

# Do We Need More Training Data?

Xiangxin Zhu · Carl Vondrick · Charless C. Fowlkes · Deva Ramanan

Received: 1 June 2013 / Accepted: 2 March 2015  
© Springer Science+Business Media New York 2015

**Abstract** Datasets for training object recognition systems are steadily increasing in size. This paper 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 ( $\sim 10$  templates and  $\sim 100$  positive training examples per template). This is not a limitation of the feature space as compositional mixtures that share template parameters via parts and that can synthesize new templates not encountered during training yield significantly better performance. Based on our analysis, we conjecture that the greatest gains in detection performance will continue to derive from

improved representations and learning algorithms that can make efficient use of large datasets.

**Keywords** Object detection · Mixture models · Part models

## 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. 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 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 (Halevy et al. 2009). 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 absent the development of more complex object detection models?

To tackle this question, we collected a massive training set that is an order of magnitude larger than existing collections such as PASCAL (Everingham et al. 2010). We follow the dominant paradigm of scanning-window templates trained with linear SVMs on HOG features (Dalal and Triggs 2005; Felzenszwalb et al. 2010; Bourdev and Malik 2009; Malisiewicz et al. 2011), and evaluate detection performance as a function of the amount of training data and the model complexity.

---

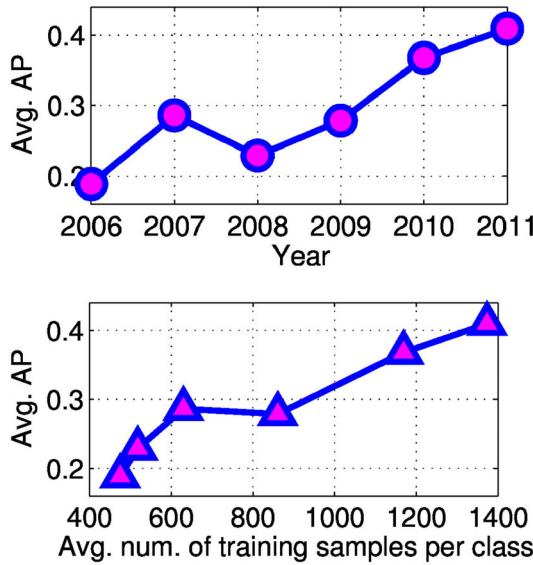
Communicated by Antonio Torralba and Alexei Efros.

X. Zhu (✉) · C. C. Fowlkes · D. Ramanan  
Department of Computer Science, UC Irvine, Irvine, USA  
e-mail: xzhu@ics.uci.edu

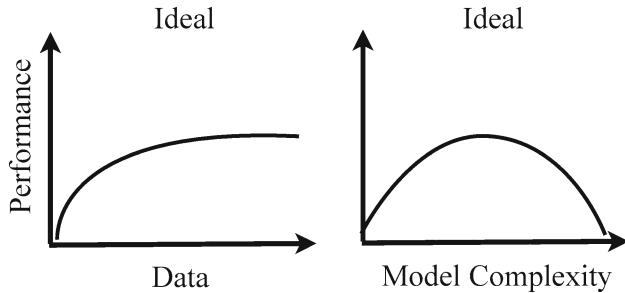
C. C. Fowlkes  
e-mail: fowlkes@ics.uci.edu

D. Ramanan  
e-mail: dramanan@ics.uci.edu

C. Vondrick  
CSAIL, MIT, Cambridge, USA  
e-mail: vondick@mit.edu



**Fig. 1** The best reported performance on PASCAL VOC challenge has shown marked increases since 2006 (top). This could be due to various factors: the dataset itself has evolved over time, the best-performing methods differ across years, etc. In the bottom-row, 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?



**Fig. 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. (McAllester 1999)

### 1.1 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). 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 non-parametric models that grow with the amount of training data, we quickly encountered diminishing returns in performance with only modest amounts of training data.

We show that the apparent performance ceiling 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 non-parametric mixture models. However, DPMs have fixed complexity and still saturate in performance with current amounts of training data, even when scaled to mixtures of DPMs. This difficulty is further exacerbated by the computational demands of non-parametric mixture models, which can be impractical for many applications.

### 1.2 Proposed Solutions

In this paper, 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 (Felzenszwalb et al. 2010), appearance features (Divvala et al. 2012), and keypoint labels (Bourdev and Malik 2009) 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.

To efficiently take advantage of additional training data, we introduce a non-parametric extension of a DPM which we call an exemplar deformable part model (EDPM). Notably, EDPMs increase the expressive power of DPMs with only a negligible increase in computation, making them practically useful. We provide evidence that suggests that compositional representations of mixture templates provide an effective way to help target the “long-tail” of object appearances by sharing local part appearance parameters across templates.

Extrapolating beyond our experiments, we see the striking difference between classic mixture models and the non-parametric compositional model (both mixtures of linear classifiers operating on the same feature space) as evidence that the greatest gains in the near future will not be had with simple models + bigger data, but rather through improved representations and learning algorithms.

We introduce our large-scale dataset in Sect. 2, describe our non-parametric mixture models in Sect. 3, present extensive experimental results in Sect. 4, and conclude with a discussion in Sect. 5 including related work.

## 2 Big Detection Datasets

Throughout the paper 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.

### 2.1 PASCAL

Our first dataset is a newly collected data set that we refer to as PASCAL-10X and describe in detail in the following section.<sup>1</sup> This dataset covers the 11 PASCAL categories (see Fig. 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 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 10,000+ images.

### 2.2 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 (Gross et al. 2010), 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 (Dalal and Triggs 2005). Detection accuracy of face models are evaluated on the annotated face in-the-wild (AFW) (Zhu and Ramanan 2012), which contains images from real-world environments and tend to have cluttered backgrounds with large variations in both face viewpoint and appearance.

### 2.3 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 Table 1. We built

**Table 1** PASCAL 2010 trainval and our data set for select categories

Category	PASCAL 2010		Our data set	
	Images	Objects	Images	Objects
Bicycle	471	614	5027	7401
Bus	353	498	3405	4919
Cat	1005	1132	12,204	13,998
Cow	248	464	3194	6909
Dining table	415	468	3905	5651
Horse	425	621	4086	6488
Motorbike	453	611	5674	8666
Sheep	290	701	2351	6018
Sofa	406	451	4018	5569
Train	453	524	6403	7648
TV monitor	490	683	5053	7808
Totals	4609	6167	50,772	81,075

Our data set is an order of magnitude larger

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.

#### 2.3.1 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.

#### 2.3.2 Filtering

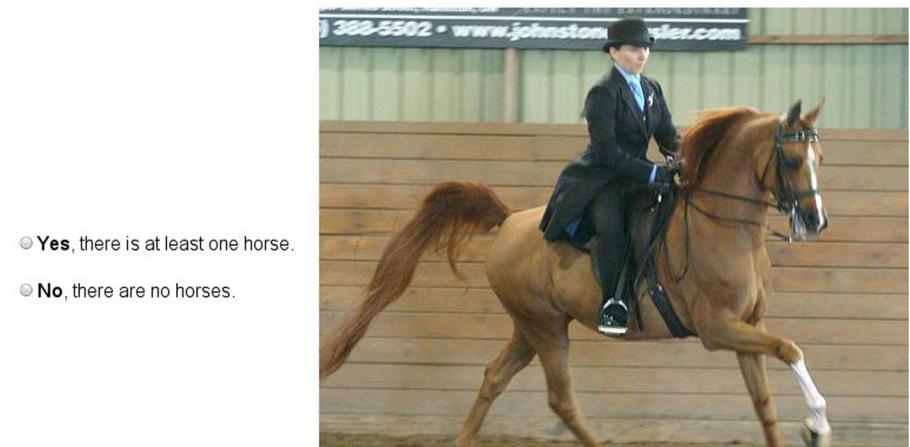
The downloaded images from Flickr did not necessarily contain objects for the category that we were targeting. We created MTurk tasks that asked workers to classify the downloaded images on whether they contained the category of interest. Our user interface in Fig. 3 gave workers instructions on how to handle special cases and this resulted in acceptable annotation quality without finding agreement between workers.

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

**Fig. 3** Our MTurk user interfaces for image classification and object annotation. We provided detailed instructions to workers, resulting in acceptable annotation quality

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

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



- 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**.

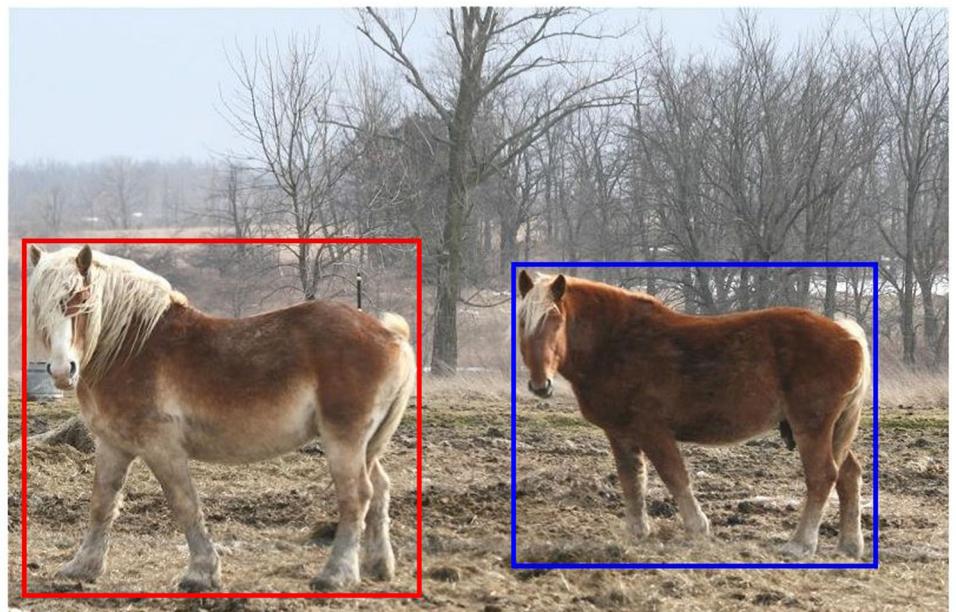


Image does not contain **horses**.

[Start Over](#) Your work will directly impact active research. [Submit HIT »](#)

### 2.3.3 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. 3, although many workers gave us more boxes. On average, our system received annotations at 3 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.

## 2.4 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.

### 2.4.1 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 Table 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

**Table 2** Frequencies of attributes (percent) across images in our 10x horse data set compared to the PASCAL 2010 train-val data set

Attributes	Us	PASCAL	
		2010	2007
Truncated	30.8	31.5	<b>15.8</b>
Occluded	5.9	8.6	7.1
Jumping	4.0	4.3	<b>15.8</b>
Standing	69.9	68.8	<b>54.6</b>
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	<b>57.5</b>
Person Besides	8.8	10.0	8.6
No Person	66.0	59.8	<b>33.8</b>

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

those in PASCAL so we scaled our images down to construct our final dataset.

### 2.4.2 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.

### 2.4.3 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.

## 3 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. In this section, we describe various approaches for learning and representing mixture models. 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:

$$S(I) = \max_m \left[ w_m \cdot \phi(I) + b_m \right], \quad (1)$$

where  $m$  is a discrete mixture variable,  $\phi(I)$  is a HOG image descriptor (Dalal and Triggs 2005),  $w_m$  is a linearly-scored template, and  $b_m$  is an (optional) bias parameter that acts as a prior that favors particular templates over others.

### 3.1 Independent Mixtures

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.

### 3.1.1 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.

```

Input:  $\{N_n\}; \{S^{(i)}\}$ 
Output:  $\{C_n^{(i)}\}$ 
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  $z$ th cluster
       without replacement
6   end
7 end

```

**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 samples  $N_n$  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.

### 3.1.2 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 sub-sample 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 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.

### 3.1.3 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. Figs. 4 and 5 show example clusters for faces and buses.

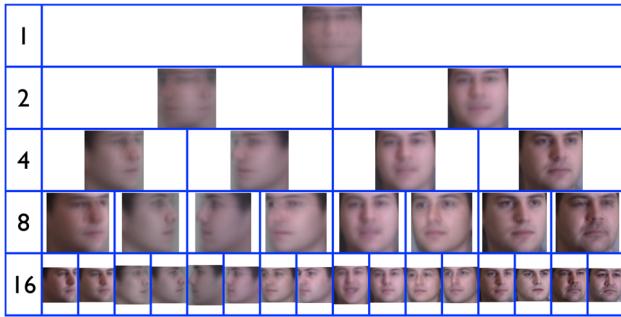
## 3.2 Compositional Mixtures

In this section, we describe various architectures for compositional mixture models that share information between mixture components. We share local spatial regions of templates, or parts. We begin our discussion by reviewing standard architectures for DPMs, and show how they can be interpreted and extended as high-capacity mixture models.

### 3.2.1 Deformable Part Models (DPMs)

We begin with an analysis that shows that DPMs are equivalent to an exponentially-large mixture of rigid templates Eq. (1). 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:

$$S_{DPM}(I) = \max_{x, y} S(I, x, y) \quad \text{where}$$



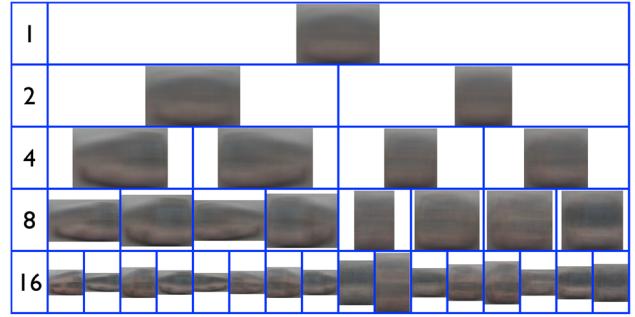
(a) Unsupervised

(b) Supervised

**Fig. 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

$$S(I, x, y) = \sum_{i=1}^P \sum_{(u, v) \in W_i} \alpha_i[u, v] \cdot \phi(I, x_i + u, y_i + v) + \sum_{ij \in E} \beta_{ij} \cdot \psi(x_i - x_j - a_{ij}^{(x)}, y_i - y_j - a_{ij}^{(y)}), \quad (2)$$

where  $W_i$  defines the spatial extent (length and width) of part  $i$ . The first term defines a local appearance score, where  $\alpha_i$  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,



(a) Unsupervised

(b) Supervised

**Fig. 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. 11

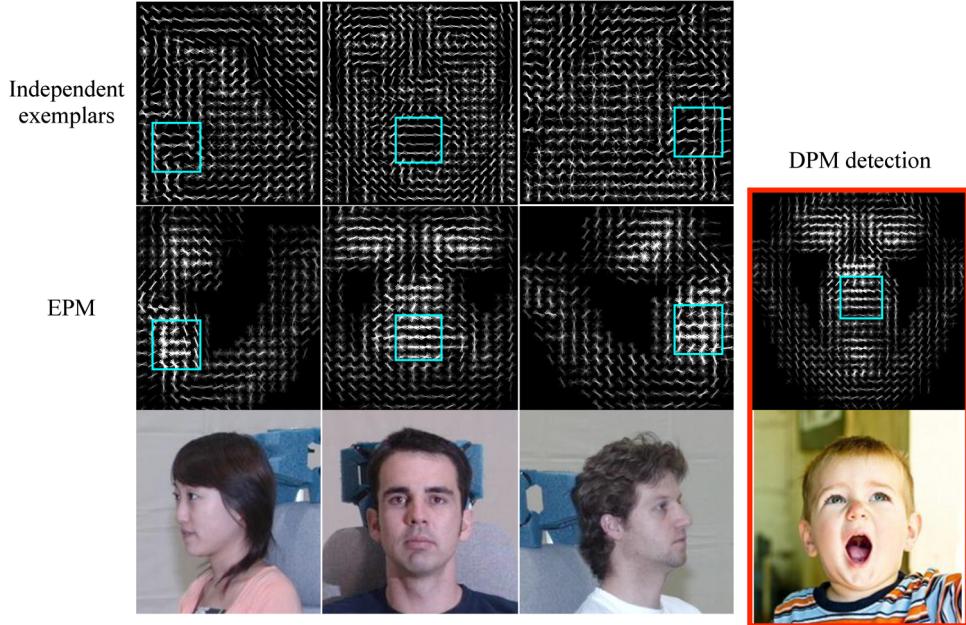
where  $\Omega$  is the space of possible part placements. Given that each of  $P$  parts can be placed at one of  $L$  locations,  $|\Omega| = L^P \approx 10^{20}$  for our models.

By defining index variables in image coordinates  $u' = x_i + u$  and  $v' = y_i + v$ , we can rewrite Eq. (2) as:

$$S(I, x, y) = \sum_{u', v'} \sum_{i=1}^P \alpha_i[u' - x_i, v' - y_i] \cdot \phi(I, u', v') + \sum_{ij \in E} \beta_{ij} \cdot \psi(x_i - x_j - a_{ij}^{(x)}, y_i - y_j - a_{ij}^{(y)}) = \left( \sum_{u', v'} w(x, y)[u', v'] \cdot \phi(I, u', v') \right) + b(x, y) = w(x, y) \cdot \phi(I) + b(x, y), \quad (3)$$

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  $b(x, y)$



**Fig. 6** Classic exemplars vs EPMs. On the *top row*, we show three rigid templates trained as independent exemplar mixtures. 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

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.

### 3.2.2 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

$$S_{EPM}(I) = \max_{(x,y) \in \Omega_N} S(I, x, y) \quad \text{where } \Omega_N \subseteq \Omega. \quad (4)$$

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 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

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. 14, 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 (Color figure online)

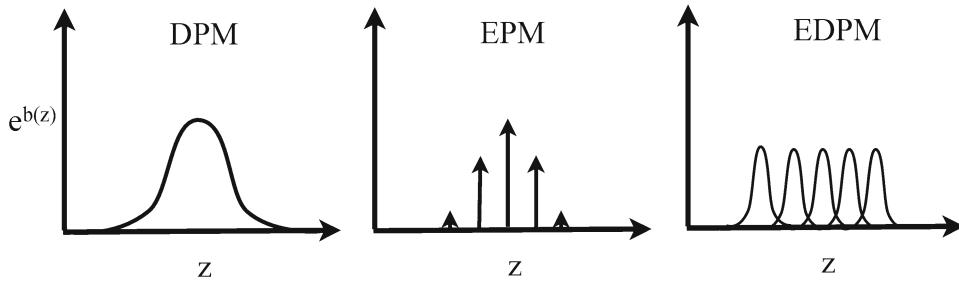
of sharing, but cannot synthesize new templates that were not present in the training data. We visualize example EPM templates in Fig. 6.

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.

### 3.2.3 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:

$$S(I) = \max_{m \in \{1 \dots M\}} \max_{z \in \Omega} \left[ w(z) \cdot \phi(I) + b_m(z) \right], \quad (5)$$



**Fig. 7** 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

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 Eq. (3). Because the appearance model does not depend on  $m$ , we can write:

$$S(I) = \max_{z \in \Omega} [w(z) \cdot \phi(I) + b(z)], \quad (6)$$

where  $b(z) = \max_m b_m(z)$ . Interestingly, we can write the DPM, EPM, and EDPM in the form of Eq. (6) by simply changing the shape model  $b(z)$ :

$$b_{DPM}(z) = \sum_{ij \in E} \beta_{ij} \cdot \psi(z_i - z_j - a_{ij}) \quad (7)$$

$$b_{EDPM}(z) = \max_{m \in \{1, \dots, M\}} \sum_{ij \in E} \beta_{ij} \cdot \psi(z_i - z_j - a_{ij}^m) \quad (8)$$

$$b_{EPM}(z) = b_{DPM}(z) + b_{EDPM}^*(z), \quad (9)$$

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  $\beta$  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. 7.

### 3.2.4 Inference

We now show that inference on EDPMs (Eq. 8) 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 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:

$$S_{EDPM}(I) = \max_{z_1, m} [\alpha_1 \cdot \phi(I, z_1) + \sum_{j > 1} m_j(z_1 + a_{1j}^m)] \quad (10)$$

$$m_j(z_1) = \max_{z_j} [\alpha_j \cdot \phi(I, z_j) + \beta_{1j} \cdot \psi(z_1 - z_j)]. \quad (11)$$

The maximization in Eq. (11) needs only be performed once across mixtures, and can be computed efficiently with a single min-convolution or distance transform (Felzenszwalb and Huttenlocher 2012). The resulting message is then shifted by mixture-specific anchor positions  $a_{1j}^m$  in Eq. (10). 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. 7).

## 4 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 train-val set and faces from the Annotated Faces in the Wild test set (Zhu and Ramanan 2012). For each category, we train the model with varying number of samples ( $N$ ) and mix-

tures ( $K$ ). To train our independent mixtures, we learn rigid HOG templates (Dalal and Triggs 2005) with linear SVMs (Chang and Lin 2011). We calibrated SVM scores using Platt scaling (Platt 1999). 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 train our compositional mixtures, we use a locally-modified variant of the codebase from (Felzenszwalb et al. 2010). To show the uncertainty of the performance with respect to different sets of training samples, we randomly re-sample the training data five times for each  $N$  and  $K$  following the partitioned sampling scheme described in Sect. 3. The best regularization parameter  $C$  for the SVM was selected by cross validation. For diagnostic analysis, we first focus on faces and buses.

#### 4.1 Evaluation

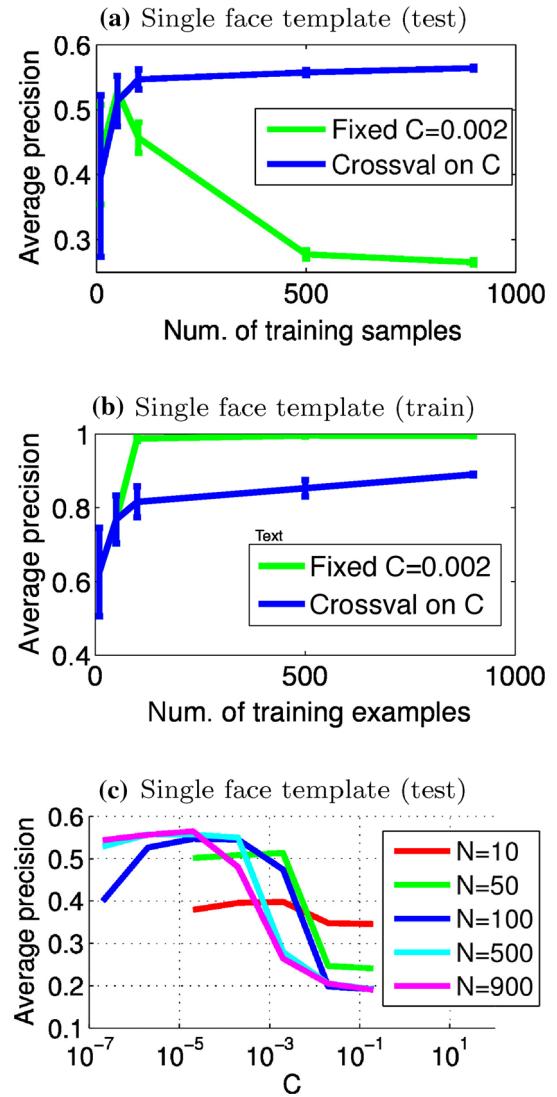
We adopt the PASCAL VOC precision-recall protocol for object detection (requiring 50 % overlap), and report average precision (AP). While learning theory often focuses on analyzing 0–1 classification error rather than AP (McAllester 1999), we experimentally verified that AP typically tracks 0–1 classification error and so focus on the former in our experiments.

#### 4.2 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. 8 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. 8a). 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 (Felzenszwalb et al. 2010; Dalal and Triggs 2005) may also be suffering from suboptimal regularization and might show an improvement by proper cross-validation.

#### 4.3 The Importance of Clean Training Data

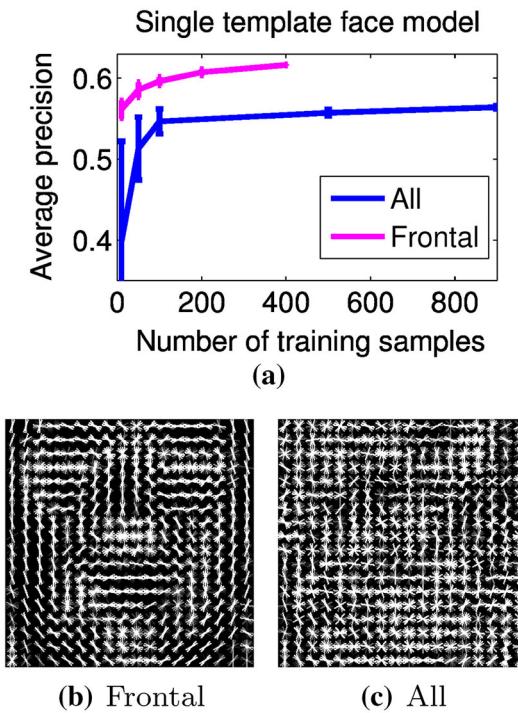
Although proper regularization parameters proved to be crucial, we still discovered scenarios where additional training data hurt performance. Fig. 9 shows an experiment with a



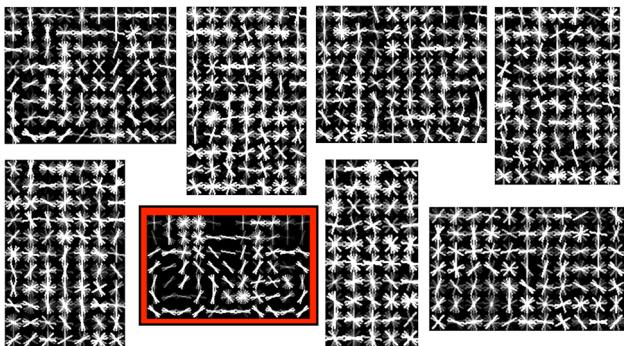
**Fig. 8** **a** More training data could hurt if we did not cross-validate to select the optimal  $C$ . **b** Training error, when measured on a fixed training set of 900 faces and 1218 negative images, always decreases as we train with more of those images. This further suggests that overfitting is 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  $N$

fixed set of  $N$  training examples where we train two detectors: (1) *All* is trained 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

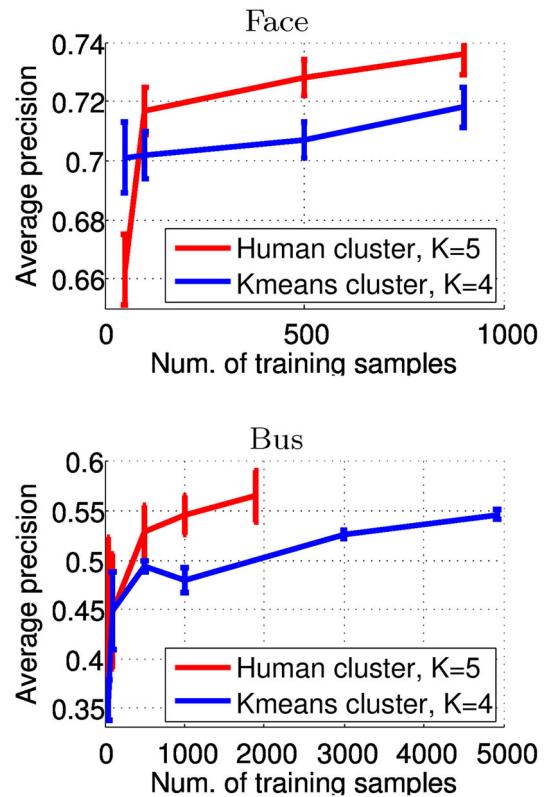


**Fig. 9** In a, we compare the performance of a single HOG template trained with  $N$  multi-view face examples, versus a template trained with a subset of those  $N$  examples corresponding to frontal faces. The frontal-face template b looks “cleaner” and makes fewer classification errors on both testing 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



**Fig. 10** The single bicycle template (marked with red) alone achieves  $ap=29.4\%$ , which is almost equivalent to the performance of using all eight 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 (Color figure online)

same training images (900 vs 470). 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.



**Fig. 11** 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 %

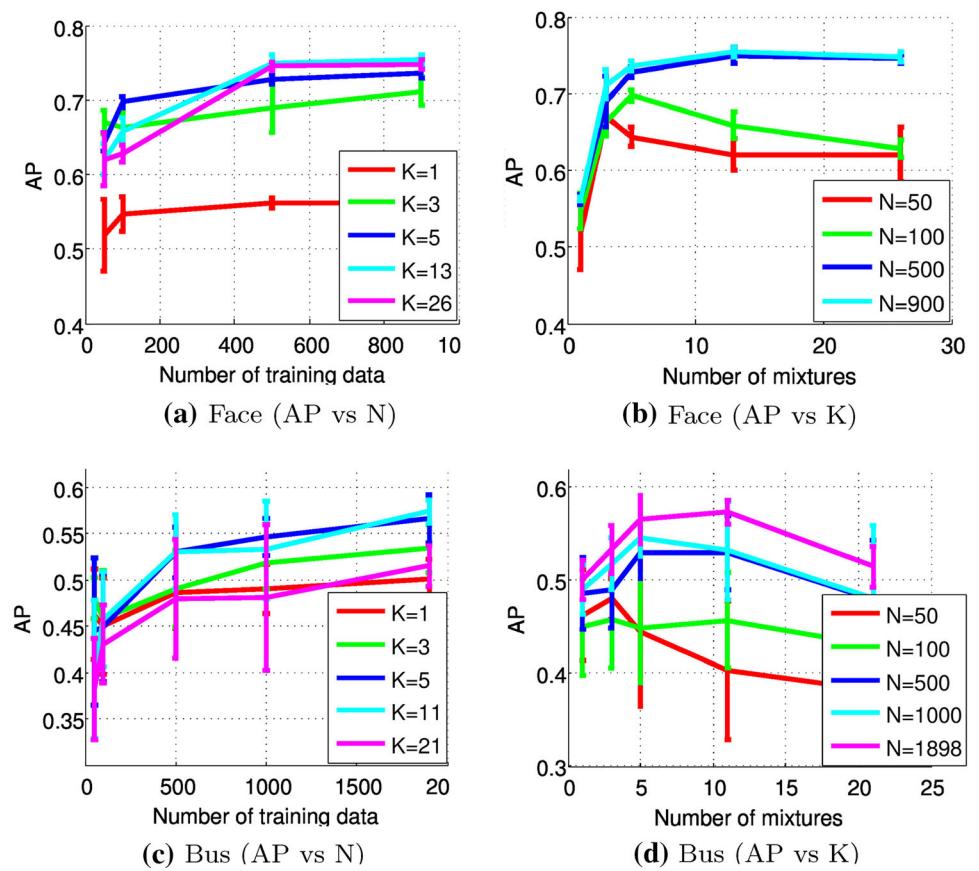
Interestingly, latent mixture models can mimic the behavior of non-convex bounded loss functions (Wu and Liu 2007) by placing noisy examples into junk clusters that simply serve to explain outliers in the training set. In some cases, a single “clean” mixture component by itself explains most of the test performance (Fig. 10).

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. 11, we see a small 2–5 % 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 (Felzenszwalb et al. 2010) 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.

#### 4.4 Performance of Independent Mixtures

Given the right regularization and clean mixtures trained independently, we now evaluate whether performance asymp-

**Fig. 12** **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



totes as the amount of training data and the model complexity increase.

Figure 12 shows performance as we vary  $K$  and  $N$  after cross-validating  $C$  and using supervised clustering. Fig. 12a 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  $\sim 100$  training examples per mixture and with  $\sim 10$  mixtures. Fig. 12b 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: 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.

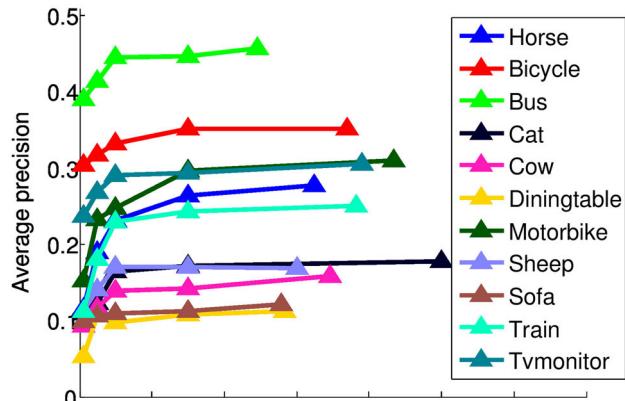
We performed similar experiments for all 11 PASCAL object categories in our PASCAL-10X dataset shown in Fig. 13. We evaluate performance on the PASCAL 2010 train-val set since the testset annotations are not public. We cluster the training data into  $K = [1, 2, 4, 8, 16]$  mixture components, and  $N = [50, 100, 500, 1000, 3000, N_{max}]$  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. 13a, 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 scale (Fig. 13b), 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. 13b (motorbike), we will need  $10^{12}$  motorbike samples to reach 95 % AP!

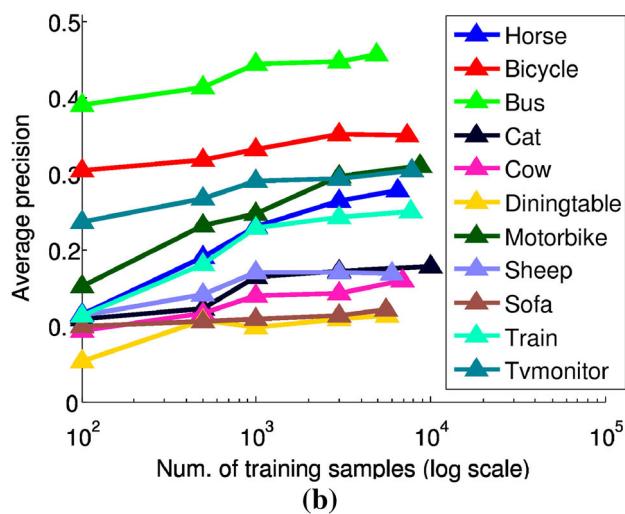
Of course 95 % AP may not be an achievable level of performance. There is some upper-bound imposed by the Bayes risk associated with the HOG feature space which no amount of training data will let us surpass. *Are classic mixtures of rigid templates approaching the Bayes optimal performance?* Of course we cannot compute the Bayes risk so this is hard to answer in general. However, the performance of any system operating on the same data and feature space provides a lower bound on the optimal performance. We next analyze the performance of compositional mixtures to provide much better lower bound on optimal performance.

#### 4.5 Performance of Compositional Mixtures

We now perform a detailed analysis of compositional mixture models, including DPMs, EPMs, and EDPMs. We focus



(a)



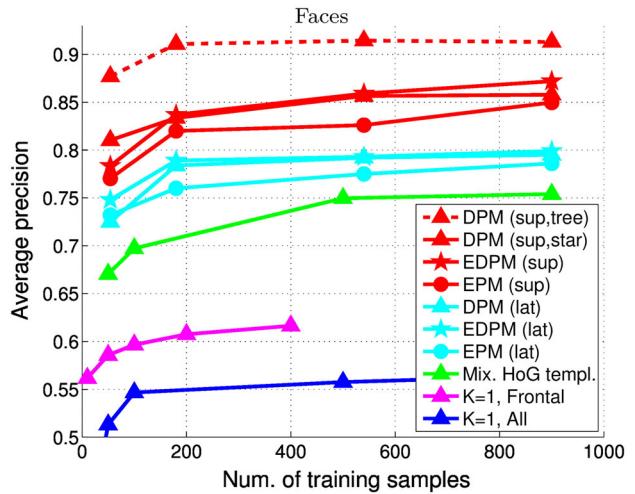
(b)

**Fig. 13** 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 95 % AP if we keep growing at the same rate

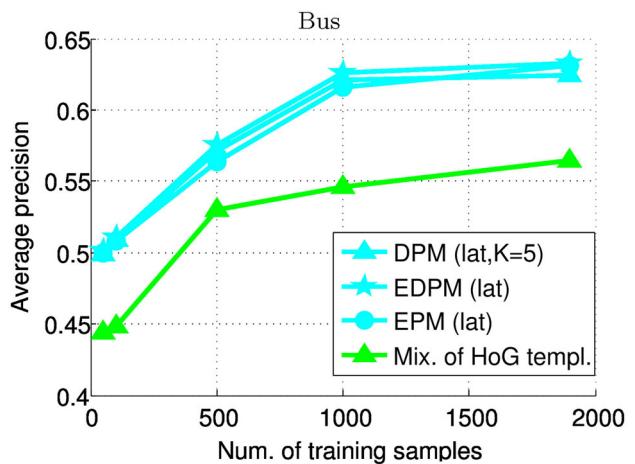
on face detection and Pascal buses. We consider the latent star-structured DPM of (Felzenszwalb et al. 2010) as our primary baseline. For face detection, we also compare to the supervised tree-structured DPM of (Zhu and Ramanan 2012), which uses facial landmark annotations in training images as supervised 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. 14 and buses in Fig. 15.

#### 4.5.1 Supervised DPMs

For face detection, we first note that a supervised DPM can perform quite well (91 % AP) with less than 200 example



**Fig. 14** 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. 9). 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



**Fig. 15** 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

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 cross-validated rigid mixture 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. To analyze where this performance gap is coming from, we now evaluate the performance of various compositional mixtures models.

#### 4.5.2 Latent Parts

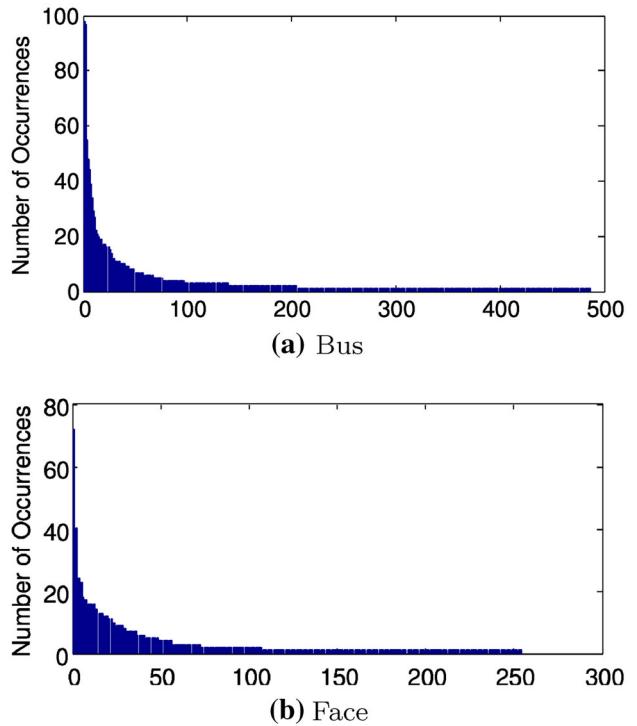
We begin by analyzing the performance of compositional mixtures defined by latent parts, as they can be constructed for both faces and Pascal buses. Recall that EPMs have the benefit of sharing parameters between rigid templates, but they cannot extrapolate to new shape configurations not seen among the  $N$  training examples. EPMs noticeably improve performance over independent mixtures, 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. But interestingly, with a modest number of observed shapes ( $\approx 1000$ ), exemplar-based methods with parameter sharing can approach the performance of DPMs. This in turn suggests that extrapolation to unseen shapes is may not be crucial, at least in the latent case. This is further evidenced by the fact that EDPMs, the deformable counterpart to EPMs, perform similarly to both EPMs and DPMs.

#### 4.5.3 Supervised Parts

The story changes somewhat for supervised parts. Here, supervised EPMs outperform independent mixtures 85–76 %. Perhaps surprisingly, EPMs even outperform latent DPMs. However, supervised EPMs still 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. (Zhu and Ramanan 2012) 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.

#### 4.5.4 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 tem-



**Fig. 16** We plot the number of distinct shape patterns in our training set of buses and faces. Each training example is “binned” into a discrete shape by quantizing a vector of part locations. The above histograms count the number of examples that fall into a particular shape bin. In both cases, the number of occurrences seems to follow a long-tail distribution: a small number of patterns are common, while there are a huge number of rare cases. Interestingly, there are less than 500 unique bus configurations observed in our PASCAL-10X dataset of 2000 training examples. This suggests that one can build an exemplar part model (EPM) from the “right” set of 500 training examples and still perform similarly to a DPM trained on the full dataset (Fig. 15)

plates has the potential to be much more significant. We see this as a consequence of the “long-tail” distribution of object shape (Fig. 16); many object instances can be modeled with a few shape configurations, but there exists of long tail of unusual shapes. Examples from the long tail may be difficult to observe in any finite training dataset, suggesting that extrapolation is crucial for recognizing these cases. 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 (Zhu and Ramanan 2012) with 50 face images.

#### 4.5.5 Relation to Exemplar SVMs

In the setting of object detection, we were not able to see significant performance improvements due to our non-parametric compositional mixtures. However, EDPMs may be useful for other tasks. Specifically, they share an attractive property of exemplar SVMs (ESVMs) (Malisiewicz et al.



**Fig. 17** We visualize detections using our exemplar DPM (EDPM) model. As opposed to existing exemplar-based methods (Malisiewicz et al. 2011), 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

2011): 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. (Malisiewicz et al. 2011) 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. 17.

## 5 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 cate-

gories (Deng et al. 2010), visual properties (Hoiem et al. 2012), dataset collection bias (Torralba and Efros 2011), and component-specific analysis of recognition pipelines (Parikh and Zitnick 2011).

### 5.1 Object Detection

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 (Krizhevsky et al. 2012), local feature analysis (Tuytelaars and Mikolajczyk 2008), kernel methods (Vedaldi et al. 2009), and decision trees (Bosch et al. 2007), 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 regarding parameter sharing and extrapolation may generally hold for other architectures.

### 5.2 Non-parametric Models in Vision

Most relevant to our analysis is work on data-driven models for recognition. Non-parametric scene models have been used for scene completion (Hays and Efros 2007), geolocation (Hays and Efros 2008). Exemplar-based methods have also been used for scene-labeling through label transfer (Liu et al. 2011; Tighe and Lazebnik 2010). Other examples include nearest-neighbor methods for low-resolution image analysis (Torralba et al. 2008) and image classification (Zhang et al. 2006; Boiman et al. 2008). The closest approach to us is (Malisiewicz et al. 2011), who learn exemplar templates for object detection. Our analysis suggests that it is crucial to share information between exemplars and extrapolate to unseen templates by re-composing parts to new configurations.

### 5.3 Scalable Nearest-Neighbors

We demonstrate that compositional part models are one method for efficient nearest-neighbor computations. Prior work has explored approximate methods such as hashing (Shakhnarovich et al. 2003, 2005) and kd-trees (Muja and Lowe 2009; Beis and Lowe 1997). Our analysis shows that one can view parts as tools for exact and efficient indexing into an exponentially-large set of templates. This suggests an alternative perspective of parts as computational entities rather than semantic ones.

## 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, and introduced several non-parametric models to diagnose the state of affairs.

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 (Dalal and Triggs 2005)).

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. We found that larger gains were possible by enforcing richer constraints within the model, often through non-parametric compositional representations that could make better use of additional 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, and non-parametric encodings. 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 non-parametric compositional approaches that implicitly make use of augmented training sets appear the most promising directions.

**Acknowledgments** Funding for this research was provided by NSF IIS-0954083, NSF DBI-1053036, ONR-MURI N00014-10-1-0933, a Google Research award to CF, and a Microsoft Research gift to DR.

## References

Beis, J.S., & Lowe, D.G. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on IEEE* (pp. 1000–1006).

Boiman, O., Shechtman, E., & Irani, M. (2008). In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on IEEE* (pp. 1–8).

Bosch, A., Zisserman, A., & Muoz, X. (2007). Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on IEEE* (pp. 1–8).

Bourdev, L., & Malik, J. (2009). Poselets: Body part detectors trained using 3D human pose annotations. In *International Conference on Computer Vision*.

Chang, C., & Lin, C. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(27), 1–27:27, software <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR 2005*.

Deng, J., Berg, A., Li, K., & Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us?. In *International Conference on Computer Vision*.

Divvala, S.K., Efros, A.A., & Hebert, M. (2012). How important are deformable parts in the deformable parts model? In *European Conference on Computer Vision (ECCV), Parts and Attributes Workshop*.

Everingham, M., Van Gool, L., Williams, C., Winn, J., & Zisserman, A. (2010). The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2), 303–338.

Felzenszwalb, P., & Huttenlocher, D. (2012). Distance transforms of sampled functions. *Theory of Computing*, 8(1), 415–428.

Felzenszwalb, P., Girshick, R., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE TPAMI*, 32(9), 1627–1645.

Gross, R., Matthews, I., Cohn, J., Kanade, T., & Baker, S. (2010). Multiple. *Image and Vision Computing*, 28(5), 807–813.

Halevy, A., Norvig, P., & Pereira, F. (2009). The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2), 8–12.

Hays, J., & Efros, A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26, 4.

Hays, J., & Efros, A.A. (2008). Im2gps: Estimating geographic information from a single image. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on IEEE* (pp. 1–8).

Hoiem, D., Chodpathumwan, Y., & Dai, Q. (2012). Diagnosing error in object detectors. In *Computer Vision ECCV 2012* (Vol. 7574, pp. 340–353). Berlin: Springer.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1106–1114.

Liu, C., Yuen, J., & Torralba, A. (2011). Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12), 2368–2382.

Malisiewicz, T., Gupta, A., & Efros, A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *IEEE, International Conference on Computer Vision* (pp. 89–96).

McAllester, D. A. (1999). Some pac-bayesian theorems. *Machine Learning*, 37(3), 355–363.

Muja, M., & Lowe, D.G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISSAPP09)* (pp. 331–340).

Parikh, D., & Zitnick, C. (2011). Finding the weakest link in person detectors. In *Computer Vision and Pattern Recognition IEEE* (pp. 1425–1432).

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers* (pp. 61–74), MIT Press.

Shakhnarovich, G., Darrell, T., & Indyk, P. (2005). *Nearest-neighbor methods in learning and vision: Theory and practice*. Cambridge: MIT press.

Shakhnarovich, G., Viola, P., & Darrell, T. (2003). Fast pose estimation with parameter-sensitive hashing. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on IEEE* (pp. 750–757).

Tighe, J., & Lazebnik, S. (2010). Superparsing: Scalable nonparametric image parsing with superpixels. In *Computer Vision-ECCV 2010* (pp. 352–365). Springer.

Torralba, A., & Efros, A. (2011). Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition IEEE* (pp. 1521–1528).

Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 Million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11), 1958–1970.

Tuytelaars, T., & Mikolajczyk, K. (2008). Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3), 177–280.

Vedaldi, A., Gulshan, V., Varma, M., & Zisserman, A. (2009). Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on IEEE* (pp. 606–613).

Wu, Y., & Liu, Y. (2007). Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102(479), 974–983.

Zhang, H., Berg, A. C., Maire, M., & Malik, J. (2006). Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on IEEE* (Vol. 2, pp. 2126–2136).

Zhu, X., & Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition*.