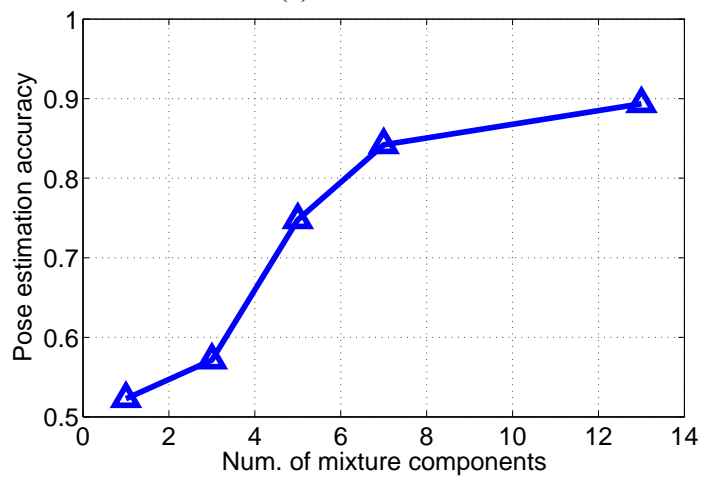


(c) Landmark

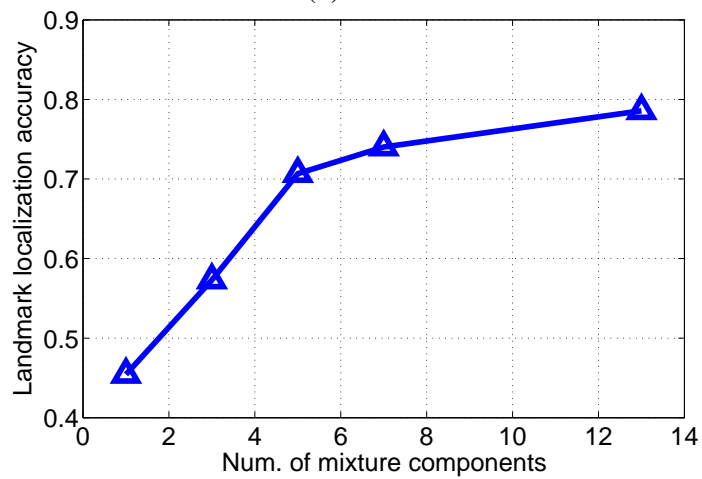
Figure 5.16: Performance with varying number of negative images. There is no clear trend of improvement when growing the number of negative images.



(a) Detection



(b) Pose



(c) Landmark

Figure 5.17: Performance with varying number of mixture components (before in-plane rotation).

Level of sharing:

In this experiment, we explore two levels of sharing as described in Sec. 5.6.2. More sharing reduces the number of parts to 318 to 162. The results in Fig. 5.19 show that more sharing performs worse on all three tasks.

In general, we find that less sharing improves performance. In the extreme case of no sharing, or exemplar parts, we hypothesize that performance would drop (as suggested by the performance of global exemplars in Fig. 5.9). This because sharing acts as a form of “regularization” that prevents overfitting; we can learn a less-noisy nose template if we average in the appearance of other similar-looking (in terms of viewpoints) nose patches.

Finally, another practical benefit of sharing is that one can trade small accuracy loss for possibly large computation reduction by compressing the pool of parts.

Spatial structure (tree vs. star): We replace the learned tree structure with a simple star structure, commonly used to capture object shape (as in [35]). The two structures are illustrated in Fig. 5.18. We compare the performance of both models on all the three tasks on AFW in Fig. 5.19. Stars perform surprisingly well for detection and pose estimation, but perform considerably worse for landmark localization. This suggests that the learned tree structure can better represent the deformation of faces.

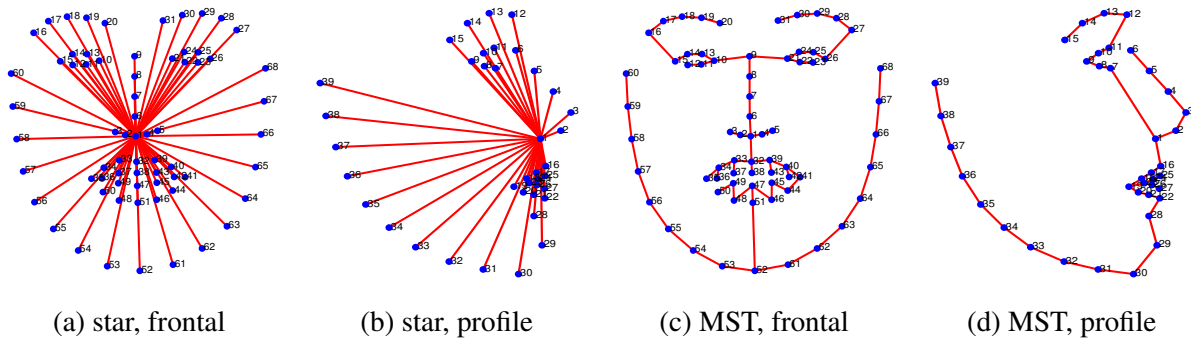


Figure 5.18: Visualization of star structures and the learned tree structures

In-plane rotated mixtures: Removing the ± 30 degrees in-plane rotated mixtures would speed up the model by 3X, and also results in a mild loss in performance (Fig. 5.19).

No extrapolation to unseen shape (EPMs): As described in Chapter 2, Exemplar part models (EPMs) can only represent the shape configurations included in the training data, while FaceDPMs can deform the parts to represent an exponential number of unseen new shapes that EPMs can not extrapolate to.

The results in Fig. 5.19 show that removing our model’s ability to extrapolate would consistently hurt the performance for all three tasks, and especially causes huge accuracy drop in landmark localization as it can not match any shapes that do not appear in the training data.

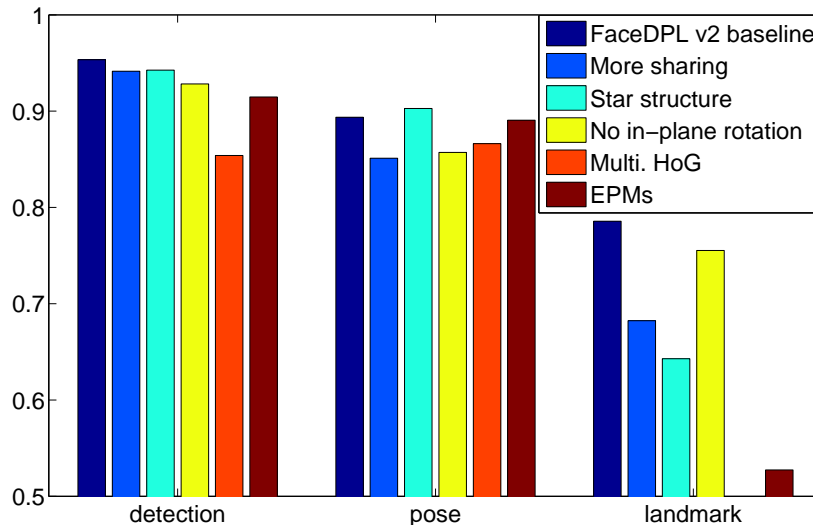


Figure 5.19: Effects on the performance when changing various settings

5.8 Conclusion

We present a unified model for face detection, pose estimation and landmark localization using a mixture of trees with a shared pool of parts. Our tree models are surprisingly effective in capturing global elastic deformation, while being easy to optimize. Our model outperforms state-of-the-art methods, including large-scale commercial systems, on all three tasks under both constrained and

in-the-wild environments. To demonstrate the latter, we present a new annotated dataset which we hope will spur further progress.

Bibliography

- [1] <http://www.cs.brown.edu/~pff/latent/voc-release4.tgz>.
- [2] <http://www.cs.berkeley.edu/~rgb/latent/voc-release5.tgz>.
- [3] <http://picasa.google.com/>.
- [4] www.faceplusplus.com.
- [5] <http://face.com/>.
- [6] <http://vis-www.cs.umass.edu/fddb/results.html>.
- [7] O. Aghazadeh, H. Azizpour, J. Sullivan, and S. Carlsson. Mixture component identification and learning for visual recognition. In *ECCV*, 2012.
- [8] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *ICCV*. IEEE, 2011.
- [9] H. Azizpour and I. Laptev. Object detection using strongly-supervised deformable part models. In *ECCV*, 2012.
- [10] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. In *CVPR 2011*, 2011.
- [11] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *IEEE TPAMI*, 2003.
- [12] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [13] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *International Conference on Computer Vision*, 2009.
- [14] K. Bowyer and C. Dyer. Aspect graphs: An introduction and survey of recent results. *International Journal of Imaging Systems and Technology*, 1990.
- [15] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

- [16] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint cascade face detection and alignment. In *ECCV 2014*, pages 109–122. Springer International Publishing, 2014.
- [17] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE TIT*, 1968.
- [18] C. Chu, S. Kim, Y. Lin, Y. Yu, G. Bradski, A. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. *NIPS*, 19:281, 2007.
- [19] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE TPAMI*, 2001.
- [20] T. Cootes, K. Walker, and C. Taylor. View-based active appearance models. In *IEEE FG 2000*, 2000.
- [21] D. Cristinacce and T. Cootes. Feature detection and tracking with constrained local models. In *BMVC 2006*, 2006.
- [22] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR 2005.*, 2005.
- [23] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- [24] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR*, 2013.
- [25] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 1814–1821. IEEE Computer Society, 2013.
- [26] J. Deng, A. Berg, K. Li, and L. Fei-Fei. What Does Classifying More Than 10,000 Image Categories Tell Us? In *International Conference on Computer Vision*, 2010.
- [27] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009.
- [28] S. K. Divvala, A. A. Efros, and M. Hebert. How important are deformable parts in the deformable parts model? In *European Conference on Computer Vision (ECCV), Parts and Attributes Workshop*, 2012.
- [29] S. K. Divvala, A. A. Efros, and M. Hebert. Object instance sharing by enhanced bounding box correspondence. In *BMVC*, 2012.
- [30] J. Dong, W. Xia, Q. Chen, J. Feng, Z. Huang, and S. Yan. Subcategory-aware object classification. In *CVPR*, 2013.
- [31] C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. In *Computer Vision ECCV 2012*, volume 7574, pages 301–311. Springer Berlin Heidelberg, 2012.

- [32] M. Everingham, J. Sivic, and A. Zisserman. “Hello! My name is... Buffy” – automatic naming of characters in TV video. In *BMVC 2006*, 2006.
- [33] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.
- [34] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2241–2248, June 2010.
- [35] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE TPAMI*, 2010.
- [36] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8(19), September 2012.
- [37] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005.
- [38] V. Ferrari and A. Zisserman. Learning visual attributes. In *NIPS*, Dec. 2007.
- [39] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 2010.
- [40] C. Gu, P. Arbelaez, Y. Lin, K. Yu, and J. Malik. Multi-component models for object detection. In *ECCV*, 2012.
- [41] L. Gu and T. Kanade. 3d alignment of face in a single image. In *CVPR 2006*, 2006.
- [42] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12, 2009.
- [43] B. Heisele, T. Serre, and T. Poggio. A Component-based Framework for Face Detection and Identification. *IJCV*, 2007.
- [44] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *Computer Vision ECCV 2012*, volume 7574, pages 340–353. Springer Berlin Heidelberg, 2012.
- [45] S. Ioffe and D. Forsyth. Mixtures of trees for object recognition. In *CVPR 2001.*, 2001.
- [46] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, UMass, Amherst, 2010.
- [47] M. Jones and P. Viola. Fast multi-view face detection. In *CVPR 2003*, 2003.
- [48] Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC 2008.*, 2008.

- [49] M. Köstinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies 2011*, 2001.
- [50] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.
- [51] D.-J. Kroon. Active shape model and active appearance model. <http://www.mathworks.com/matlabcentral/fileexchange/26706>.
- [52] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, pages 951–958. IEEE, 2009.
- [53] T. Leung, M. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *ICCV 1995*, 1995.
- [54] J. J. Lim, R. Salakhutdinov, and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *NIPS*, 2011.
- [55] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur. Dog breed classification using part localization. In *ECCV*, 2012.
- [56] T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, pages 89–96, 2011.
- [57] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *ECCV*, 2014.
- [58] I. Matthews and S. Baker. Active appearance models revisited. *IJCV*, 60(2):135–164, 2004.
- [59] D. A. McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.
- [60] E. Murphy-Chutorian and M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE TPAMI*, 2009.
- [61] M. Osadchy, Y. L. Cun, and M. L. Miller. Synergistic face detection and pose estimation with energy-based models. *JMLR*, 2007.
- [62] P. Ott and M. Everingham. Shared parts for deformable part-based models. In *CVPR*, pages 1513–1520, june 2011.
- [63] D. Parikh and C. Zitnick. Finding the weakest link in person detectors. In *Computer Vision and Pattern Recognition*, pages 1425–1432. IEEE, 2011.
- [64] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *CVPR 1994.*, 1994.
- [65] P. Phillips, H. Moon, S. Rizvi, and P. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE TPAMI*, 2000.

- [66] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.
- [67] N. Razavi, J. Gall, P. Kohli, and L. V. Gool. Latent hough transforms for object detection. In *ECCV*, 2012.
- [68] E. Rosch. Prototype classification and logical classification: The two systems. *New trends in conceptual representation: Challenges to Piagets theory*, pages 73–86, 1983.
- [69] E. Rosch and C. Mervis. Family resemblances: Studies in the internal structure of categories. *Cognitive psychology*, 7(4):573–605, 1975.
- [70] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE TPAMI*, 1998.
- [71] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR*, pages 1481–1488. IEEE, 2011.
- [72] J. Saragih, S. Lucey, and J. Cohn. Deformable model fitting by regularized landmark mean-shift. *IJCV*, 2011.
- [73] X. Shen, Z. Lin, J. Brandt, and Y. Wu. Detecting and aligning faces by image retrieval. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13*, pages 3460–3467. IEEE Computer Society, 2013.
- [74] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, pages 73–86. Springer, 2012.
- [75] Y. Taigman and L. Wolf. Leveraging Billions of Faces to Overcome Performance Barriers in Unconstrained Face Recognition. *ArXiv e-prints*, Aug. 2011.
- [76] Y. Taigman and L. Wolf. Leveraging billions of faces to overcome performance barriers in unconstrained face recognition. *Arxiv preprint arXiv:1108.1122*, 2011.
- [77] A. Torralba and A. Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition*, pages 1521–1528. IEEE, 2011.
- [78] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE TPAMI*, 29(5):854–869, 2007.
- [79] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280, 2008.
- [80] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613. IEEE, 2009.
- [81] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 2004.

- [82] Y. Wang and G. Mori. A discriminative latent model of object classes and attributes. In *ECCV*, pages 155–168. Springer, 2010.
- [83] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE TPAMI*, Jul 1997.
- [84] Y. Wu and Y. Liu. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102(479):974–983, 2007.
- [85] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. *NIPS*, 17:1537–1544, 2004.
- [86] J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection. June 2014.
- [87] J. Yan, X. Zhang, Z. Lei, and S. Z. Li. Face detection by structural models. *Image and Vision Computing*, 32(10):790 – 799, 2014.
- [88] M.-H. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: a survey. *IEEE TPAMI*, 2002.
- [89] Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *CVPR 2011*, 2011.
- [90] W. Zhang, J. Sun, and X. Tang. Cat head detection - how to effectively exploit shape and texture features. In *ECCV*, 2008.
- [91] W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 2003.
- [92] L. Zhu, Y. Chen, A. L. Yuille, and W. T. Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, 2010.
- [93] X. Zhu, D. Anguelov, and D. Ramanan. Capturing long-tail distributions of object subcategories. In *Computer Vision and Pattern Recognition*, 2014.
- [94] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition*, 2012.
- [95] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes. Do we need more training data or better models for object detection? In *BMVC*, 2012.