Comparative object similarity for improved recognition with few or no examples

Gang Wang[1] David Forsyth[2] Derek Hoiem[2]

[1] Dept. of Electrical and Computer Engineering
University of Illinois Urbana-Champaign (UIUC)
gwang6@uiuc.edu

[2] Dept. of Computer Science
University of Illinois Urbana-Champaign (UIUC)

Abstract

Learning models for recognizing objects with few or no training examples is important, due to the intrinsic long-tailed distribution of objects in the real world. In this paper, we propose an approach to use comparative object similarity. The key insight is that: 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. We develop a regularized kernel machine algorithm to use this category dependent similarity regularization. Our experiments on hundreds of categories show that our method can make significant improvement, especially for categories with no examples.

1. Introduction

There are very many object names. Training a system with many examples of each is likely to be difficult (most categories have few examples as shown in Figure 1). Even if we could train such a system, doing so would not yield much insight into how people recognize objects. People seem to manage with few or no *visual* examples, because there is much other information available to help identify objects. An important cue is being told what an object is "like". 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 statement defining a new category in terms of existing categories.

Current methods to exploit similarity information in computer vision cannot deal with such statements. The usual method is metric learning. 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 [10, 26, 25]; analogous procedures can be applied

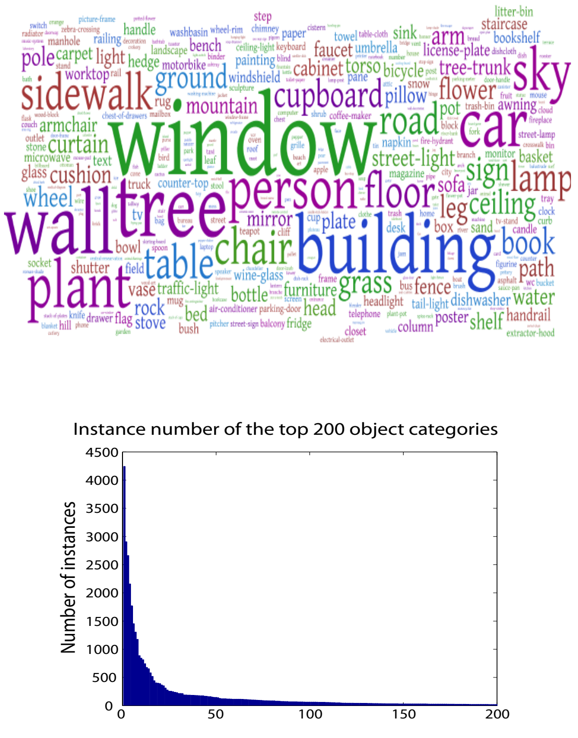

Figure 1. Most categories in the dataset we use (which is a part of Labelme) have few or no examples. The top image shows "object clouds". Objects with bigger names have more instances. Most objects have small names because they have few examples. The bottom image shows number of instances for the top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

to measures of similarity that are not metric [4]. These methods cannot use explicit inter-category information. In the absence of category labels, data-dependent measures of smoothness can be used to weight features [9]. 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 [22] (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.

In this paper, we exploit category similarity on a category by category basis. If we wish to learn a model of a "serval", we obtain a short list of similar categories from a human labeller; this would contain "leopard". We could just use these "leopard" instances as positive examples to train the model. This 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 argue 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 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" than to "grass" and "bird". This process acts as a category dependent similarity regularizer.

We use this category dependent similarity regularization in a kernel machine framework [12]. It reduces the dimensions of the function space for learning. We show that doing so leads to significant performance improvements in hard discrimination tasks, trained with very little data.

Objects might be similar in different ways, for example, a "zebra" is similar to a "horse" in shape, but similar to "crosswalk" in texture. For each aspect, we label similar and dissimilar categories and train a regularized kernel machine. Learned kernel machines are combined as the final output.

The following papers are relevant to our work. Fei-Fei *et al* [7] exploit the tendency of object models to be similar to one another with a strong prior. Discriminative scores can provide more stable features, so fewer examples are required to learn a model [24]. There may be written descriptions of the object to be named, which can be exploited if the object is modelled in terms of *attributes*, properties of objects that can be observed and help describe them [6, 14, 13, 18, 23]. None of this work can exploit the comparative similarity of object categories.

We introduce our approach in Section 2. Experimental settings and results are shown in Section 3. Conclusions are made in Section 4.

2. Learning object models with comparative similarity

We assume we wish to learn a model to identify a named object. We have few or no positive training examples, many negative examples, and some categories identified as "similar" or "dissimilar". We first show how to incorporate this information into the training process for a kernel machine. We then show how to evaluate different aspects of similarity (e.g. similarity in color; in texture; and so on).

2.1. Incorporating comparative similarity into training

We aim to learn an object model F for each name. We follow [27, 15] and use a kernel machine. Write $F = \sum_{i=1}^{N} K(x_i, \bullet)$, where $K(x_i, \bullet)$ is a kernel basis function (we use a histogram intersection kernel). We have a set of T training examples in a feature representation $\{(x_t, y_t), t = 1, ..., T, y_t \in \{+1, -1\}\}$.

The usual procedure is to learn F by minimizing:

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

where $\frac{\lambda}{2}\|F\|^2$ is a regularization term, and L is the hinge loss $L(F(x_t), y_t) = \max(0, 1 - y_t F(x_t))$. However, accurate learning requires numerous positive examples (see section 3).

We do have examples from categories that are similar to the name we seek to learn. A simple approach is to use these as positive instances; we use this approach as a baseline (see section 3). When we append them as positive instances, we weight them with a parameter in the similar form of Equation 3.

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 [11]. 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, so 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. 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 must choose F to minimize

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

where $\alpha(T_p)$ represents the weight on similarity as a function of the number of positive training examples. 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. Generally, if there are few positive examples, $\alpha(T_p)$ should be large, and if there are many, it should be small. We use

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

where A, B and C are parameters chosen by a cross validation procedure applied to set of categories, which have multiple training instances. These parameters are fixed for all other categories. Training is by stochastic gradient descent; details in section 2.3. The resulting F is a ranking function.

2.2. Learning with aspect based similarity

Objects might be similar in different ways, which are called aspects in this paper. Using aspect based similarity could: (1) help human labelers to label similar categories; (2) help design object representation (e.g., if two objects are similar because of color, we should use color features to represent objects and learn F). We have three types of aspects: texture, shape and scene. We label different similar categories and use different features for different aspects: SIFT words for texture, pyramid HOG for shape, and SIFT words on the image for scene. More details can be found in Section 3.2.

Using the idea described in the above section, we train a kernel machine for each aspect of similarity. Each produces a ranking function F_a for a particular aspect a. We can get better results by combining responses produced by multiple aspects based similarity. We calibrate the responses, then sum them to obtain an overall ranking. Calibration is important, because the same degree of similarity might get different values of ranking function for different aspects.

We calibrate by applying a linear transformation to the ranker output so that the strongest (resp. weakest) response on a dataset is 1 (resp 0). The same dataset is used for each aspect, so that the same number of positives should be present in each case, but labels are not known. Again, no more complex procedure is possible, because we have few positive examples.

We linearly combine the calibrated classification responses. The combinations weights are learning using validation on categories with multiple training instances.

2.3. Training

To train the models, we use a stochastic gradient descent method [12, 24]. Stochastic gradient descent selects some terms in the objective function at random and takes a small step in a direction that will minimize them. Our terms could be either loss at a labelled example, or loss for a similar-dissimilar pair. The algorithm becomes:

At the ith iteration, select a single loss term at random:

1. If this is loss at a labelled example x_t, then follow the procedure of [12, 24]. The update is:

$$F_i = (1 - \lambda\eta_i)F_{i-1} + \frac{1}{T}\eta_i\sigma_i y_t K(x_t, \bullet) \quad (4)$$

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

2. If we have a similar-dissimilar pair $\{g_n^s, g_n^d\}$, the term $\widehat{O}$ for that pair is

$$\frac{1}{N}\alpha(T_p)L(F(g_n^s) - F(g_n^d), 1) + \frac{\lambda}{2}\|F\|^2 \quad (5)$$

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

$$\frac{1}{N}\alpha(T_p)\sigma_i(K(g_n^d, \bullet) - K(g_n^s, \bullet)) + \lambda F_{i-1} \quad (6)$$

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

$$F_i = (1 - \lambda\eta_i)F_{i-1} + \frac{1}{N}\alpha\eta_i\sigma_i(K(g_n^s, \bullet) - K(g_n^d, \bullet)) \quad (7)$$

We use a histogram intersection kernel for K, allowing us to use the fast training algorithm of [24]. We modify their public training code to learn our models; an alternative is to use the method of [17].

object	similar categories	dissimilar categories	similarity type
shrub	flower, grass, hedge	bowl, vas, 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. Example categories and their general similarity annotation. The similarity type is labeled by a second volunteer.

3. Experiments

3.1. Procedures

Dataset: We use 2831 images from Labelme [20] as a test bed. Object regions are fully annotated within images. We manually reword the 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 our dataset are 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.

General similarity annotation: We select 90 object categories, which have more than 60 instances, as prototype categories. We use 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 these prototype categories without seeing any images from the dataset. In this labeling procedure, no extra instructions are given on which aspect (e.g., shape or texture) should be used to judge similarity. So it is called "general similarity", in contrast with the "aspect based similarity" described below.

This work 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 labeller felt both flap in the sky); and very different (e.g. "ceiling fan";"flower"). More examples on similarity annotation are shown in Table 1.

Aspect based similarity: In addition to the general similarity labeled for 225 categories, we also label aspect based similarity for 50 categories. As mentioned before, there are three types of aspects: texture, shape and scene. The similarity is also labeled without seeing any images in the dataset. Scene similarity is obtained by finding prototype objects which tend to appear in the same scene (scene types are mainly restricted to "indoor" or "outdoor") . All images containing instances of these prototype objects are of similar scenes.

3.2. Features and parameters

For general similarity and texture similarity, we represent objects as a 800-dimensional histogram of SIFT [16] words. For scene similarity, we also represent images as a 800-dimensional histogram of SIFT words. For shape-based similarity, we use the PHOG (for Pyramid of Histograms of Orientation Gradients) described in [3].

We use around 20,000 negative examples and 20,000 pairs of similar-dissimilar examples to train the models. When training the kernel machines, 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.

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. Since our goal in this paper is to investigate how similarity helps to categorize uncommon objects rather than detect objects as in [8], we directly use the ground truth segmentations of test images to extract object regions. There are 21,803 test regions in total.

3.3. 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. **Baseline 2** uses instances from similar categories as positive examples (weighted with a weight in the similar form of Equation 3). For aspect based similarity, each baseline is obtained as described for each aspect.

3.4. General 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 the method will correctly identify them. 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

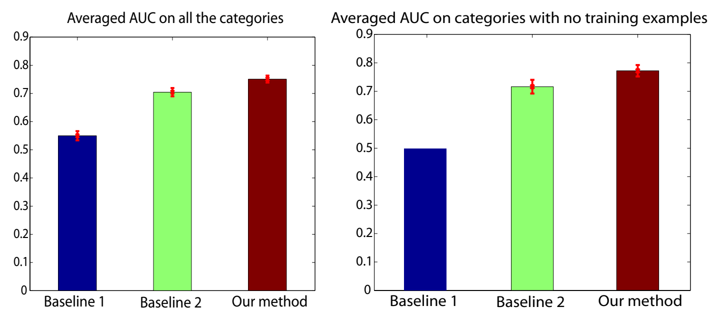

Figure 2. General similarity improves AUC, even when there are no examples. On the **left**, AUC averaged over all the 225 test categories for baselines and for our method; on the **right** this comparison for the 110 test categories which have *no* positive examples, both with standard error bars.

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 super common categories. Our general similarity method produces strong improvements in AUC for all test categories, especially when there are no positive examples (Figure 2). AP is a less helpful measure, because there are very few positive examples and approximately 20, 000 negative examples so all scores are very small and unstable.

We also show some qualitative results in Figure 3. 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 they have no training examples. Figure 4 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 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 5. We can see that even if the similar prototypes are at best “different” or “very different”, using similarity yields a better AUC.

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

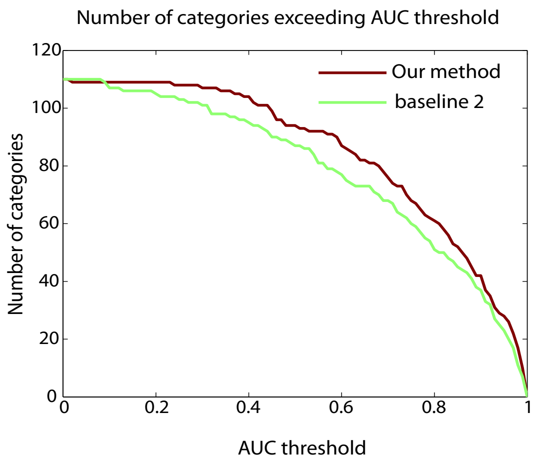

Figure 4. Number of categories for which AUC is greater than a threshold. These categories have no positive training examples. The x-axis denotes the AUC thresholds. The y-axis denotes the number of categories whose AUC values exceed the thresholds. There are more than 40 categories with bigger AUC values than 0.9 using our method. Our method consistently gets more categories than baseline 2 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.

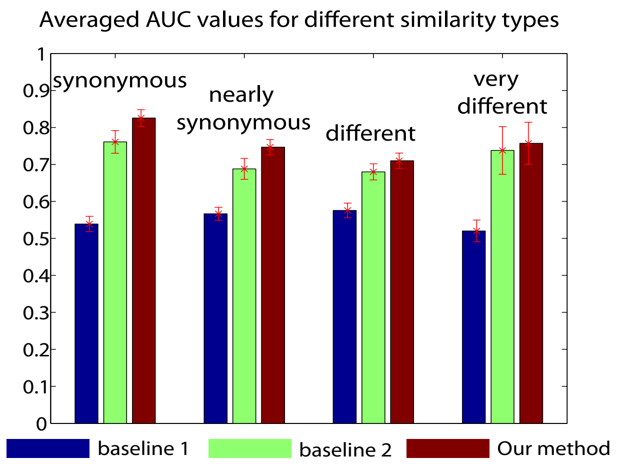

Figure 5. General similarity improves average AUC score; the effect is not due to synonymy. We show the average for all categories with at least one synonymous similar prototype; all with at least one near synonymous similar prototype and no stronger; all with at least one different similar prototype 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.

matters rather than the number of similar categories.

3.5. Aspect based similarity improves AUC

We test Multi-aspect similarity on 50 categories (mainly indoor objects such as bookcase and dish towel). When

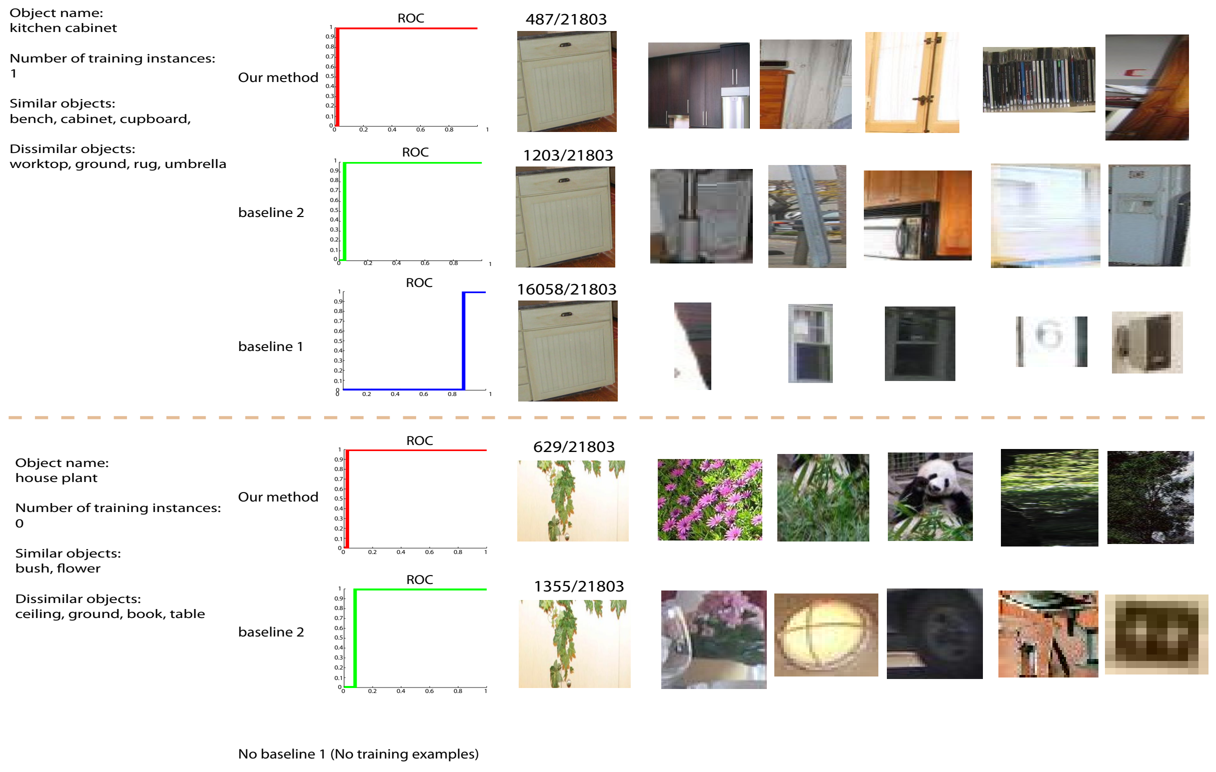

Figure 3. Classification results for two categories. For each category, the first row shows results using our method; the second row shows results using baseline 2; the third row shows results using baseline 1 (if there are positive training examples). At each row, the first figure shows the ROC curve; 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.

combining classification scores from different aspects, we first calibrate them as introduced in Section 2.2. Then we weight them linearly. The weights are learned using validation set on 5 categories, which at least 2 training examples. The learned weights are 0.554, 0.341 and 0.106, for texture aspect, shape aspect and scene aspect respectively. The results are shown in Table 2. Combining cues from different aspects helps. The scene cue is useful, especially for objects which only appear in specific scenes. If one object appears in very diverse scenes (such as "person"), the scene cue is not going to help.

Within these 50 categories, there are 27 categories which are also labeled with the general similarity. The average AUC of these 27 categories are compared in Table 3.

T	Sh	Sc	T+Sh+Sc
0.735	0.725	0.686	0.783

Table 2. Average AUC values on 50 categories for different aspect similarity and their combinations. "T" denotes the result using texture based similarity, "Sh" denotes the result using shape based similarity, "Sc" denotes the result using scene based similarity. Combining multiple aspects helps.

G	T	Sh	Sc	T+Sh+Sc
0.769	0.748	0.702	0.653	0.771

Table 3. Average AUC values on 27 categories for general similarity, different aspect based similarity and their combination. "G" denotes the result using general similarity. Combining multiple aspects works comparably well.

3.6. General 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 [19]), where weights are given by calibrated classification scores.

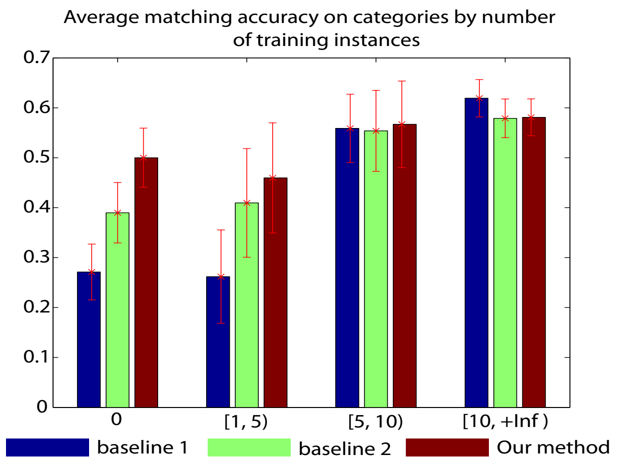

Figure 6. Average **matching accuracy** on categories by number of training instances. On the categories with zero or few training examples, using similarity helps a lot. Our method also gets better results than baseline 2.

Baseline 1	Baseline 2	Our method
0.440	0.486	0.535

Table 4. Average **matching accuracy** using classification scores by different methods. The accuracy is averaged over categories. Using our method can establish better correspondence.

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 4).

In Figure 6, 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. This is important because the distribution of objects in the real world is long-tailed. Correspondence examples are shown in Figure 7.

4. Conclusion and discussion

An opportunistic model of comparative object similarity that acts as a category dependent regularizer produces significant improvements in AUC and correspondence for hundreds of categories with few or no training examples. The model is wholly general and should apply to a wide variety of problems.

One problem of the current approach is that learned models might confuse test instances from similar categories with target ones. One potential solution is to partition categories using taxonomy [2, 21], 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.

We also plan to extend our aspect based similarity approach by combing it with the visual attributes ideas [6, 14]. Visual attributes define more specific aspects.

5. Acknowledgement

The authors would like to thank Jiahui Shi's help for this project.

This work was supported in part by the National Science Foundation under IIS-0534837 and 0803603 and in part by the Office of Naval Research under N00014-01-1-0890 as part of the MURI program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the NSF or the ONR.

References

[1] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D.M. Blei, and M.I. Jordan. Matching words and pictures. *JMLR*, 2003. 6

[2] E. Bart, I. Porteous, P. Perona, and M. Welling. Unsupervised learning of visual taxonomies. In *CVPR*, 2008. 7

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*. ACM, 2007. 4

[4] Daphna Weinshall David W. Jacobs and Yoram Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. In *PAMI*, 2000. 1

[5] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Proc. CVPR*, 2009. 2, 7

[6] L. Fei-Fei, R. Fergus, and P. Perona. One-Shot learning of object categories. *PAMI*, 2006. 2

[7] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. *CVPR*, 2008. 4

[8] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised Learning in Gigantic Image Collections. In *NIPS*, 2009. 1

[9] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally consistent local distance functions for shape-based image retrieval and classification. In *Proc. ICCV*, 2007. 1

[10] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *KDD*, 2002. 2

[11] J. Kivinen, AJ Smola, and RC Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004. 2, 3

[12] N. Kumar, A.C. Berg, P.N. Belhumeur, and S.K. Nayar. Attribute and Simile Classifiers for Face Verification. *ICCV*, 2009. 2

[13] C.H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Proc. CVPR*, 2009. 2, 7

[14] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2

[15] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004. 4

[16] S. Maji and A.C. Berg. Max-Margin Additive Classifiers for Detection. ICCV, 2009. 3

[17] M. Palatucci, D. Pomerleau, G. Hinton, and T.M. Mitchell. Zero-Shot Learning with Semantic Output Codes. In *NIPS*, 2009. 2

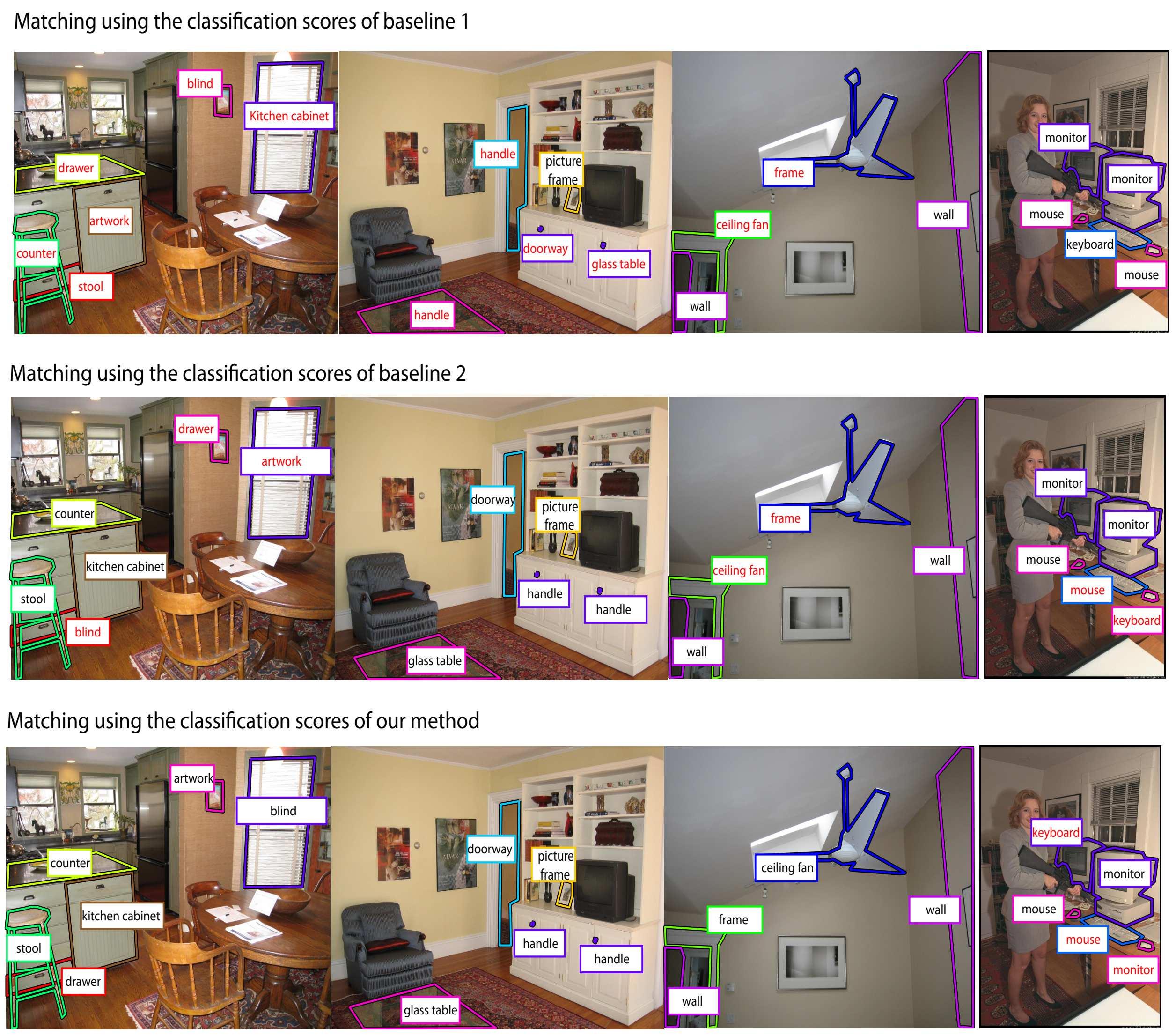

Figure 7. Examples showing found correspondence between regions and weak class labels. On each image, regions are depicted by polygons in different colors, the found corresponding class names are surrounded by squares in the same colors. Incorrect correspondence is indicated with red object names. Each 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).

[18] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Pubns, 1998. 6

[19] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. LabelMe: a database and web-based tool for image annotation. *IJCV*, 2008. 4

[20] J. Sivic, B.C. Russell, A. Zisserman, W.T. Freeman, and A.A. Efros. Unsupervised discovery of visual object class hierarchies. In *CVPR*, 2008. 7

[21] A. Tversky et al. Features of similarity. *Psychological review*, 1977. 2

[22] G. Wang and D. Forsyth. Joint learning of visual attributes, object classes and visual saliency. In *ICCV*, 2009. 2

[23] G. Wang, D. Hoiem, and D. Forsyth. Learning Image Similarity from Flickr Groups Using Stochastic Intersection Kernel Machines. In *ICCV, 2009*. 2, 3

[24] K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009. 1

[25] 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. 1

[26] 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. 2