Improved object categorization and detection using comparative object similarity

Gang Wang, *Member, IEEE,* David Forsyth, *Fellow, IEEE,* Derek Hoiem, *Member, IEEE*

Abstract—Due to the intrinsic long-tailed distribution of objects in the real world, we are unlikely to be able to train an object recognizer/detector with many visual examples for each category. We have to share visual knowledge between object categories to enable learning with few or no training examples. In this paper, we show that local object similarity information — statements that pairs of categories are similar or dissimilar — is a very useful cue to tie different categories to each other for effective knowledge transfer. The key insight: given a set of object categories which are similar and a set of categories which are dissimilar, a good object model should respond more strongly to examples from similar categories than to examples from dissimilar categories. To exploit this category dependent similarity regularization, we develop a regularized kernel machine algorithm to train kernel classifiers for categories with few or no training examples. We also adapt the state-of-the-art object detector [10] to encode object similarity constraints. Our experiments on hundreds of categories from the Labelme dataset show that our regularized kernel classifiers can make significant improvement on object categorization. We also evaluate the improved object detector on the PASCAL VOC 2007 benchmark dataset.

Index Terms—Comparative Object Similarity, Object Categorization, Object Detection, Kernel Machines, SVM, deformable part model, PASCAL VOC, sharing.

1 Introduction

People can often learn names of new objects from few visual examples. In part, we are able to quickly learn about new object categories because we can relate them to known similar objects. For example, few people know what a “serval” is, but when told it is like a leopard, but with longer legs and lighter body, most can identify one in a picture. “A serval is like a leopard” is a similarity statement defining a new category in terms of existing categories.

In this paper, we exploit such category based similarity statements to learn object models with few or even no training examples. This is an interesting problem, because in the real world, most object categories have very few examples. In Figure 1, we show the number of instances of the 200 most frequent object categories in a part of the Labelme dataset [30], ranked in decreasing order. We can see the distribution is heavily long-tailed. For most categories, we cannot find enough data to train reliable classifiers. We have to transfer information from object categories with many instances to categories with few instances. To transfer information, we must first know what categories should be used for transferring. Similarity is a very good cue for this purpose, as used by human beings.

To computationally exploit object similarity, when learning a model of a new category (e.g., “serval”), we first obtain a short list of similar categories from a human labeler; this would contain “leopard”. We could just use

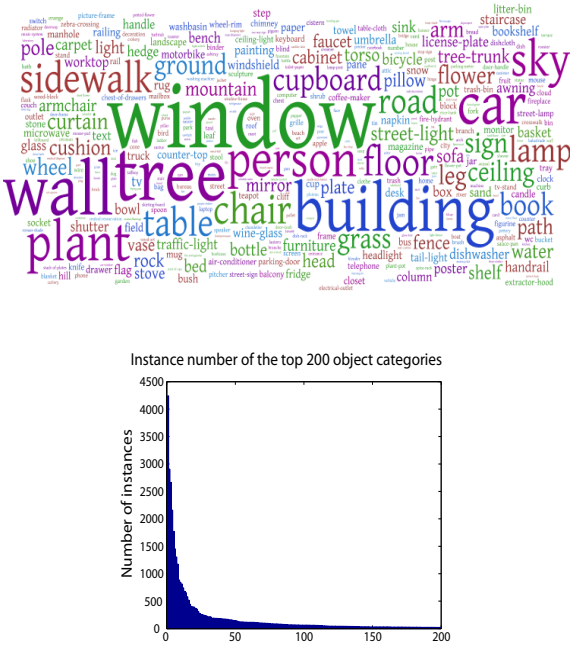


Fig. 1. Most categories in a dataset we use for categorization (which is a part of Labelme [30]) have few or no examples. The top image shows “object clouds”. Objects with bigger names have more instances. Most objects have small names because they have few examples. The bottom image shows number of instances for the top 200 objects. The top 5 categories are: window, tree, wall, building and car. The number of instances decays very quickly.

- *G. Wang is with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore and Advanced digital science center, Singapore.*
  E-mail: wanggang@ntu.edu.sg
- *D. Hoiem and D. Forsyth are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 61801.*

these "leopard" images as positive examples to train the model. But this strategy is not attractive, because the two categories are not the same. Worse, for many categories there may be nothing that is strongly similar: our labeler marks "lamp" and "flower" as similar to "ceiling fan". These categories are similar enough to be helpful, but so different that we cannot mix them together. Due to the uncertain degree of closeness between a target object class and its similar categories, we think that labeled similar categories are just more similar to the target object than to other categories. For simplicity, we call these less similar categories as "dissimilar categories" from now on. Our method uses similarity constraints as a form of regularization during learning. We require that the learned object model should respond more strongly to examples of similar categories than to examples of dissimilar categories. For example, to learn a model of "serval", we obtain a few similar categories (e.g. "leopard") and some dissimilar categories (e.g. "grass" and "bird"). Then we require the model to respond more strongly to "leopard" examples than to "grass" and "bird" examples. This process acts as a category dependent similarity regularizer.

We first apply this mechanism in a kernel machine framework [18]. We force kernel machine classifiers to give higher responses to examples from similar categories than to examples from dissimilar categories. This procedure usually involves tens of thousands of "similar-dissimilar" examples. With a histogram intersection kernel, we can efficiently learn kernel machines using a variant of our previously developed algorithm SIKMA [40]. We evaluate our method on hundreds of object categories, many of which even have no training examples at all. We show that doing so leads to significant performance improvements on hard categorization tasks.

Object *detection* is another important research topic [36], [5], [10], which aims to localize object instances using bounding boxes in clutter images. Training an object detection system requires a large number of positive instances, due to the high complexity object models. To enable detection with few training examples, we adapt the state of the art object detection system [10] to encode object similarity constraints. Similarly, we force an object detector to respond more strongly to an instance from a similar category than to an instance from a dissimilar category. This procedure helps when training examples are rare. We evaluate our approach on a benchmark dataset (PASCAL VOC 2007 dataset [7]), which is widely recognized as a challenging dataset for object detection. Our approach consistently outperforms [10] when there are 20 training examples. When there are 50 positive instances, we still see improvement on some categories.

This paper is a longer version of a published conference paper [38], which only applies comparative object similarity to train kernel classifiers for the object categorization task. In this paper, we also adapt a state-of-the-art object detector to detect object instances with few training examples using the same mechanism (Section 3.2).

2 Related Work

Compared to numerous papers on recognizing / detecting objects using many training examples [39], [22], [10], [6], [11], [40], categorization / detection with few or no training examples receives much less attention. A pioneering work by Miller *et al.* [25] learns a new object category with one training example by effectively bringing test data into correspondence with the model defined by other categories. Fei-Fei *et al.* [9] exploit the tendency of object models to be similar to one another with a strong prior. Bart and Ullman [3] learn a novel class from a single training example by using the experience with already learned classes. Our work differs from these papers by using object similarity to link different object categories together to allow effective knowledge transfer.

Recently, several papers [8], [20] propose to use attributes as intermediate representation to share information across object categories. Attributes refer to properties of objects such as "furry", "has legs", etc. We can learn attribute classifiers from common objects and apply them to recognize novel objects which don't have many training examples. Attributes can also be used to describe novel objects even they are not successfully recognized. Kumar *et al.* [19] use learned attribute classifiers for face verification. The attributes here more specifically refer to visual properties associated with faces such as "nose size", "nose shape", "hair style", etc. Wang *et al.* [37] jointly learns attribute and object detectors, and enable recognizing "attribute object" pairs without any training examples. Palatucci *et al.* [27] apply an interesting idea in neural activity decoding, where the goal is to determine the word or object a person is thinking about by observing an image of that persons neural activity. In this application scenario, it is intractable to collect neural training images for every possible word in English, so they define the notion of a semantic output code classifier (SOC) which utilizes a knowledge base of semantic properties of all the categories. Attributes are parallel to similarity and are both intermediate representations that can tie different categories up. The two can be further integrated to better represent novel objects with existing object categories. More discussions can be found in the "conclusions and discussions" section.

We use visual object similarity as extra supervision to link up different object categories to share features. In [34], [26], [15], there is no human supervision on which categories should be associated. Salakhutdinov *et al.* [31] leverages Wordnet to define a fixed tree hierarchy to share information at different levels and also tries to learn the sharing structure automatically.

Similarity information is popularly used to learn distance metrics. Here one measures similarity with some distance in a feature space, and adjusts feature weights to make objects more similar to those in the same category

and dissimilar to those in different categories [12], [42], [41]; analogous procedures can be applied to measures of similarity that are not metric [16]. These methods cannot use explicit inter-category information; they just ask examples from the same category to be similar to each other. In the absence of category labels, data-dependent measures of smoothness can be used to weight features [11]. In each case, the result is a *global similarity procedure* — the metric is adjusted to be consistent with all available similarity information.

An alternative global similarity procedure uses multidimensional scaling (MDS) to obtain an embedding that is consistent with all similarity data. There is compelling evidence that this is a poor model of human similarity judgements [35] (e.g., human judgements are not symmetric). Similarity judgements may not be consistent with one another or with new information (e.g. "a car is like a van"; "a van is like a bus"; "a bus is like a train" do not mean that a "car is like a train"). MDS resolves this by seeking the best global embedding that is consistent with all statements. The method is also impractical for many categories, because we do not expect to have much detailed pairwise similarity information. Different from these methods, our approach can exploit local similarity statements.

Our work is also relevant to "learning to rank" [17], [43], [23], [14], [13], which is mainly developed by the information retrieval community. The goal is to automatically construct a ranking model from training data, which usually consists of lists of items with some partial order specified between items in each list. Rather than rank individual documents, in this paper, we rank different object categories according to object similarity for knowledge transfer. In some sense, our learnt object models can be considered as ranking models.

3 Learning Object Models with Comparative Similarity

We aim to learn models to recognize / detect named objects. We have few or no positive training examples for target objects, but many negative examples. Some negative categories are identified by human labelers as "similar" or "dissimilar". Examples of similarity annotation are shown in Table 1. Kernel machines and discriminatively trained part based object models [10] are popularly employed to recognize and detect object instances in images respectively. We adapt these two methods to encode comparative object similarity constraints to learn from few examples.

3.1 Incorporating comparative object similarity to train kernel machines for categorization

In this section, we learn an object model F for each name for categorization. A kernel machine classifier is used following [44], [22]. Write the kernel machine classifier F as $\sum_{i=1}^{N} \alpha_i K(x_i, \bullet)$, where $K(x_i, \bullet)$ is a kernel basis function (we use a histogram intersection kernel as in [22], but some other kernels such as the RBF kernel can also be applicable), x_i is the ith support vector and α_i is the weight. In the training procedure, we learn F from T training examples in feature representation along with ground truth class labels $\{(x_t, y_t), t = 1, ..., T, y_t \in \{+1, -1\}\}$.

The usual procedure without incorporating object similarity information is to learn F by minimizing:

$$\frac{1}{T}\sum_{t=1}^{T} L(F(x_t), y_t) + \frac{\lambda}{2}\|F\|^2 \quad (1)$$

where L is the hinge loss $L(F(x_t), y_t) = \max(0, 1 - y_t F(x_t))$; $\frac{\lambda}{2}\|F\|^2$ is a regularization term. Following [18], all possible hypotheses of F are assumed to form a Reproducing Kernel Hilbert space, where $\|F\|^2 =< F, F >$. Accurate learning of F in this way requires numerous positive examples. When only few positive training examples are present, learned classifiers may not be robust.

Then how can we achieve a robust object model with limited positive training examples, but given similar and dissimilar examples? We argue that a good object model would respond strongly to whatever positive examples there happen to be, but would also respond more strongly to similar examples than to dissimilar examples. This lends the problem an ordinal character — our method should rank similar examples more highly than dissimilar examples, rather like an ordinal SVM [13]. Ordinal SVM attempts to learn a function $h(\mathbf{x})$ such that $h(\mathbf{x}_i) > h(\mathbf{x}_j)$ for any pair of examples where $\text{rank}(\mathbf{x}_i) > \text{rank}(\mathbf{x}_j)$. However, we do not have a full ranking of all examples, instead, we only want to rank categories. So we cannot use a conventional ordinal SVM.

Our model F should: be positive for positive instances; be negative for negative instances; and be larger for similar instances than for dissimilar instances. The first two requirements are straightforward to express with the hinge loss. For the third, if g_n^s is an instance from a similar category and g_n^d is an instance from a dissimilar category, $F(g_n^s)$ should always be larger than $F(g_n^d)$, with some margin. We impose this constraint by preparing a set of N similar-dissimilar pairs, and scoring $L(F(g_n^s) - F(g_n^d), 1)$, where L is the hinge loss, but some other loss functions can also be applicable. This acts as a regularization term.

There could be very many pairs. If there are many positive examples, then the similarity constraint is less significant, but if there are few, it is an important constraint. This means the weight placed on this similarity term should depend on the number of positive examples T_p. We choose F to minimize

$$\frac{1}{T}\sum_{t=1}^{T} L(F(x_t), y_t) + \alpha(T_p)\frac{1}{N}\sum_{n=1}^{N} L(F(g_n^s) - F(g_n^d), 1) + \frac{\lambda}{2}\|F\|^2 \quad (2)$$

object	similar categories	dissimilar categories	similarity type
shrub	flower, grass, hedge	bowl, vase, bottle, umbrella	synonymous
number	text	tv, sink, box, bush	synonymous
body	torso	grass, road, motorbike, hedge	nearly synonymous
picture frame	painting	towl, bed, floor, sofa, bush	nearly synonymous
plant pot	bowl	rug, sign, sand, door	different
bird house	box, book	flower, sofa, floor, motorbike	different
gloves	curtain, flag	grass, desk, door, bottle	very different
fireplace	fence, railing	step, path, bicycle, snow	very different

TABLE 1

Examples of similarity annotation. For each target category, we identify its "similar" and "dissimilar" categories. We also label the similarity type by a second volunteer.

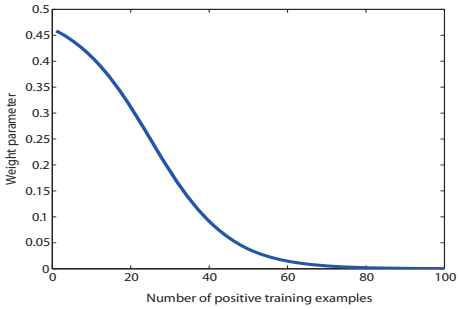


Fig. 2. Weights of similarity term for different number of positive training examples. When there are more positive examples, similarity term contributes less; otherwise, it contributes more.

The first term is the empirical loss of the target category, the second term imposes similarity constraints (it is also considered to be a regularization term), and the third is the conventional regularizer. $\alpha(T_p)$ denotes the weight on similarity as a function of the number of positive training examples. Generally, if there are few positive examples, $\alpha(T_p)$ should be large, and if there are many, it should be small. Hence we set $\alpha(T_p)$ as the output of a function with the input T_p:

$$\alpha(T_p) = \frac{A}{1 + e^{B(T_p - C)}} \tag{3}$$

Discussion on weight parameter setting for the similarity term. Commonly, weights for multiple cues are set using cross-validation. But we have few positive examples and many categories, making cross-validation ineffective and computationally expensive. Instead, we assume that the parameters should depend primarily on the number of training examples and set parameters that work well for a variety of categories. In our implementation, A, B, and C are chosen through validation on five categories which have enough training examples. Then they are fixed and applied to all the other categories with few training examples. The final parameters we use are: $A = 0.5$, $B = 0.1$, and $C = 25$. Figure 2 shows how $\alpha(T_p)$ changes with different number of training examples T_p.

3.1.1 Training

Learning F from Eq 2 involves a large number of similar-dissimilar example pairs. Training can be very slow using traditional batch based algorithms [29]. Stochastic gradient descent is an efficient method to learn from large-scale datasets [32], [18], [40], [21]. Different from the traditional gradient descent algorithm, at each iteration, this method only uses a subset of data points to approximately calculate the gradient and take a step. Extensive empirical results such as those in [32], [18], [40], [21] show that it can achieve very close performance and meanwhile be much faster, compared to batch based algorithms.

In our implementation, at each iteration, we only choose one single data point (here, a similar-dissimilar example pair is considered as a data point) to calculate the gradient. The algorithm is:

At the ith iteration, select a single data point at random:

1) If a labelled example x_t is selected, then follow the procedure of [40], the new objective function is:

$$\frac{1}{T}L(F(x_t), y_t) + \frac{\lambda}{2}\|F\|^2 \tag{4}$$

The sub-gradient is:

$$\lambda F_{i-1} - \frac{1}{T}\sigma_i K(x_t, \bullet) \tag{5}$$

Where $\sigma_i = \begin{cases} 1 \text{ if } y_t F_{i-1}(x_t) < 1 \\ 0 \text{ otherwise} \end{cases}$

Then F is updated as:

$$F_i = (1 - \lambda\eta_i)F_{i-1} + \frac{1}{T}\eta_i\sigma_i y_t K(x_t, \bullet) \tag{6}$$

2) If a similar-dissimilar pair $\{g_n^s, g_n^d\}$ is selected, the new objective function $\widehat{O}$ for this pair is

$$\frac{1}{N}\alpha(T_p)L(F(g_n^s) - F(g_n^d), 1) + \frac{\lambda}{2}\|F\|^2 \tag{7}$$

the sub-gradient $\frac{\partial \widehat{O}}{\partial F}|_{F=F_{i-1}}$ is:

$$\frac{1}{N}\alpha(T_p)\sigma_i(K(g_n^d, \bullet) - K(g_n^s, \bullet)) + \lambda F_{i-1} \tag{8}$$

where $\sigma_i = \begin{cases} 1 \text{ if } F_{i-1}(g_n^s) - F_{i-1}(g_n^d) < 1 \\ 0 \text{ otherwise} \end{cases}$
and the update becomes

$$F_i = (1 - \lambda\eta_i)F_{i-1} + \frac{1}{N}\alpha\eta_i\sigma_i(K(g_n^s, \bullet) - K(g_n^d, \bullet)) \quad (9)$$

We use a histogram intersection kernel for K, allowing us to use the fast training algorithm of [40] without sacrificing accuracy.

3.2 Incorporating comparative object similarity to train part based object models for detection

The above section introduces a method on adapting kernel machines for object categorization. Object detection, which aims to predict bounding boxes of object instance in cluttered images, also usually requires a lot of training data. This section presents an approach to incorporate object similarity information to train part based object detectors.

We adapt the state of the art detection system [10] to encode object similarity constraints. The approach is sketched here for reader's convenience.

In Felzenszwalb *et al.*'s procedure [10], an object category is modeled as a star-structured part-based model with model parameters β. A test instance x_t is scored by:

$$f_\beta(x_t) = \max_{z \in Z(x_t)} \beta \cdot \Phi(x_t, z) \quad (10)$$

where Z denotes all the possible object configurations, z specifies a configuration, $\Phi(x_t, z)$ denotes the concatenation of HOG features [5] extracted from image subwindows defined by z and part deformation features.

This latent structure model is trained by minimizing the following objective function:

$$\frac{1}{T}\sum_{i=1}^{T} L(f_\beta(x_t), y_i) + \lambda\frac{1}{2}\|\beta\|^2 \quad (11)$$

where y_i is the class label for the ith training example, and L is the hinge loss. This formulation is different from a standard support vector machine because it has a latent structure. A latent SVM leads to a non-convex optimization problem that becomes convex once the latent structure is specified for positive examples. In [10], a "coordinate descent" approach is adopted to learn model parameters. At each round, it first chooses the best latent structure in each positive image that responds most strongly to the current model, and then solve a convex optimization problem using the specified latent structures.

Learning object models using Eq 11 usually requires many training instances because of the sophisticated model structure and also because of a large number of model parameters. Many object categories only have few training examples available. To encode similarity constraints, likewise, we require an object model to respond more strongly to examples from similar categories than to examples from dissimilar categories.

Suppose we also have examples from similar and dissimilar categories except the positive and negative training examples. Write $\{g_n^s, g_n^d\}$ for a pair of examples, where g_n^s is from a similar category while g_n^d is from a dissimilar category. Then $f_\beta(g_n^s)$ should be larger than $f_\beta(g_n^d)$, with a margin. We obtain the following optimization function:

$$\frac{1}{T}\sum_{i=1}^{T} L(f_\beta(x_t), y_t) + \alpha\frac{1}{N}\sum_{n=1}^{N} L(f_\beta(g_n^s) - f_\beta(g_n^d), 1) + \lambda\frac{1}{2}\|\beta\|^2 \quad (12)$$

where the second term penalizes the failure of the model to respond more strongly to similar examples than to dissimilar examples; again, we use the hinge loss for this term. We set α as the same as in Eq 3. When the number of positive examples is large, α is small; when the number of positive examples is small, α is large.

3.2.1 Training

An attractive feature of this method is that incorporating it into the existing code of the Felzenswzalb *et al.* detector is straightforward. We modify their "coordinate descent" procedure to optimize Eq 12, obtaining:

1) For each positive example, find the latent structure which scores highest: $z_t = \text{argmax}_{z \in Z(x_t)} \Phi(x_t, z)$. We also find the best latent structure for each g_n^s and g_n^d in the similarity constraints using the same idea.
2) Fix the latent structure z, and optimize β by solving a convex optimization problem. As in [10], we adopt a stochastic gradient descent method, so we can easily modify their code to train our detector. The procedure described here is also very similar to the one described in Section 3.1.1. After step 1, we have:
$f_\beta(x_t) = \beta\Phi(x_t, z_t(\beta)))$
$f_\beta(g_n^s) = \beta\Phi(g_n^s, z_n^s(\beta)))$
$f_\beta(g_n^d) = \beta\Phi(g_n^d, z_n^d(\beta)))$

   In the gradient descent procedure, the gradient of Eq 12 is calculated as:

$$\frac{1}{T}\sum_{i=1}^{T} p(\beta, x_t, y_t) + \alpha\frac{1}{N}\sum_{n=1}^{N} q(\beta, g_n^s, g_n^d) + \lambda\beta \quad (13)$$

$$p(\beta, x_t, y_t) = \begin{cases} 0 \text{ if } y_i f_\beta(x_t) \geq 1 \\ -\Phi(x_t, z_t(\beta))) \text{ otherwise} \end{cases}$$

$$q(\beta, g_n^s, g_n^d) = \begin{cases} 0 \text{ if } f_\beta(g_n^s) - f_\beta(g_n^d) \geq 1 \\ -\Phi(g_n^s, z_n^s(\beta))) + \Phi(g_n^d, z_n^d(\beta))) \text{ o/w} \end{cases}$$

   In the stochastic gradient descent, the gradient Eq 13 is calculated using a subset of examples (can be positive / negative, or similar / dissimilar pairs) and a step is taken in the negative direction at each iteration. We follow [10] to set parameters.

All other important implementation components of [10] such as feature extraction, data-mining hard examples,

and post-processing can be used without any changes. For each category, we train a mixture model with 2 mixture components. We flip each positive training examples as in [10], but similar-dissimilar pairs are not flipped.

4 EXPERIMENTS

We presented two methods to exploit object similarity for object categorization and detection respectively. In this section, we evaluate their performance on benchmark datasets and compare them with two baseline approaches. Note that in each experiment, we evaluate on binary categorization rather than multi-class classification.

4.1 Experiments for object categorization

4.1.1 Experimental settings

Dataset: We choose 2,831 images from the Labelme [30] dataset as a test bed for our categorization experiments. They are realistic street and indoor images and their regions are fully annotated. We are doing categorization rather than detection, so we use the ground truth polygon annotation to crop objects out and perform experiments on these clean object regions (for both training and test). Experimental results on object detection are presented in Section 4.2. We manually reword object names by correcting misspelled words, removing non-noun words (e.g., "a"), and passing to the most common nouns (e.g., replacing "pedestrian walking" with "person"). This leaves 972 object categories in total.

As Figure 1 shows, the distribution of object categories in the dataset is heavily long-tailed (which is also suspected to be true in the real world). Around 600 categories have less than 6 instances. Only 70 categories have more than 100 instances. We randomly select 1,500 images as training data and the other 1,331 images as test data.

We select 90 object categories, which have more than 60 instances, as prototype categories. Categories with few examples are not considered as prototype categories. We finally have 225 test categories, each of which has at least one test instance. For each category, a human volunteer identified up to five similar objects from prototype categories without seeing any images from the dataset. In this labeling procedure, the annotator is asked to label visually similar and dissimilar objects. No extra instructions are given on which aspect (e.g., shape or texture) should be used to judge similarity.

Similarity annotation was checked by a second volunteer, who broke the similarity judgements into four cases: synonymous (e.g. category "beach rock" and prototype "rock"); near synonymous (e.g. "worktop"; "bar counter"); different (e.g. "bird"; "flag" — the labeler felt both flap in the sky); and very different (e.g. "ceiling fan";"flower"). More examples on similarity annotation are shown in Table 1.

When training one object model, all the other classes are used as negative. In the test procedure, we classify each test image region and output a classification score. AUC values are calculated for each class. In this experiment, we directly use the ground truth segmentations of test images to extract object regions. There are 21,803 test regions in total.

4.1.2 Features and parameters

We densely extract patches from each object region, and represent each patch as a SIFT descriptor [24]. We sample a subset of descriptors to cluster, and form a visual vocabulary with 800 visual words. We use histograms of these visual words as feature representation. In the training procedure, we use around 20,000 negative examples and 20,000 pairs of similar-dissimilar examples to learn each object model. The learning rate η_i is set to be $\frac{1}{i+100}$, and λ is set to be 0.00005. It usually takes $50 \sim 120$ seconds to train one object model, which is very fast.

4.1.3 Baselines

We compare our algorithm with two baselines. Baseline 1 uses all available positive and negative examples in the usual way. If there are no positive examples, it outputs a random guess. We call this baseline "No similarity information", abbreviated as "No Similarity". Baseline 2 directly uses instances from similar categories as positive examples (weighted with a weight which is defined in the similar form of Eq 3) to train object models. We call this baseline "Similar categories as positive categories", abbreviated as "Similar=Pos". For this baseline, the classifier is learned via the following objective function:

$$\frac{1}{T}\sum_{t=1}^{T} L(F(x_t), y_t) + \alpha(T_p)\frac{1}{N}\sum_{n=1}^{N} L(F(g_n^s), 1) + \frac{\lambda}{2}\|F\|^2 \quad (14)$$

Where g_n^s denotes the feature representation of the nth similar image. This baseline uses the same similarity information, but its performance is not as good as our comparative similarity approach, shown by the following experimental studies.

4.1.4 Similarity improves AUC

If we present a method with one positive and one negative example, area under the receiver operating curve (AUC) gives the probability that the positive example will score higher than the negative. AUC is a good measure of performance for a task like naming with few examples. Instead of using the standard AUC, we adopt a balanced AUC, where each test example is weighted by $\frac{1}{N}$ (N is the total number of test instances from the same category). This can better measure how well the learned models are against all the other categories rather than against some very common categories. Our similarity method produces strong improvements in AUC for all test categories, especially when there are no positive examples (Figure 3). AP is a less helpful measure, because there are very few positive examples and approximately

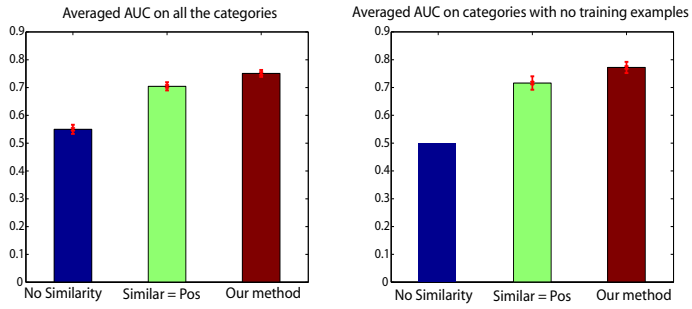


Fig. 3. Similarity information improves AUC, even when there are no examples. On the **left**, AUC averaged over all the 225 test categories for two baselines and for our method; on the **right**, comparison on the 110 test categories which have *no* positive examples, both with standard error bars.

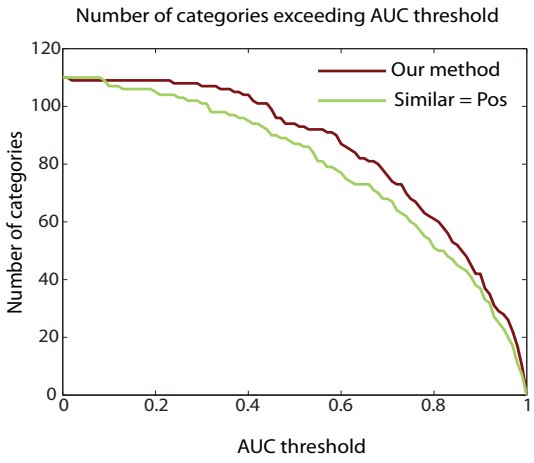


Fig. 5. Number of categories for which AUC is greater than a threshold. These categories have no positive training examples. The x-axis denotes the AUC thresholds. The y-axis denotes the number of categories whose AUC values exceed the thresholds. There are more than 40 categories with AUC values higher than 0.9 using our method. Our method consistently gets more categories than baseline 2 (Similar = Pos) in the high AUC area. Note that some categories have AUC values smaller than 0.5, because the AUC is unstable when there are only few positive test examples.

20000 negative examples and therefore all scores are very small and unstable.

We also show some qualitative results in Figure 4. Our method gets better AUC values and ranks more sensible regions on the top.

Our method can reach very high AUC scores on many categories even if they have no training examples. Figure 5 shows the number of categories (from the 110 categories with no training examples) whose AUC values exceed a set of AUC thresholds. More than 40 categories have bigger AUC values than 0.9.

This effect is not purely due to synonymy in the labels. We sort categories by the strongest similar prototype (strongest: synonymous to weakest: very different). Overall, there are 53 categories with at least one synonymous similar prototype; 70 categories with at least one

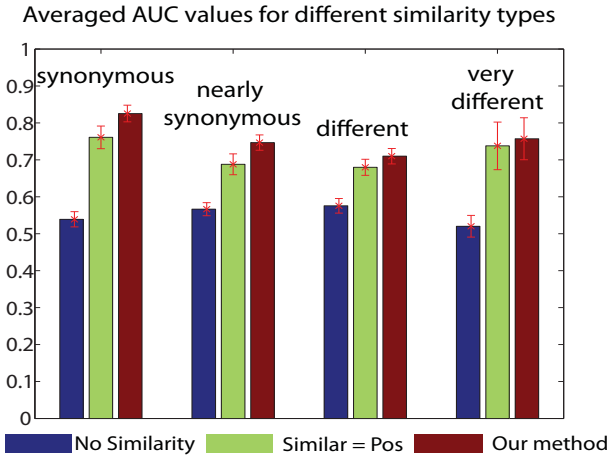


Fig. 6. Similarity improves average AUC score; the effect is not due to synonymy. We show the average for all categories with at least one synonymous similar prototype; all with at least one near synonymous similar prototype and no stronger; all with at least one different similar prototype and no stronger; and all which have only very different similar prototype. Note there are across the board improvements, which are strong compared to error bars.

Number of examples	Number of classes	Average improvement
0	144	0.26
1-5	47	0.155
6-10	31	0.072
11-20	47	0.061
21-30	11	0.024
31-40	4	-0.027

TABLE 2
Average AUC improvement of our approach over the Baseline 1 as a function of the number of the training examples.

near synonymous similar prototype and no stronger; 90 categories with at least one different similar prototype and no stronger; and 12 categories which have only very different similar prototype. We show average AUC scores for each type in Figure 6. We can see that even if the similar prototypes are at best "different" or "very different", using similarity yields a better AUC.

We also show the Average AUC improvement of our approach over the Baseline 1 as a function of the number of the training examples in Table 2. We can observe that comparative similarity can help most when there are fewer positive training examples. When the number of positive examples increases, using the comparative similarity gains little.

The number of similar classes for our examples ranges from one to three, with a very few categories having more. We investigated the effect of the number of similar classes on the improvement in AUC, but found no effect. We believe that it is the quality of labeled similar categories that matters rather than the number of similar categories.

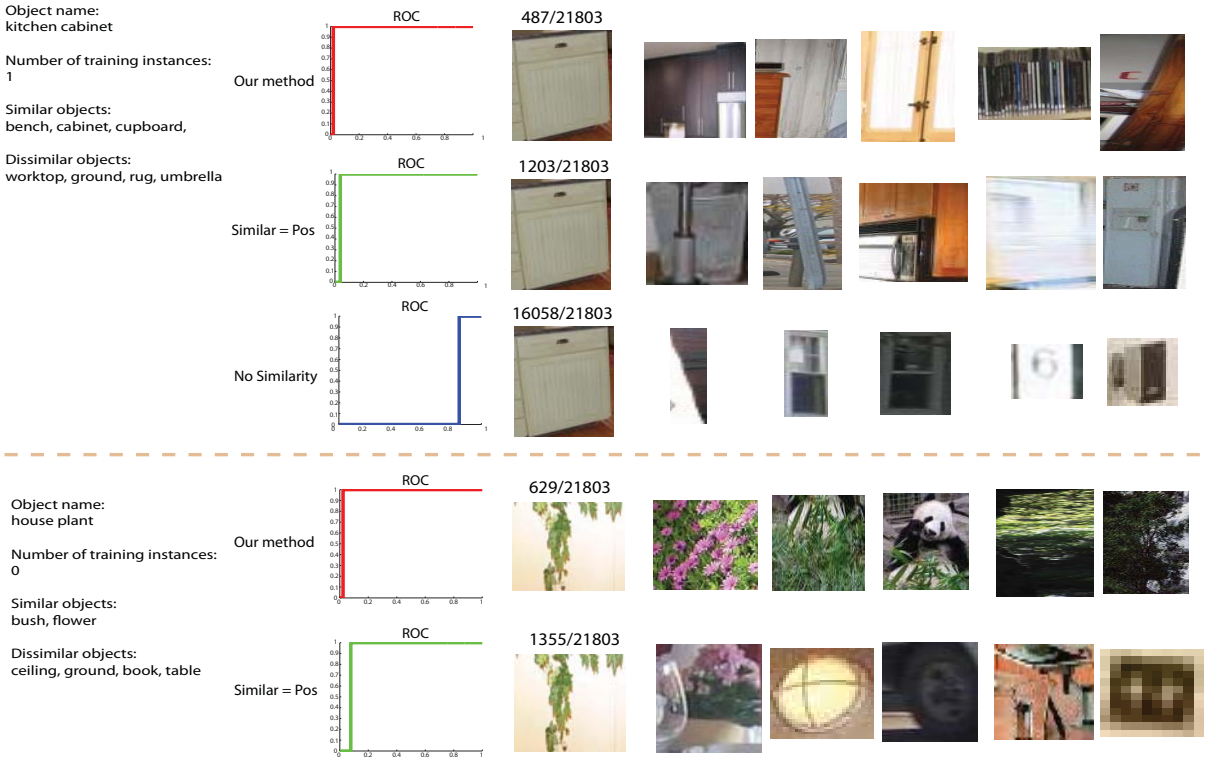


Fig. 4. Classification results for two categories. For each category, the first row shows results using our method; the second row shows results using baseline 2 (Similar = Pos); the third row shows results using baseline 1 (No Similarity), if there are positive training examples. At each row, the first figure shows the ROC curve; the second image shows the top ranked positive test instance (for each of these two object classes, there is only one positive test instance), the number above the image shows the rank (out of 21,803 test instances); the following five images show top ranked test regions. Our method gets better AUC (rank) than baseline 2 and baseline 1. It also ranks more reasonable regions on the top.

4.1.5 *Similarity improves correspondence*

An improvement in AUC is very helpful in finding correspondence between regions and weakly labeled object names [1].

We choose 197 images from the test set which have at least three regions from any of our 225 test categories. Their labels are weakly labeled, meaning that we don't know which label goes to which region. Our task is to use the learned models to establish the correspondence.

We solve for correspondence with a maximum weight bipartite matching (using the Hungarian algorithm [28]), where weights are given by calibrated classification s-cores. The matching results are region labels. We calculate matching accuracy for each class. The values are averaged for comparison to avoid effects of large categories (see Table 3).

In Figure 8, we show the average accuracy values on categories by the number of training instances. Our method gets a large improvement on categories with zero or few training instances.

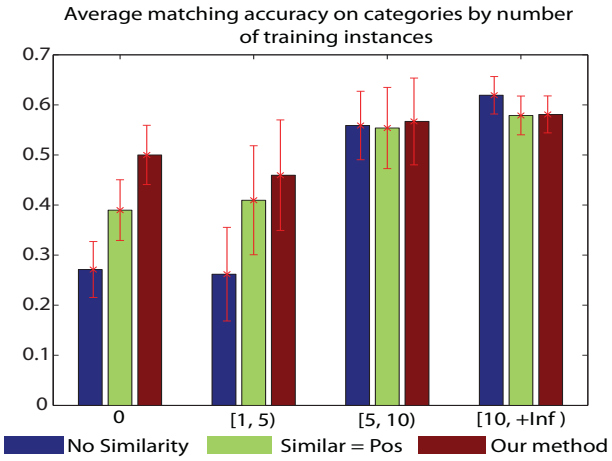


Fig. 8. Average **matching accuracy** on categories by number of training instances. On the categories with zero or few training examples, using similarity helps a lot. Our method also gets better results than baseline 2(Similar = Pos).

4.2 Experiments for detection

4.2.1 *Experimental settings*

We evaluate our proposed detection method (Section 3.2) on an object detection task, which aims to predict bounding boxes of object instances in images (if there are any). We adopt the PASCAL VOC 2007 comp3 challenge protocol [7]: at test, we predict bounding boxes and their confidence values, and draw a precision-recall curve for each category. An AP (average precision) value is

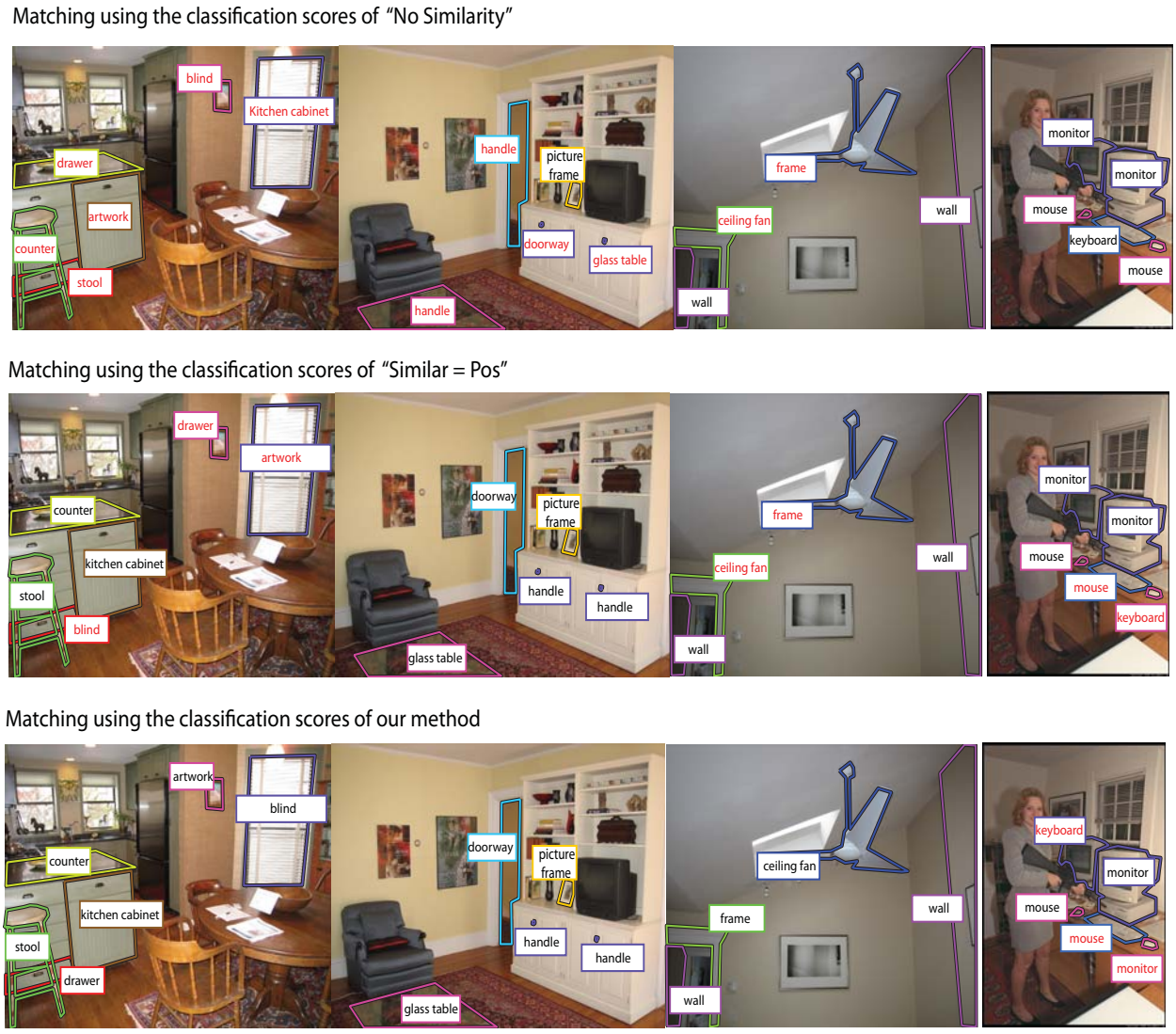


Fig. 7. Examples showing found correspondence between regions and weak class labels. On each image, regions are depicted by polygons in different colors, the found corresponding class names are surrounded by squares in the same colors. Incorrect correspondence is indicated with red object names. Each column shows comparison on the same image. For the first three images, using classification scores by our method find better correspondence. This is mainly because many categories such as "artwork" and "ceiling fan" have few or no training examples, the baseline classifiers cannot learn good models for them. Our method doesn't work well on the fourth image, because "mouse" and "keyboard" have strong similarity correlation (their similar categories are both labeled as "book" and "box"). One mouse (keyboard) model trained with similarity may be more likely to confuse keyboard (mouse).

No Similarity	Similar = Pos	Our method
0.440	0.486	0.535

TABLE 3

Average **matching accuracy** using classification scores by different methods. The accuracy is averaged over categories. Using our method can establish better correspondence.

calculated accordingly as the performance measurement. Predicted bounding boxes are considered correct only if they overlap more than 50% with the ground-truth bounding boxes. For the weight of the similarity term, we set $A = 0.5$, $B = 0.1$, and $C = 25$, the same as the ones used for categorization.

We use the same feature implementation of [10] and parameters such as the support vector machine cost parameter.

Our approach should be best applied to train object detectors of unfamiliar objects for which we cannot find enough training examples. But currently, there are no established benchmark datasets for this task. So we evaluate our system on the PASCAL VOC 2007 dataset [7], which is largely recognized as a challenging dataset for object detection.

This dataset has twenty categories, each object category usually has several hundreds of training instances and several hundred of test instances. To test how our

Fig. 9. Five top ranked regions using different approaches. 20 positive training examples are used for these experiments. For each category, regions in magenta are the ones ranked by the method without using any similarity information; regions in green are by our approach. Approaches using similarity rank more true positive regions on the top.

object	similar categories	dissimilar categories
bicycle	motorbike	person car horse cat chair
bus	car train	person chair horse cat pottedplant
car	bus train	person chair horse cat pottedplant
cat	dog horse sheep	person bottle car chair
chair	diningtable	person sofa dog bus bicycle
cow	horse sheep	person car pottedplant bird
diningtable	chair	person sofa dog bus bicycle
dog	cat horse	person car train chair sofa
horse	cow sheep dog	person car chair bottle bus
motorbike	bicycle	person car horse cat chair
sheep	horse cow	person car pottedplant chair bottle
train	car bus	person chair horse cat pottedplant

TABLE 4

Twelve object categories in PASCAL VOC 2007 dataset and their similar and dissimilar categories. Similarity constraints help learn object detectors when positive training examples are few.

approaches can help when training examples are rare in the training procedure, instead of using all the trainval set, we only use a subset (e.g., 20 positive examples) to train detectors. Note that all the negative images in the train set are used. To get stable test performance numbers, we use all the test data.

Similar to the method of evaluating categorization performance, we compare our detection algorithm with two baselines: no similarity information and similar categories as positive categories. In the training subset selection procedure, we randomly choose a certain number (e.g., 20) of positive training examples, and repeat for five times to calculate the mean value for comparison. When training object models using our approach, there could be too many similarity pairs if we enumerate each of them. Instead, at each time, we randomly sample 500 example pairs for each object pair. This usually results in around 5,000 example pairs in total (depending on how many similar-dissimilar object pairs are used.)

The PASCAL VOC 2007 dataset has twenty categories, but for eight of which we cannot find similar categories in the same dataset (e.g., "person" and "bottle"). One solution is to get similar categories from other bigger datasets such as Labelme [30]. In this paper, we simply ignore these eight categories and only test on the remaining twelve categories, which have enough categories to evaluate our approaches' performance. These twelve object categories and their similarity annotation are listed in Table 4.

4.2.2 Performance Comparison

We first compare our approach with the two baselines on each of the twelve object categories listed in Table 4, using 20 positive training instances. It usually takes around one day to train one object detector, because of the thousands of similar-dissimilar example pairs. In the test procedure, we choose a low threshold to make sure enough bounding boxes (usually 100) are kept for each test image. The results (in AP scores) are compared for each category in Table 5 when 20 positive training examples are used. Using similarity consistently improves AP scores. And our approach works better than directly appending similar categories to positive categories.

We also compare our approach with the baselines using 30 and 50 positive training examples in Table 6. According to the sigmoid function, weights of the similarity term are set to be 0.23 and 0.04 respectively. We also experiment with all the positive training examples. Performance gets better with more positive training examples. And similarity is more helpful when a small number of positive training examples are available.

Discussion on when comparative similarity can be helpful. Both Table 2 and Table 6 show that comparative object similarity helps when the positive examples are few. However, when the number of positive examples increases, comparative object similarity may even decrease the performance. This can be interpreted via the theory of bias-variance [4]: similarity constraints make imperfect assumptions. They create additional constraints, which should reduce the variance of parameter estimation, but they introduce bias. As the number of genuine training examples increases, the benefit to reduced variance is decreased, while the bias remains. That is why lower or no gain is observed with a sufficiently large number of training examples. Comparative similarity can only help when the benefit of reducing variance exceeds the loss of introducing bias. The tipping point depends on the distribution of data points in the feature space and on the particular similarity constraints.

Fig. 10. Top false positive regions of "horse" and "motorbike". For each category, regions in magenta are ranked by the Baseline 1 (No Similarity); regions in green are by our approach.

In Figure 9, we show top ranked regions on the right, produced by Baseline 1 (No Similarity) and our approach. Using similarity ranks more true positive regions on the top.

Figure 10 shows top ranked false positive regions of "horse" and "motorbike". Not surprisingly, detectors trained using similarity constraints find regions from similar categories as false positives. But note that leveraging information from similar and dissimilar categories help a detector to be more discriminative from background regions (one can see baseline detectors find many false positive regions on the background).

In Figure 11, we show true positive test regions whose rank is improved most significantly by using similarity compared to Baseline 1 (without similarity information). When tested with the baseline detector, these regions have low confidence values and are ranked low; when using the detector trained with our approach, they receive much higher confidence scores and are highly ranked in all the predicted bounding boxes. Using sim-

	No Similarity	Similar = Pos	Our Approach
bicycle	0.382±0.033	0.423±0.015	0.419±0.017
bus	0.069±0.013	0.117±0.014	0.168±0.010
car	0.188±0.015	0.201±0.008	0.217±0.012
cat	0.002±0.000	0.009±0.007	0.014±0.002
chair	0.002±0.000	0.024±0.005	0.031±0.006
cow	0.015±0.005	0.041±0.006	0.054±0.004
diningtable	0.031±0.008	0.029±0.007	0.027±0.007
dog	0.010±0.007	0.016±0.008	0.013±0.004
horse	0.112±0.018	0.122±0.009	0.163±0.012
motorbike	0.223±0.020	0.217±0.017	0.247±0.015
sheep	0.094±0.018	0.103±0.021	0.136±0.015
train	0.121±0.023	0.115±0.018	0.120±0.014
MEAN	0.104	0.119	0.134

TABLE 5

AP values of different approaches on twelve PASCAL VOC 2007 object categories. We used 20 positive training examples for each category. Our approach outperform the two baselines on most object categories.

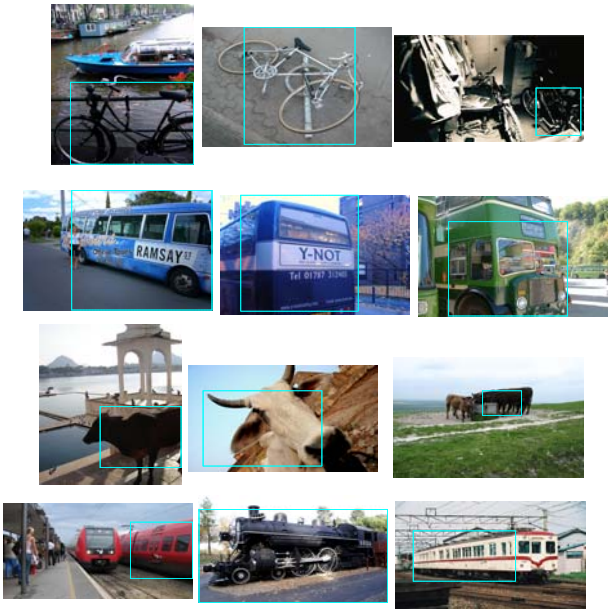


Fig. 11. True positive test regions whose rank is most significantly improved using our approach. They have low confidence values when tested using the baseline 1 (No Similarity) detector, but high confidence values when tested using the detector trained by our approach.

ilarity constraints helps because more relevant information is used. For example, some bus regions look like trains. They are ranked higher using our approach when the train is used as a similar object category to train the bus detector.

5 Conclusions and Discussions

In this paper, we present a method of using comparative object similarity to help learn object models with few or even no positive training examples. We adapt state-of-the-art categorization and detection systems to incorporate object similarity constraints. But note that the model is wholly general and can be applied to a wide variety of problems. For object categorization, experimental results show that our method leads to significant improvements on recognizing hundreds of categories with few or no training examples. For object detection, our proposed approach shows improvement on a benchmark detection dataset PASCAL VOC 2007 when using a small number of positive training examples (typically, 20).

Some future work can be done to better exploit object similarity for visual categorization. Currently, all the similarity constraints are equally considered. However, some constraints should contain more valuable information. For example, “a cat is like a dog” is more useful than “a cat is like a sheep”. We will consider assigning bigger weights to example pairs from similarity constraints with higher quality.

Another possible extension is to integrate similarity and attributes [8], [20], [37] for more powerful knowledge transfer. In some sense, objects are considered similar because they share common visual attributes. For example, a cat and a dog are similar because they both have “legs” and “tail”, and are both “furry”. If we know what attributes make two object categories similar, then we can build a machinery to only transfer visual knowledge corresponding to these specified attributes. This will reduce noise in the knowledge transfer procedure and lead to more effective sharing. An interesting property of attributes is that they can be used to describe a novel object class. For example, “A serval is furry and has legs.” However, there are many different types of legs, such as cats’ legs, leopards’ legs, and even tables’ legs. These legs can be quite different. Only saying “has legs” is not accurate enough to describe, and there is no taxonomy available to define different legs in a fine-grained manner. When similarity information is appended, we can say a serval has legs, which are similar to leopards’ legs. In this way, similarity can heavily enhance the descriptive power of attributes.

When using similarity, the learned object models might confuse the target categories with similar categories, since information from similar categories is shared. As shown in Figure 10, when detecting motorbikes, many bicycle instances are ranked top. One

	No Similarity (mean AP values)	Similar = Pos	Our Approach
20 positive examples	0.104	0.119	0.134
30 positive examples	0.162	0.168	0.175
50 positive examples	0.187	0.183	0.183
All positive examples	0.278	0.267	0.270

TABLE 6

Performance comparison of Baseline 1, Baseline 2, and our approach with 20, 30, 50, and all the positive training examples respectively. The performance is shown in mean AP values of the twelve PASCAL object categories.

solution is to specify shared attributes as introduced above, then we can use other attributes to disambiguate. Another potential solution is to partition categories using taxonomy [2], [33], or using scenes: are the objects found in kitchen or in park? For each object class, we only need to train a model that is against categories from the same partition. Then we can choose similar categories from other partitions to avoid the confusion.

REFERENCES

[1] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D.M. Blei, and M.I. Jordan. Matching words and pictures. *The Journal of Machine Learning Research*, 2003.

[2] E. Bart, I. Porteous, P. Perona, and M. Welling. Unsupervised learning of visual taxonomies. In *CVPR*, 2008.

[3] E. Bart and S. Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. 2005.

[4] L. Breiman. Arcing classifier. *The annals of statistics*, 26(3):801–849, 1998.

[5] N. Dalai, B. Triggs, I. Rhone-Alps, and F. Montbonnot. Histograms of oriented gradients for human detection. *CVPR*, 1, 2005.

[6] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

[7] M. Everingham, L. Van Gool, CKI Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007.

[8] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Proc. CVPR*, 2009.

[9] L. Fei-Fei, R. Fergus, and P. Perona. One-Shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *PAMI*, 2009.

[11] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised Learning in Gigantic Image Collections. In *NIPS*, 2009.

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

[13] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *KDD*, 2002.

[14] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, 2008.

[15] R. Salakhutdinov. J. J. Lim and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *NIPS*, 2011.

[16] D.W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(6):583–600, 2000.

[17] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[18] J. Kivinen, AJ Smola, and RC Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.

[19] Neeraj Kumar, Alexander C. Berg, Peter N. Belhumeur, and Shree K. Nayar. Describable visual attributes for face verification and image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[20] C.H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Proc. CVPR*, 2009.

[21] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *The Journal of Machine Learning Research*, 10:777–801, 2009.

[22] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[23] T.Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[24] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[25] E.G. Miller, N.E. Matsakis, and P.A. Viola. Learning from one example through shared densities on transforms. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 464–471. IEEE, 2000.

[26] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 1, pages 3–10, 2006.

[27] M. Palatucci, D. Pomerleau, G. Hinton, and T.M. Mitchell. Zero-Shot Learning with Semantic Output Codes. In *NIPS*, 2009.

[28] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Pubns, 1998.

[29] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. 1999.

[30] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. LabelMe: a database and web-based tool for image annotation. *IJCV*, 2008.

[31] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to share visual appearance for multiclass object detection. In *IEEE computer society conference on Computer vision and pattern recognition*, 2011.

[32] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM New York, NY, USA, 2007.

[33] J. Sivic, B.C. Russell, A. Zisserman, W.T. Freeman, and A.A. Efros. Unsupervised discovery of visual object class hierarchies. In *CVPR*, 2008.

[34] A. Torralba, K. Murphy, and W.T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. of the 2004 IEEE CVPR.*, 2004.

[35] A. Tversky et al. Features of similarity. *Psychological review*, 1977.

[36] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple. In *Proceedings of CVPR2001*, volume 1.

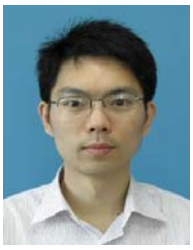
[37] G. Wang and D. Forsyth. Joint learning of visual attributes, object classes and visual saliency. In *ICCV*, 2009.

[38] G. Wang, D. Forsyth, and D. Hoiem. Comparative object similar for improved recognition with few or zero examples. In *CVPR*, 2010.

[39] G. Wang, D. Hoiem, and D. Forsyth. Building text features for object image classification. In *CVPR*, 2009.

[40] G. Wang, D. Hoiem, and D. Forsyth. Learning Image Similarity from Flickr Groups Using Stochastic Intersection Kernel Machines. *ICCV*, 2009.

[41] K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.
[42] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *NIPS*, pages 521–528, 2003.
[43] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.
[44] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 73(2):213–238, 2007.



Gang Wang is an Assistant Professor in Electrical and Electronic Engineering at the Nanyang Technological University. He is also a Research Scientist of the Advanced Digital Science Center. He received the B.S. degree from Harbin Institute of Technology, China, in 2005 and the Ph.D. degree from the University of Illinois at Urbana-Champaign, Urbana. His research interests include computer vision and machine learning.

David Forsyth (Fellow, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from the University of the Witwatersrand, Johannesburg, South Africa, and the D.Phil. degree from Balliol College, Oxford, U.K. He has published more than 100 papers on computer vision, computer graphics, and machine learning. He is a coauthor (with J. Ponce) of Computer Vision: A Modern Approach (Englewood Cliffs, NJ: Prentice-Hall, 2002). He was a Professor at the University of California, Berkeley. He is currently a Professor at the University of Illinois at Urbana-Champaign. Prof. Forsyth was Program Cochair for IEEE Computer Vision and Pattern Recognition (CVPR) in 2000, General Cochair for CVPR 2006, and Program Cochair for the European Conference on Computer Vision 2008 and the CVPR 2011. He received an IEEE Technical Achievement Award in 2005.

Derek Hoiem is an assistant professor in Computer Science at the University of Illinois at Urbana-Champaign (UIUC). Before joining the UIUC faculty in 2009, Derek completed his Ph.D. in Robotics at Carnegie Mellon University in 2007, advised by Alexei A. Efros and Martial Hebert, and was a postdoctoral fellow at the Beckman Institute from 2007-2008. Derek's research on scene understanding and object recognition has been recognized with a 2006 CVPR Best Paper award, a 2008 ACM Doctoral Dissertation Award honorable mention, and a 2011 NSF CAREER award.