Image Retrieval and Classification Using Local Distance Functions

Andrea Frome
Department of Computer Science
UC Berkeley
Berkeley, CA 94720
afrome@cs.berkeley.edu

Yoram Singer
Google, Inc.
Mountain View, CA 94043
singer@google.com

Jitendra Malik
Department of Computer Science
UC Berkeley
malik@cs.berkeley.edu

Abstract

In this paper we introduce and experiment with a framework for learning local perceptual distance functions for visual recognition. We learn a distance function for *each individual training image* as a combination of elementary distances between visual features. We apply these combined local distance functions to the tasks of image retrieval and classification of novel images. On the Caltech 101 object recognition benchmark, we achieve 59% recognition using 15 training images, which matches the best published performance by Zhang, et al.

1 Introduction

Visual recognition poses many challenges for machine learning techniques. Two particularly noteworthy ones are:

(1) There is a large number of diverse classes with large intraclass variation. Estimates of how many categories can be distinguished by humans range from 30,000 to 100,000, and that is before we consider the identification problem, for example distinguishing among the thousands of faces with which we may be familiar. The variation in a category can be considerable due to pose, lighting and internal articulation as in human or animal figures.

(2) Distances measured between shape features often are only meaningful for small distances. It is generally accepted that the most important cue for visual recognition is shape. Embedding visual shape into a global vector space is problematic, but locally we might expect a friendlier manifold structure. It is convenient to capture this local structure by means of a local "perceptual distance" function around an exemplar image.

A central goal of this paper is to develop and experiment with a framework for learning such local perceptual distance functions in the context of visual recognition. There will be as many of these as there are exemplar images, and for a given exemplar image, its distance function is trained by the guiding principle that the perceptual distances to positive examples (in the same category as the exemplar) should be smaller that the perceptual distances to negative examples (from all other categories). A distance function for a particular exemplar ("focal image") is a linearly weighted combination of elementary feature distance functions, each of which is based on comparisons between image patches. We learn these weights using a variant of the constrained optimization formulation proposed by Schultz and Joachims [9] for relative comparison data.

Using these local distance functions, we address applications in image browsing, retrieval and classification. We show results on the Caltech 101 object recognition benchmark, that has now become a *de facto* standard for multi-category classification. The classification performance on this benchmark is 59% using only fifteen exemplar images per category, which matches the best published recognition rate in [12].

2 Visual Features

The rest of the paper presents a method for learning local distance functions for a set of *focal images*, and using those local distance functions for ranking, retrieval, and classification. Our method is general in that it can be used with any feature type. One strength of the method is that, for a given focal image, it chooses the features that best capture the similarity of that image to other images. Patch-based features are an obvious choice, in part because their locality allows our algorithm to choose the pieces of the image that are most salient. Another strength of the method is that it naturally allows for the combination of different types of features. To demonstrate this, we use a combination of shape and color patch-based features.

Many papers have shown the benefits of using filter-based patch features such as *SIFT* [7] and *geometric blur* features [1] for shape- or texture-based object matching and recognition [6][4][1]. We chose to use geometric blur descriptors, which were used by Zhang et al. in [12] in combination with their KNN-SVM method to give the best published results on the Caltech 101 image recognition benchmark. Like SIFT, geometric blur features summarize oriented filter responses within a patch of the image, but are designed to be more robust to affine transformation and differences in the periphery of the patch. For a full description of geometric blur descriptors, see [1]. In previous work using geometric blur descriptors on the Caltech 101 data set [1] [12], the patches used are centered at some number of edge points sampled from the image, and features are computed on patches of a fixed scale and orientation. We follow this methodology as well, though one could use an interest point operator to determine scale and orientation from low-level information, as is typically done with SIFT features. We use two different scales of geometric blur features, the same as used in [12]. The larger uses a patch radius of 70 pixels, and the smaller uses a patch radius of 42 pixels. Both use four oriented filters and 51 sample points, to give features with 204 dimensions. As is done in [1], we default to normalizing the feature vector so that the L_2 norm is equal to one.

For color, we computed color histograms for eight-pixel radius patches centered at edge pixels in the image. Any "pixels" in a patch that were off the edge of the image were counted in a "undefined" bin, and we converted the HSV coordinates of the remaining points to a Cartesian space where the z direction is value and (x, y) is the Cartesian projection of the radial hue/saturation dimensions. We divided the (x, y) space into an 11×11 grid, and made three divisions in the z direction. These were the only parameters that we tested with the color features, and while we could have used cross validation to choose the best parameters, we chose not to so that we could highlight the performance of our algorithm without optimizing for the Caltech 101 data set. We normalize the bins by the total number of pixels in the patch. [1]

Our learning algorithm in the next section learns, for a given *focal image* $\mathcal{F}$, a weighting over elementary distance functions which are computed between the focal image $\mathcal{F}$ and another image $\mathcal{I}$. Again, our method is general in that any elementary distance function over any image feature type can be used, so long as it returns a non-negative value. For a given focal image and any other image $\mathcal{I}$, we have $j \in [1, M]$ such elementary distance measures, and we denote each $d_j^{\mathcal{F}}(\mathcal{F}, \mathcal{I})$. [2] In our experiments, we use an elementary distance function for each of our patch features. The elementary distance function for the jth patch is the smallest L_2 distance between the jth feature in $\mathcal{F}$ and the set of features of the same type (e.g., large geometric blur feature) from the other image. Expressing this formally, if we take $\mathbf{p}_j^{\mathcal{F}}$ to be jth patch feature from image $\mathcal{F}$, and $\{P^{\mathcal{I}}\}$ to be the set of features

[1] As a post-processing step, we normalized the distances between color features to have the same standard deviation as the distribution over distances between shape features in the training images.

[2] We use $\mathcal{F}$ in both the superscript and as an argument to the function to emphasize that the elementary distance function is both particular to the focal image and computed on its contents.

from $\mathcal{I}$ that are of the same type as $\mathbf{p}_j^{\mathcal{F}}$ (e.g. the same scale geometric blur feature):

$$d_j^{\mathcal{F}}(\mathcal{F},\mathcal{I}) = \min_{\mathbf{p}^{\mathcal{I}} \in \{P^{\mathcal{I}}\}} \sqrt{\left\|\mathbf{p}_j^{\mathcal{F}} - \mathbf{p}^{\mathcal{I}}\right\|^2} \tag{1}$$

Note that this is an asymmetric distance, and that in general our method can use any distance measures, including ones that take into account geometric relationships between patches.

3 Learning To Combine Elementary Distance Functions

In this section we describe the core mechanism of our retrieval and classification. We have a training set of focal images, and for each we want to learn a distance function that takes any image and returns a non-negative number. For a focal image $\mathcal{F}$ and any other image $\mathcal{I}$, we denote this function as $D^{\mathcal{F}}(\mathcal{F},\mathcal{I})$. We could use such a function to rank any set of images with respect to the focal image $\mathcal{F}$, and ideally the resulting ordering would rank the images similar to $\mathcal{F}$ ahead of dissimilar images. The input to the learning problem will be derived from a rank ordering over the training images, though the rank ordering can be very coarse, as it is in our experiments.

Naturally, we want this function to be based on the content of the images, and choose a linear combination of elementary distance measures computed from image content features, such as those described in the previous section. The learning goal is to find a non-negative set of weights that combine the M elementary distance functions into one distance function, where we denote the jth elementary distance function computed for the jth feature from the focal image $\mathcal{F}$ by $d_j^{\mathcal{F}}(\mathcal{F},\mathcal{I})$:

$$D^{\mathcal{F}}(\mathcal{F},\mathcal{I}) = \sum_{j=1}^{M} w_j^{\mathcal{F}} d_j^{\mathcal{F}}(\mathcal{F},\mathcal{I}) = \left\langle \mathbf{w}^{\mathcal{F}} \cdot \mathbf{d}^{\mathcal{F}}(\mathcal{F},\mathcal{I}) \right\rangle \tag{2}$$

We would like to emphasize that both the set of elementary distance functions and the learned weights are particular to the focal image $\mathcal{F}$. We need a learning algorithm to learn the weight vector $\mathbf{w}^{\mathcal{F}}$ which attains the following properties: (1) the algorithm should enforce that $\forall j$, $w_j^{\mathcal{F}}$ is non-negative; [3] (2) it should generalize well from a fairly small set of training images, such that a novel image that is similar to the focal image is ranked well; (3) it should work with *any* elementary distance functions between visual features; and (4) the algorithm should be able to cope with data expressed as pairs of images that are more and less similar than the focal image, rather than strict positive and negative examples. [4]

Properties (3) and (4) led us to a formulation where the input to our algorithm is derived from *triplets* of images, as is also used by Schultz and Joachims in [9]. Assume that we have a lexicographical ordering over images $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_N$. Given that image $\mathcal{I}_i$ is an image "more similar" to the focal image $\mathcal{F}$ than image $\mathcal{I}_j$, we have a triplet $(\mathcal{F},\mathcal{I}_i,\mathcal{I}_j)$ such that we would ideally want

$$D^{\mathcal{F}}(\mathcal{F},\mathcal{I}_j) > D^{\mathcal{F}}(\mathcal{F},\mathcal{I}_i) \ , \tag{3}$$

so that a rank ordering based on our distance function places these two images in their correct relative positions. In equation (2) we defined this distance function to be weighted combination of individual measures, so this condition is equivalent to $\left\langle \mathbf{w}^{\mathcal{F}} \cdot \mathbf{d}^{\mathcal{F}}(\mathcal{F},\mathcal{I}_j) \right\rangle > \left\langle \mathbf{w}^{\mathcal{F}} \cdot \mathbf{d}^{\mathcal{F}}(\mathcal{F},\mathcal{I}_i) \right\rangle$ or $\left\langle \mathbf{w}^{\mathcal{F}} \cdot \mathbf{x}_{i,j} \right\rangle > 0$, where $\mathbf{x}_{i,j} = \mathbf{d}^{\mathcal{F}}(\mathcal{F},\mathcal{I}_j) - \mathbf{d}^{\mathcal{F}}(\mathcal{F},\mathcal{I}_i)$ captures the relative rank of the triplet $(\mathcal{F},\mathcal{I}_i,\mathcal{I}_j)$, and is one vector in the set of training vectors $T^{\mathcal{F}}$ input to our algorithm. While it would be nice to satisfy Equation (3) precisely, in practice our data is very noisy by nature and it is impossible to find weights which satisfy the constraints for all possible triplets. In addition, we would like to constrain the norm of $\mathbf{w}^{\mathcal{F}}$ in order to guard from overfitting (see property (2)). We thus cast the problem as a constrained optimization problem with a slack variable associated with each triplet so as to allow some of the distance constraints to be violated. We arrive at the following

[3] The positivity constraint is not strictly necessary for the learning framework, but is an intuitive constraint and in practice has the effect of removing confounding features and promoting sparsity.

[4] While we don't fully exploit this last property in the experiments in this paper, it allows the technique to generalize to settings where the training data does not come from a binary source, such as user feedback or click streams.

maximal margin formulation which employs slack variables for each of the triplets and includes a primal positivity constraint on $\mathbf{w}$:

$$\arg\min_{\mathbf{w}^{\mathcal{F}},\boldsymbol{\xi}} \quad \frac{1}{2}\left\|\mathbf{w}^{\mathcal{F}}\right\|^2 + C\sum_{ij}\xi_{ij} \tag{4}$$

$$\text{s.t.}: \qquad \forall(i,j)\in T^{\mathcal{F}} : \left\langle \mathbf{w}^{\mathcal{F}}\cdot\mathbf{x}_{i,j}\right\rangle \geq 1-\xi_{ij}, \xi_{ij}\geq 0, w_k^{\mathcal{F}}\geq 0 \tag{5}$$

We chose to use L2 regularization instead of the L1 to be more robust to noise, perhaps at the expense of increased sparsity.

The constrained optimization problem defined above is a close variant of that proposed by Schultz and Joachims in [9] for distance metric learning. However, our setting is different from theirs in two ways. First, their triplets do not share the same focal image since they apply their method to learning one metric for all classes and instances. Second, they arrive at their formulation by assuming that the elements of their distance vectors fit the form $d_k(\mathcal{I}_i,\mathcal{I}_j) = (p_k^{\mathcal{I}_i} - p_k^{\mathcal{I}_j})^2$, where $p_k^{\mathcal{I}_i}$ is the kth element of the feature vector for the item $\mathcal{I}_i$. This assumption amounts to the restrictions that (1) each feature from our image be a single number, and (2) we only use a L_2^2 distance between these features. This would appear to preclude our use of patch features and more interesting distance measures, however we have shown that this is an unnecessary restriction on the algorithm. Thus, a contribution of this paper is to show that the algorithm in [9] is more widely applicable than originally presented, thus making it more useful for difficult machine vision problems.

We used a custom solver for the optimization problem, which runs on the order of a second for about 3,000 triplets. The dual optimization includes a dual variable $\alpha_{i,j}$ for each triplet and a dual variable $\boldsymbol{\mu}$, which enforces the positivity constraint on $\mathbf{w}^{\mathcal{F}}$. In each epoch of training, we iterate over the set of $\alpha_{i,j}$ variables that violate the KKT constraints for our problem, and for each, we first increase the dual with a closed-form update to α_{ij}, then we update $\boldsymbol{\mu}$ to project the current solution into the feasible region. This approach is similar to the row action approach described in [2]. [5]

4 Using Distance Functions for Browsing, Retrieval, and Classification

Given a set of K training images, we can use each as a focal image $\mathcal{F}_k$ and use the remaining $K-1$ images to learn the distance function $D^{\mathcal{F}_k}(\mathcal{F}_k,\cdot)$. Each of the K distance functions that we learn induces a ranking over the other $K-1$ images. In the next few sections, we will discuss how we can leverage this rich source of information for content-based image applications.

4.1 Image Rankings for Image Browsing

If we have rankings on a closed set of K images, we can create a simple image browsing application that captures the similarity relationships between these images. The user starts on a page showing the ranking for one of the K images. If the user clicks on any of the images in the ranking, they are shown the ranking for that image. This allows the user to navigate "image space" using the local distance functions that we have learned. Figure 1 shows one such ranking learned from a subset of the Caltech101 data set. In our supplemental material, we provide HTML pages showing rankings for a subset of the Caltech 101 images, and the user can navigate to the rankings for other images for which we were able to supply the ranking pages. [6] [7] We do not have a quantitative evaluation of these rankings, but one can get a qualitative idea of how the images have been organized by the learning algorithm. These pages also serve as a nice visualization of the learned image similarity functions, and are helpful as a basis for understanding how we leverage the rankings for image retrieval and classification.

[5] In the Appendix to the paper, included as `appendix.pdf` in the supplemental materials, we give the derivation of the algorithm we used to solve the optimization.

[6] The number of pages is limited due to the restriction on the size of the supplemental materials.

[7] Unzip `browse_localmetrics.tgz` and view `file:///XXX/archive_localmetrics/browse/index.html` in your browser, where `XXX` is the directory into which you expanded the tar file.

4.2 Image Retrieval from Distance Functions

Given the K distance functions and a new query image $\mathcal{Q}$, we would like to return a listing of the K training images in order of similarity to $\mathcal{Q}$. While we can use the K distance functions to compute the distance from each of the focal images $\mathcal{F}_k$ to $\mathcal{Q}$, these distances are on different scales and are not directly comparable. This is because (1) the weight vectors for each of the focal vectors are not constrained to share any properties other than non-negativity, and (2) the number of elementary distance functions and the elementary functions themselves are different for each focal image. This challenge is a research problem unto itself, and for the scope of this work, we employed a combination of two simple heuristics that works surprisingly well. We hope that these heuristics provide insight into more principled solutions.

The first heuristic attempts to rescale the K focal image distance functions to make them more comparable. For each focal image, we divide the distance function by the distance to the closest image in the learning set, thus making the smallest distance to the focal image the unit distance. Thus, if i ranges over the images used to learn the kth distance function, the new distance function would be $D(\mathcal{F}_k, \mathcal{Q})$ divided by $\min_{\mathcal{I}_i \neq \mathcal{F}_k} D(\mathcal{F}_k, \mathcal{I}_i)$. If the distance to the closest training image is zero, then we take the distance of the closest training image that is nonzero.

Another approach would be, for each test image $\mathcal{Q}$, to compute some measure of confidence for each focal image $\mathcal{F}_i$, which could be incorporated into the score for each focal image used to rank them relative to one another for $\mathcal{Q}$. In this spirit we developed a second heuristic which approximates the quality of the ranking of a test image relative to a focal image by simply counting the number of out-of-class training images that were ranked above the test image by that focal image. [8] If $\mathcal{Q}$ is very similar to $\mathcal{F}_k$, and the distance function learned for $\mathcal{F}_k$ captures this, then there should be few dissimilar training images ranked above $\mathcal{Q}$, thus the larger the value, the less similar we believe $\mathcal{Q}$ is to $\mathcal{F}_k$, relative to the other focal images. For example, Figure 1 shows the raw distances for each of the images to the focal image in the upper-left corner. There are two negative training examples in this ranking, the lotus in the 11th position, and the sunflower in the 12th position. All test images before the 11th position would be given an error penalty of zero, and the test image of the sunflower in the 13th position would be given an error penalty of two.

These two heuristics are complementary, and we combined them in an ad-hoc manner to generate a score for each test image $\mathcal{Q}$ to each focal image $\mathcal{F}_k$ by simply multiplying the normalized distance by the error penalty plus one (to avoid zeros). We do not quantitatively evaluate the performance on the retrieval task, but describe in the next section how we use these retrieval rankings to perform classification on the Caltech101 data set. These heuristics are the weakest part of our method, and a better algorithm for comparing new images across spaces is likely to even further improve performance both in retrieval and recognition. We have in the K rankings of the other $K-1$ images a very rich source of information about how all the training images relate to one another, and our heuristics only make use of a small portion of that information.

4.3 Image Classification from Image Retrieval

If we have methods for determining image similarity and performing retrieval that perform well, then classification can simply be a post-process on ranked retrieval lists when labels are available for the training data. From the scores computed between a query image $\mathcal{Q}$ and each focal image as described in the last section, we have an ordering over our training images. Given class labels for the training images, we can use a nearest neighbor classifier to assign a class label to $\mathcal{Q}$. In our experiments, we use a variant of a 2-NN classifier where, if we do not find two labels that agree in the first three items of the list, then we continue looking down the list to find the first two that agree. If there are not two that agree within the top ten items of the list, we assign the label from the first in the list.

[8] If the training data isn't in the form of in- and out-of-class examples, then we could instead count the number of similar-dissimilar inversions in the ranking.

5 Caltech101 Experiments

We test our approach on the Caltech101 data set [3][9]. This data set has artifacts that make a few classes easy, but many are quite difficult, and due to the important challenges it poses for scalable object recognition, it has up to this point been one of the *de facto* standard benchmarks for multi-class image categorization/object recognition. The dataset contains images from 101 different categories, with the number of images per category ranging from 31 to 800, with a median of about 50 images. We ignore the background class and work in a forced-choice scenario with the 101 object categories, where a query image must be assigned to one of the 101 categories.

We use the same testing methodology and mean recognition reporting described in Grauman et. al. [4]: we use varying numbers of training set sizes (given in number of examples per class), and in each training scenario, test with all other images in the Caltech101 data set, except the `BACKGROUND_Google` class. Recognition rate per class is computed, then averaged across classes. This normalizes the overall recognition rate so that the performance for categories with a larger number of test images does not skew the mean recognition rate.

5.1 Training data

We begin with resized versions of the images. The aspect ratio is maintained, but all images are scaled down to be around 200×300. We computed features for each of these image as described in Section 2. We computed at most 400 of each type of feature (two sizes of geometric blur and one color), for a maximum total of 1,200 features per image. For images with few edge points, we computed fewer features so that the features were not overly redundant.

A given run of the learning algorithm is always with respect to one focal image $\mathcal{F}_k$, so that if we train with 15 images from each of the 101 classes, we run our learning algorithm 1,515 times. For each focal image we choose a set of more/less similar triplets for training, and since we are learning similarity for the purposes image classification, we use the category labels on the images in the training set; images that have the same label as the focal image are considered more similar than all images that are out of class. Note that our use of triplets allows for a more nuanced training set where an image could be more similar with respect to one image and less similar with respect to another, but we are not fully exploiting that in these experiments.

For each focal image, we use only a subset of the full pairwise combination of all similar and dissimilar images. For clarity, we refer to *all* the images available for training as the "training set" (e.g. 1,515 images if we are training with 15 images per category), and those that are used as input to learning for a given focal image as the "learning set" for that focal image. We want in our learning set for a focal image $\mathcal{F}$ those images that are similar to the focal image according to one of our individual distance measures $d_k^{\mathcal{F}}(\mathcal{F}, \cdot)$. For each of the M distance measures, we take the top N closest images given by $d_k^{\mathcal{F}}(\mathcal{F}, \cdot)$. If that group contains both in- and out-of-class images, then we make triplets out of the full bipartite match. If all N images are in-class, then we find the closest out-of-class image according to that distance measure and make N triplets with one out-of-class image and the N similar images. We do the converse if all N images are out of class. In our experiments, we used $N = 5$, and we have not yet performed experiments to determine the effect of the choice of N. The final set of triplets for $\mathcal{F}$ is the union of the triplets chosen by the M measures. On average, we used 2,210 triplets per focal image, and mean training time was 1 second.

5.2 Results

We ran a series of experiments, each with a different number of training images per category (either 5, 10, 15, 20, or 30), where we generated 10 independent random splits of the 8,677 images from the 101 categories into training and test sets. We report the average of the mean recognition rates across these splits as well as the standard deviations. We determined the C parameter of the training algorithm using leave-one-out cross-validation on a small random subset of 15 images per category, and our final results are reported using the best value of C found (0.01). In general, however, the

[9] Information about the data set, images, and published results can be found at `http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html`

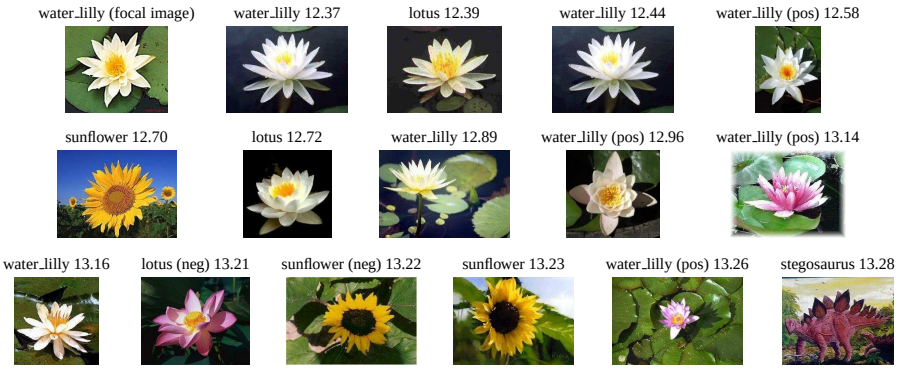

Figure 1: The first 15 images from a ranking induced for the focal image in the upper-left corner, using 15 training/category regime on a subset of the Caltech101 data set. Each image is shown with its unnormalized distance, and only those marked with (pos) or (neg) were in the learning set for this focal image. A longer version of the ranking for this image and the rest of this subset can be seen in the supplemental materials by starting at `browse/index.html`.

method was robust to the choice of C, with only changes of about 1% in recognition with orders of magnitude differences in C near the maximum.

In the 15 training images/category setting, we performed recognition experiments on each of our features separately, the combination of the two shape features, and the combination of two shape features with the color features, for a total of five different feature combinations. Recognition in the color-only experiment was the poorest at 13.0%, and in only thirteen of the categories were one-third or better of test images labeled correctly: `Leopards`, `airplanes`, `butterfly`, `car_side`, `dollar_bill`, `garfield`, `hawksbill`, `pizza`, `snoopy`, `stop_sign`, `strawberry`, `sunflower`, and `yin_yang`. Note that all images in the `car_side` category are black and white, and that many `hawksbill` and `airplane` images have blue backgrounds. The next best performance was from the small geometric blur features with 50.4% (0.5% std), followed by the large geometric blur features with 51.4% ($\pm 0.8\%$). Combining the two shape features together, we achieved 58.2% ($\pm 0.7\%$), and with color and shape, reached 59.1% ($\pm 0.8\%$), which matches the best published performance for 15 training images on the Caltech 101 data set[12]. The combined shape and color was better than color alone for almost all categories, except `hawksbill`, `lotus`, and `sunflower`, which did not improve with the addition of shape, and `pizza`, which actually degraded. [10] The combined shape and color was better than the two shape features alone for 39 of the categories, while it degraded performance for 34 of the categories, and did not change performance in the remaining 28. In Figure 5.2 we show the confusion matrix for combined shape and color using 15 training images per category. Also in that figure is a graph based on that in [12] that shows most of the published results for Caltech101 and our performance using 5, 10, 15, 20, and 30 training examples and all three features.

Almost all the processing at test time is the computation of the elementary distance functions between the focal images and the test image. In practice the weight vectors that we learn for our focal images are fairly sparse, with a median of 69% of the elements set to zero after learning, which greatly reduces the number of feature comparisons performed at test time. We measured that our unoptimized code takes about 300 seconds per test image. [11] After comparisons are computed, we only need to compute linear combinations and compare scores across focal images, which amounts to negligible processing time. This is a benefit of our method compared to the KNN-SVM method of Zhang, et al.[12], which achieves the same recognition rate, but requires the training of a multiclass SVM for every test image, and must perform all feature comparisons.

[10] The distance measure we use combined with the normalization of the geometric blur features seems to make the focal images from many categories look like pizza.

[11] To further speed up comparisons, in place of an exact nearest neighbor computation, we could use approximate nearest neighbor algorithms such as locality-sensitive hashing or spill trees.

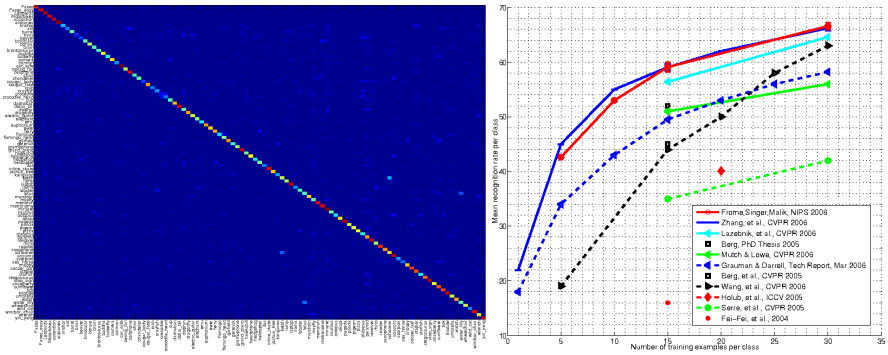

Figure 2: Average recognition rate across classes versus the number of training examples, based on the graph in [12]. Results from [12], [6], [8], [4], [1], [11], [5], [10], [3].

6 Conclusion

There are two main contributions in this paper. First, we show the usefulness of simple locally-defined distance functions for capturing similarity structure between images and demonstrate that such an approach does not necessarily overfit. This provides an alternative to the popular approach of learning a single metric for all classes, making it better suited to the vision setting where variation can be large even within a visual category. Second, we demonstrate that using these local distance metrics and a very simple heuristic, we can perform image retrieval as well as image classification. On the Caltech 101 object recognition benchmark, we are able to achieve a recognition performance of 59% using only fifteen training images per category, which matches the best reported performance. Furthermore, these results indicate that replacing the simple-minded heuristics we use to leverage our distance functions for retrieval with more principled techniques would further utilize the rich set of ranking information that we have learned and further improve retrieval and recognition.

References

[1] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *CVPR*, 2005.

[2] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1998.

[3] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach testing on 101 object categories. In *Workshop on Generative-Model Based Vision, CVPR*, 2004.

[4] K. Grauman and T. Darrell. Pyramic match kernels: Discriminative classficiation with sets of image features (version 2). Technical Report MIT_CSAIL_TR_2006-020, MIT, March 2006.

[5] A. D. Holub, M. Welling, and P. Perona. Combining generative models and fisher kernels for object recognition. In *ICCV*, 2005.

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

[7] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1000–1015, Sep 1999.

[8] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, 2006.

[9] Schutlz and Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2003.

[10] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005.

[11] G. Wang, Y. Zhang, and L. Fei-Fei. Using dependent regions for object categorization in a generative framework. In *CVPR*, 2006.

[12] H. Zhang, A. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearset neighbor classification for visual category recognition. In *CVPR*, 2006.