SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition

Hao Zhang Alexander C. Berg Michael Maire Jitendra Malik
Computer Science Division, EECS Department
Univ. of California, Berkeley, CA 94720
{nhz,aberg,mmaire,malik}@eecs.berkeley.edu

Abstract

We consider visual category recognition in the framework of measuring similarities, or equivalently perceptual distances, to prototype examples of categories. This approach is quite flexible, and permits recognition based on color, texture, and particularly shape, in a homogeneous framework. While nearest neighbor classifiers are natural in this setting, they suffer from the problem of high variance (in bias-variance decomposition) in the case of limited sampling. Alternatively, one could use support vector machines but they involve time-consuming optimization and computation of pairwise distances.

We propose a hybrid of these two methods which deals naturally with the multiclass setting, has reasonable computational complexity both in training and at run time, and yields excellent results in practice. The basic idea is to find close neighbors to a query sample and train a local support vector machine that preserves the distance function on the collection of neighbors.

Our method can be applied to large, multiclass data sets for which it outperforms nearest neighbor and support vector machines, and remains efficient when the problem becomes intractable for support vector machines. A wide variety of distance functions can be used and our experiments show state-of-the-art performance on a number of benchmark data sets for shape and texture classification (MNIST, USPS, CUReT) and object recognition (Caltech-101). On Caltech-101 we achieved a correct classification rate of $59.05\%(\pm 0.56\%)$ *at 15 training images per class, and* $66.23\%(\pm 0.48\%)$ *at 30 training images.*

1. Introduction

While the field of visual category recognition has seen rapid progress in recent years, much remains to be done to reach human level performance. The best current approaches can deal with 100 or so categories, e.g. the CUReT dataset for materials, and the Caltech-101 dataset for objects; this is still a long way from the the estimate of 30,000 or so categories that humans can distinguish. Another significant feature of human visual recognition is that it can be trained with very few examples, cf. machine learning approaches to digits and faces currently require hundreds if not thousands of examples.

Our thesis is that scalability on these dimensions can be best achieved in the framework of measuring similarities, or equivalently, perceptual distances, to prototype examples of categories. The original motivation comes from studies of human perception by Rosch and collaborators [32] who argued that categories are not defined by lists of features, rather by similarity to prototypes. From a computer vision perspective, the most important aspect of this framework is that the emphasis on similarity, rather than on feature spaces, gives us a more flexible framework. For example, shape differences could be characterized by norms of transformations needed to deform one shape to another, without explicitly realizing a finite dimensional feature space.

In this framework, scaling to a large number of categories does not require adding new features [1], because the perceptual distance function need only be defined for similar enough objects. When the objects being compared are sufficiently different from each other, most human observers would simply assign "entirely different"(∞) to the distance measure, or, as D'Arcy Thompson quotes [37], *heterogena comparari non possunt*. Training with very few examples is made possible, because invariance to certain transformations or typical intra-class variation, can be built in to the perceptual distance function. Goldmeier's [13] study of the human notion of shape similarity, e.g. the privileging of structural changes, suggests several such characteristics.

For readers who may or may not be swayed by the philosophical arguments above, we also note the histori-

[1] Though one could argue that feature sharing keeps this problem manageable [39]

Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)
0-7695-2597-0/06 $20.00 © 2006 IEEE

cal evidence that for most well-studied visual recognition datasets, the humble nearest neighbor classifier with a well chosen distance function has outperformed other, considerably more sophisticated, approaches. Examples are tangent distance on the USPS zip code dataset (Simard, LeCun & Denker [35]), shape context based distance on the MNIST digit dataset (Belongie, Malik & Puzicha [1]), distances between histograms of textons on the CUReT data set (Leung and Malik [22], Varma and Zisserman [40]), and geometric blur based distances on Caltech-101 (Berg, Berg & Malik [3]).

We note some pleasant aspects of the the nearest neighbor (NN) classifier: (1) Many other techniques (such as decision trees and linear discriminants) require the explicit construction of a feature space, which for some distance functions is intractable (*e.g.* being high or infinite dimensional) (2) The NN classifier deals with the hugely multiclass nature of visual object recognition effortlessly. (3) From a theoretical point of view, it has the remarkable property that under very mild conditions, the error rate of a K-NN classifier tends to the Bayes optimal as the sample size tends to infinity [8].

Despite its benefits, there is room for improvements on the NN classifier. In the practical setting of a limited number of samples, the dense sampling required by the asymptotic guarantee is not present. In these cases, the NN classifier often suffers from the often observed "jig-jag" along the decision boundary. In other words, it suffers from high variation caused by finite sampling in terms of bias-variance decomposition. Various attempts have been made to remedy this situation, notably DANN [16], LFM-SVM [11], HKNN [41]. Among those, Hastie and Tibshirani [16] carries out a local linear discriminant analysis to deform the distance metric based on say 50 nearest neighbors. Domeniconi and Gunopulos [11] also deforms the metric by feature weighting, however the weights are inferred from training an SVM on the entire data set. In Vincent and Bengio [41], the collection of 15-70 nearest neighbors from each class is used to span a linear subspace for that class, and then classification is done based not on distance to prototypes but on distance to the linear subspaces (with the intuition that those linear subspaces in effect generate many "fantasy" training examples).

Instead of distorting the distance metric, we would like to bypass this cumbersome step and arrive at classification in one step. Here we propose to train a support vector machine(SVM) on the collection of nearest neighbors. This approach is well supported by ingredients in the practice of visual object recognition.

1. The carefully designed distance function, used by the NN classifier, can be transformed in a straightforward way to the kernel for the SVM, via the "kernel trick" formula: $K(x,y) = \langle x,y\rangle = \frac{1}{2}(\langle x,x\rangle + \langle y,y\rangle - \langle x-y,x-y\rangle) = \frac{1}{2}(d(x,0) + d(y,0) - d(x,y))$ where d is the distance function, and the location of the origin(0) does not affect SVM([33]). Various other ways of transforming a distance function into a kernel are possible, too [2].

2. SVMs operate on the kernel matrix without reference to the underlying feature space, bypassing the feature space operations of previous approaches (e.g. in DANN [16], feature vectors in R^n have to be defined and their covariances have to be computed before classifying a query, see Fig. 1.) In pratice, this translates into our capability to use a wide variety of distance functions whereas previous approaches were limited to L_2 distance.

3. In practice, training an SVM on the entire data set is slow and the extension of SVM to multiple classes is not as natural as NN. However, in the neighborhood of a small number of examples and a small number of classes, SVMs often perform better than other classification methods.

4. It is observed in psychophysics that human can perform coarse categorization quite fast: when presented with an image, human observers can answer coarse queries such as presence or absence of an animal in as little as 150ms, and of course can tell what animal it is given enough time [38]. This process of a coarse and quick categorization, followed by successive finer but slower discrimination, motivated our approach to model such process in the setting of machine learning. We use NN as an initial pruning stage and perform SVM on the smaller but more relevant set of examples that require careful discrimination.

We term our method "SVM-KNN" (where K signifies the method's dependence on choice of the number of neighbors).

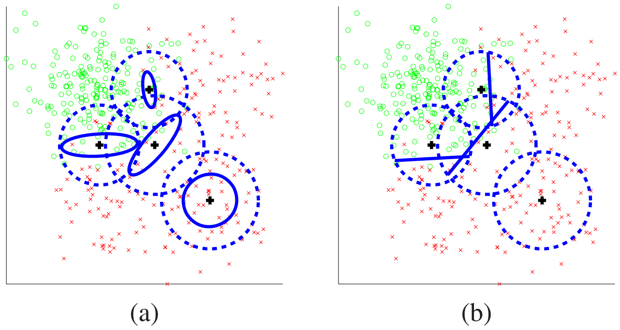

Figure 1. Difference between DANN and our method on a two class problem("o" vs "x"): (a) DANN deforms the metric based on 50 nearest neighbors (denoted by a dotted circle), on several query positions, then classifies using NN based on the new metric; (b) our method trains an SVM on the same 50 nearest neighbors(preserving the original distance metric), and directly obtains local decision boundary.

[2] For example, take $K(x,y)$ to be $\exp(-d(x,y)/\sigma^2)$, in a radial basis kernel fashion. However, we found no advantage of more complex transformation in our experiments, hence we stick with the simplest transformation so as to retain the intuitive interpretation.

Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)
0-7695-2597-0/06 $20.00 © 2006 IEEE

The philosophy of our work is similar to that of “Local Learning”, by Bottou and Vapnik [6], in which they pursued the same general idea by using K-NN followed by a linear classifier with ridge regularizer. However, by using only a L_2 distance, their work was not driven by the constraint to adapt to a complex distance function.

The rest of the paper is organized as follows: in section 2, we describe our method in detail and view it from different perspectives; section 3 introduces a number of effective distance functions, section 4 shows the performance of our method applied to those distance functions in various benchmark data sets; we conclude in section 5.

2. SVM-KNN

A naive version of the SVM-KNN is: for a query,

1. compute distances of the query to all training examples and pick the nearest K neighbors;

2. if the K neighbors have all the same labels, the query is labeled and exit; else, compute the pairwise distances between the K neighbors;

3. convert the distance matrix to a kernel matrix and apply multiclass SVM;

4. use the resulting classifier to label the query.

To implement multiclass SVM in step 3, three variants from the statistics and learning literature have been tried([21], [9], [31]) on small number of samples from our data sets. They produce roughly the same quality of classifiers and the DAGSVM([31]) is chosen for its better speed.

The naive version of SVM-KNN is slow mainly because it has to compute the distances of the query to *all* training examples. Here we again borrow the insight from psychophysics that humans can perform fast pruning of visual object categories. In our setting, this translates into the practice of computing a “crude” distance (*e.g.* L_2 distance) to prune the list of neighbors before the more costly “accurate” distance computation. The reason is simply that if the crude distance is big enough then it is almost certain that the accurate distance will not be small. This idea works well in the sense that the performance of the classifier is often unaffected whereas the computation is orders-of-magnitude faster. Earlier instances of this idea in computer vision can be found in Simard *et al.* [36] and Mori *et al.* [25]. We term this idea “shortlisting”.

An additional trick to speed up the algorithm is to cache the pairwise distance matrix in step 2. This follows from the observation that those training examples who participate in the SVM classification lie closely to the decision boundary and are likely to be invoked repeatedly during query time.

After the preceding ideas are incorporated, the steps of the SVM-KNN are: for a query,

1. Find a collection of K_{sl} neighbors using a crude distance function (*e.g.* L_2);

2. Compute the “accurate” distance function (*e.g.* tangent distance) on the K_{sl} samples and pick the K nearest neighbors;

3. Compute (or read from cache if possible) the pairwise “accurate” distance of the union of the K neighbors and the query;

4. Convert the pairwise distance matrix into a kernel matrix using the “kernel trick”;

5. Apply DAGSVM on the kernel matrix and label the query using the resulting classifier.

So far there are two perspectives to look at SVM-KNN: it can be viewed as an improvement over NN classifier, or it can be viewed as a model of the discriminative process plausible in biological vision. From a machine learning perspective, it can also be viewed as an continuum between NN and SVM: when K is small(*e.g.* $K = 5$), the algorithm behaves like a straightforward K-NN classifiers. To the other extreme, when $K = n$ our method reduces to an overall SVM.

Note, for a large data set, or when the distance function is costly to evaluate, the training of DAGSVM becomes intractable even with state-of-the-art techniques such as sequential minimal optimization(SMO) (Platt [30]) because it needs to evaluate $O(n^2)$ pairwise “accurate” distances. In contrast, SVM-KNN is still feasible as long as one can evaluate the “crude” distance for the nearest neighbor search and train the local SVM within reasonable time. A comparison in time complexity is summarized in Table 1.

	DAGSVM	SVM-KNN
Training	$O(C_{\text{accu}} n^2)$	none
Query	$O(C_{\text{accu}}\#\text{SV})$	$O(C_{\text{crude}} n + C_{\text{accu}}(K_{\text{sl}} + K^2))$

Table 1. Comparison of time complexity, where n is the number of training examples, #SV the number of support vectors, C_{accu} and C_{crude} the cost for computing accurate and crude distances, K_{sl} the length of the shortlist, and K the length of the list participating in SVM classification.

3. Shape and texture distances

In applying SVM-KNN, we focus our efforts on classifying based on the two major cues in visual object recognition: shape and texture. We introduce several well-performing distances functions as follows:

3.1. χ^2 distance for texture

Following Leung and Malik [22], an image of texture can be mapped to a histogram of “textons”, which captures the distribution of different types of texture elements. The distance is defined as the Pearson's χ^2 test statistic [5] between the two texton histograms.

3.2. Marginal distance for texture

From a statistical perspective, the χ^2 distance above for texture can be viewed as measuring the difference between two joint distributions of texture responses: a piece of texture is passed through a bank of filters, the joint distribution of responses are vector-quantized into textons, and the histogram of textons are compared. Levina *et al*. [23] found that the joint distribution can often be well distinguished from each other by simply looking at the difference in the marginals (namely, the histogram of each filter response). Therefore, another distance function for texture is to sum up the distances between response histograms from each filter. This is used in our experiments for real-world images that may contain too many types of textons to be reliably quantized.

3.3. Tangent distance

Defined on a pair of gray-scale images of digits, tangent distance [36] is defined as the smallest distance between two linear subspaces (in the pixel domain R^n where n is the number of pixels), derived from the images by including perturbations from small affine transformation of the spatial domain and change in the thickness of pen-stroke (forming a 7-dimensional linear space).

3.4. Shape context based distance

The basic idea of shape context [1] is as follows: The shape is represented by a point set, with a descriptor at a control point to capture the "landscape" around that point. Those descriptors are iteratively matched using a deformation model. And the distance is derived from the discrepancy left in the final matched shapes and a score that denotes how far the deformation is from an affine transformation.

3.5. Geometric blur based distance

A number of shape descriptors can be defined on a gray scale image, for instance the shape context descriptor on the edge map(*e.g*. [26]), or the SIFT descriptor([24]), or the geometric blur descriptor([4]). In our experiments, we focus on the geometric blur descriptor. Usually defined on an edge point, the geometric blur descriptor applies a spatially varying blur on the surrounding patch of edge responses. Points further from the center are blurred more to reflect their spatial uncertainty under deformation. After this blurring, the descriptors are normalized to have L_2 norm 1. They are used in two kinds of distances in section 4.4.

3.6. Kernelizing the distance

Asymmetry: (of shape context based distance and geometric blur based distance) We simply define a symmetric distance: $d(x, y) + d(y, x)$, because in practice the discrepancy $|d(x, y) - d(y, x)|$ is small.

Triangle Inequality: (of tangent distance, shape context based distance and geometric blur based distance) Namely, the inequality $d(x, y) + d(y, z) \geq d(x, z)$ does not hold at all times, which prevents the distance from translating into a positive-definite kernel. A number of solutions have been suggested for this issue [29]. Here, we compute the smallest eigenvalue of the kernel matrix and if it is negative, we add its absolute value to the diagonal of the kernel matrix. Intuitively, if we view the kernel matrix as a kind of "similarity measure", adding a positive constant to the diagonal means strengthening self-similarity, which should not affect the sense of expressed similarity among the examples.

4. Performance on benchmark data sets

4.1. MNIST

The MNIST data set of handwritten digits contains 60,000 examples for training and 10,000 for test: each set contains equal number of digits from two distinct populations: Census Bureau employees and high school students [20]. Each digit is a 28x28 image, except for shape context computation where each digit is resized to 70x70 image. Some example digits from the test set are shown in Fig. 2(a). A number of state-of-the-art algorithms perform under 1% error rate, among which a shape context based method performs at .67%.

Two distances are used in this experiment: L_2 and shape context distance. For shape context, since its error rate may be close to the Bayes optimal, we use only the first 10,000 training examples so as to leave room of improvement (on the 10,000 examples we perform a 10 fold cross validation). To rely purely on shape context and not on image intensities, we also drop the "appearance" term in [1].

A summary of results is in Table 2. Note that while L_2 distance is straightforward for our method, a number of workarounds were necessary for the shape context based distance. Still, in both cases the performance improves significantly.

	L_2	SC (limited training)
SVM-KNN	1.66 ($K = 80$)	1.67 ($\pm$0.49) ($K = 20$)
NN	2.87 ($K = 3$)	2.2 ($\pm$0.77) ($K = 1$)

Table 2. error rate on MNIST (in percent): the parameter K for each algorithm is selected according to best performance(in range of [1,10] for NN and [5, 10, .., 100] for SVM-KNN). In SVM-KNN, the parameter $K_{\text{sl}} \approx 10K$, larger K_{sl} doesn't improve the empirical results.

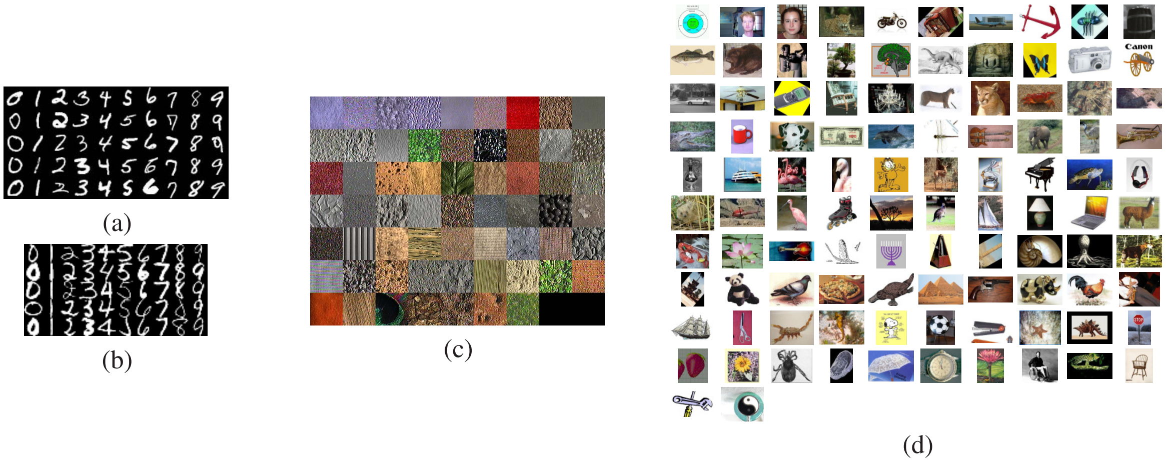

Figure 2. Data sets: (a) MNIST (b) USPS (c) CUReT (d) Caltech-101

4.2. USPS

The USPS data set contains 9298 handwritten digits (7291 for training, 2007 for testing), collected from mail envelopes in Buffalo [19]. Each digit is a 16x16 image. A collection of random test samples is shown in Fig. 2(b). It is known that the USPS test set is rather difficult: the human error rate is 2.5% [7].

We try two types of distances: L_2 and tangent distance. (Shape context is not tried because the image is too small to contain enough details for estimating deformation). For tangent distance, each image is smoothed with a Gaussian kernel of width $\sigma = 0.75$ to obtain more reliable tangents.

	L_2	tangent distance
SVM-KNN	4.285 ($K = 10$)	**2.59** ($K = 8$)
NN	5.53 ($K = 3$)	2.89 ($K = 1$)
DAGSVM	4.4 (Platt *et al.* [31])	intractable
HKNN	3.93 (Vincent *et al.* [41])	N/A

Table 3. error rate on USPS (in percent): the parameter K for SVM-KNN and NN is chosen according to best performance, respectively.

Table 3 shows that in the L_2 case, the error rates of SVM-KNN and DAGSVM are similar. However, SVM-KNN is much faster to train because each SVM only involves a local neighborhood of 10 samples, and the number of classes rarely exceeds 4 within the neighborhood. (In our experiments, the cost of training 10 examples from 4 classes is much smaller than the cost of the usual nearest neighbor search.) In contrast, DAGSVM involves training a SVM on all 45(=10x9/2) pairs of different classes, and computation of pairwise distances on all training examples. With the more costly tangent distance function, DAGSVM becomes intractable to train in our experiment, whereas the optimal SVM-KNN (where $K = 8$) is almost as fast as the usual NN classifier because the additional cost of training an SVM on 8 examples is negligible. This reflects the comparison of asymptotic complexity in section 2.

Also in the L_2 case, another adaptive nearest neighbor technique, HKNN(Vincent *et al*. [41]), performs quite well. Unfortunately, it operates in the input space and therefore cannot be extended to a distance function other than L_2.

In the tangent distance case, it is quite remarkable that SVM-KNN can improve on the performance of NN with very small additional cost, even though the latter is performing very well already (in comparison to human performance). We are therefore encouraged to think that SVM-KNN is an ideal classification method when the proper "invariance" structure of the underlying data set is captured in the distance function.

4.3. CUReT

CUReT(Dana *et al.* [10]) contains images of 61 real world textures(*e.g*. leather, rabbit fur, sponge, see Fig. 2(c)) photographed under varying illumination and viewing angle. Following [40], a collection of 92 images (where the viewing angle is not too oblique) are picked from each category, among which half are randomly selected as training and the rest as test.

From [40], we take the variant of the texton method that achieves the best performance on CUReT and substitute the last step of NN classifier with our method:

In terms of error rate, as in the case of USPS, SVM-KNN has a slight advantage over DAGSVM , both of which are significantly better than the state-of-the-art performance reported in [40]. However, DAGSVM (equivalently, SVM-KNN when $K = n$) was very slow: a total of 61x60/2=1830 pairwise classifiers to train (15130 CPU secs), whereas SVM-KNN is faster and offers a trade-off between time and

	χ^2
SVM-KNN	**1.73** (± 0.24) ($K = 70$)
NN	2.53 (± 0.28) ($K = 3$) [40]
DAGSVM	1.75 (± 0.25)

Table 4. error rate on CUReT (in percent): error rate and std dev obtained from 5 times 2 fold cross validation. The parameter K for SVM-KNN and NN is chosen according to best performance, respectively.

performance (Fig. 3).

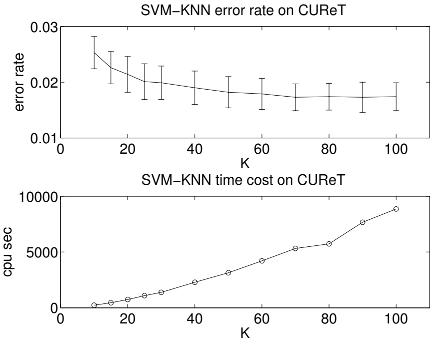

Figure 3. Trade-off between speed and accuracy of SVM-KNN in the case of texture classification

4.4. Caltech-101

The Caltech-101 data set (collected by L. Fei-Fei *et al.* [12]) consists of images from 101 object categories and an additional background class, making the total number of classes 102. The significant variation in color, pose and lighting makes this data set quite challenging. A number of previously published papers have reported results on this data set, e.g Berg *et al.* [3], Grauman and Darrell [14], and Holub *et al.* [17]. Berg *et al.* [3] constructed a correspondence procedure for matching geometric blur descriptors and use it to define a distance function between two images, the resulting distance is used in a NN classifier. In Grauman and Darrell[14], a set of local image features are matched in an efficient way using a pyramid of histograms. The resulting matching score forms a kernel that is used in an SVM classifier. In Holub *et al.* [17], Fisher scores on each image are obtained by a generative model of the object categories. An SVM is trained on a Fisher kernel based on these scores. In these proceedings, a number of groups [18, 27, 15, 42], in addition to our paper, have demonstrated results on this dataset using a common methodology.

We present two algorithms on this data. The difference lies in the choice of the distance function.

A. Algorithm A

Unlike the previous data sets, in this setting we have both shape and texture. For the shape part, geometric blur features sampled at a subset of edge points (Section 3.5, details in [3]) are used. For the texture part, the marginal distance for texture (see section 3.2) is used, where the filter bank is Leung-Malik [22]. The distance function is defined as:

$$\begin{aligned} D^A(I_L \to I_R) &= \frac{1}{m}\sum_{i=1}^{m} \min_{j=1..n} \|F_i^L - F_j^R\|^2 \\ D^A(I_L, I_R) &= D^A(I_L \to I_R) + D^A(I_R \to I_L) \qquad (1) \\ &\quad + \lambda \sum_{k=1}^{\mathrm{nfilt}} \|h_k^L - h_k^R\|_{\mathrm{L1}} \end{aligned}$$

Here $D^A(I_L, I_R)$ is the distance between left and right images. The computation is based on geometric blur features F_i^L (denoting i'th feature in the left image, respectively for F_j^R) and texture histograms h_k^L (denoting the histogram of the k's filter output on the left image, respectively for h_k^R). Note that the texture histograms are normalized to sum to 1. Rather large scale geometric blur descriptors are used, (radius ∼70 pixels), and $\lambda = \frac{1}{8}$ is set based on experiments with a small collection of images from about 10 classes.

To stay as close to the paradigm of the previous work on this dataset using geometric blur features, we followed the methodology of Berg *et al.* [3], randomly picking 30 images from each class and splitting them into 15 for training and 15 for test. We also reverse the role of training and test. The correctness rate is the average. Table 5 shows the results which can be compared to [3] (45%) and [2] (52%), all corresponding to 15 training images per class. Compared to the baseline classifiers (NN and SVM), SVM-KNN has a statistically significant gain.

	Algo. A
SVM-KNN	**59.08**(± 0.37) ($K = 300$)
NN	40.98 (± 0.47) ($K = 1$)
DAGSVM	56.40(± 0.36)

Table 5. **Correctness rate** (=1-error rate) of Algorithm A with 15 training images per class (in percentage, and std dev.). Parameter K for SVM-NN and for NN are chosen respectively according to best performance.

B. Algorithm B

In our previous work on Caltech-101 (Berg, Berg and Malik [3]), we sought to find shape correspondence in a deformable template paradigm. However, due to the special character of the Caltech-101 data set (objects are often in the center of image, and the scale does not vary much), a crude way of incorporating spatial correspondence is to add a first-order geometric distortion term when geometric blur features are being compared, where position is measured from center of image (cf. a more general approach based on second order geometric distortion, comparing pairs of points in [3]).

In this case, the overall distance function is

$$D^B(I_L \to I_R) = \frac{1}{m}\sum_{i=1}^{m} \min_{j=1..n}\left[\|F_i^L - F_j^R\|^2 + \frac{\lambda}{r_0}\|r_i^L - r_j^R\|\right]$$
$$D^B(I_L, I_R) = D^B(I_L \to I_R) + D^B(I_R \to I_L) \tag{2}$$

and r_i^L denotes the pixel coordinates of the i'th geometric blur feature on the left image, w.r.t. the image center (respectively for r_j^R). $r_0 = 270$ is the average image size. We used a medium scale of geometric blur(radius ∼42 pixels), and $\lambda = \frac{1}{4}$.

Algorithm B is tested with the benchmark methodology of Grauman and Darrell [15], where a number (say 15) of images are taken from each class uniformly at random as the training image, and the rest of the data set is used as test set. The "mean recognition rate per class" is used so that more populous (and easier) classes are not favored. This process is repeated 10 times and the average correctness rate is reported. Our experiments use the DAGSVM classifier. (We have yet to run SVM-KNN in this setting but the performance of SVM-KNN can only be better because it includes DAGSVM as a special case for $K = n$.). [3]

The performance for Algorithm B is plotted in Fig. 4, alongside other current techniques (published or in press), in the same format as that of Grauman and Darrell [15]. It is noteworthy that Algorithm B as well as the techniques of Wang et al and Lazebnik et al, have attained correctness rates in the neighborhood of 60%, a significant improvement over the first reported result of 17% only a couple of years ago. Numbers, for the 15 and 30 training images cases, can be found in table 6. The confusion matrix for 15 training images is in Fig. 5.

Ommer and Buhmann [28] used a different evaluation methodology; for which our correctness rate is 63%, compared to 57.8% of [28]

# train	Algo. B	[18]	[2]	[27]	[15]	[42]
15	59.05(±0.56)	56.4	52	51	49.52	44
30	66.23(±0.48)	64.6(±0.8)	N/A	56	58.23	63

Table 6. **Correctness rate** with 15 or 30 training images per class on Caltech-101 (in percentage, and std dev. where available)

5. Conclusion

In this paper we proposed a hybrid of SVM and NN, which deals naturally with multiclass problems. We show excellent results using a variety of distance functions on several benchmark data sets.

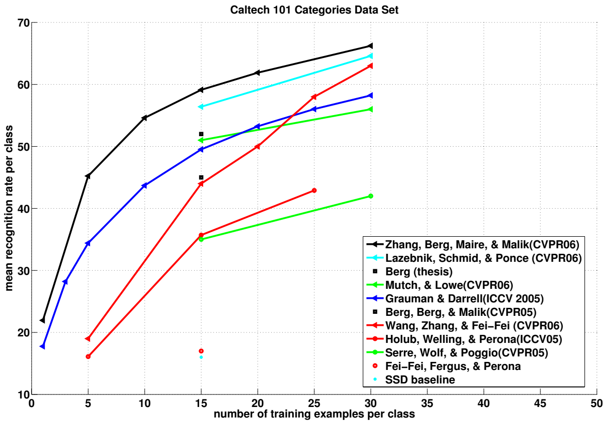

Figure 4. **Correctness rate** of Algorithm B (plotted the same format as in [42] and [15]), **best viewed in color**, Results from this work and others: Lazebnik, Schmid & Ponce [18], Berg [2], Mutch & Lowe [27], Grauman & Darrell [15], Berg, Berg & Malik [3], Wang, Zhang & Fei-Fei [42], Holub, Welling & Perona [17], Serre, Wolf & Poggio [34], and Fei-Fei, Fergus & Perona [12].

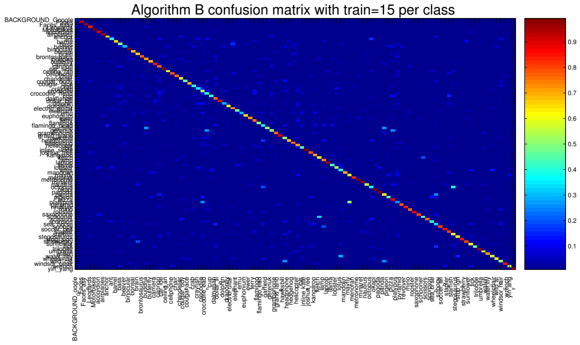

Figure 5. Confusion matrix of Algorithm B with 15 training images per class, where the rows denote true label and the columns denote reported label. The most confused pairs are (schooner,ketch), (waterlily,lotus), (platypus,mayfly), (octopus, starfish).

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[2] A. C. Berg. *Shape Matching and Object Recognition*. PhD thesis, Computer Science Division, University of California, Berkeley, 2005.

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

[4] A. C. Berg and J. Malik. Geometric blur for template matching. In *CVPR*, pages 607–614, 2001.

[5] P. Bickel and K. Doksum. *Mathematical statistics*, volume 1. Prentice Hall, 2nd edition, 2001.

[3] In our experiments, we have found virtually no difference under this evaluation methodology vs that of Berg *et al.* [3].

[6] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

[7] Bromley and Säckinger. Neural-network and K-nearest-neighbor classifiers. Technical Report 11359-910819-16TM, AT&T, 1991.

[8] T. M. Cover. Estimation by the nearest neighbor rule. *IEEE Trans. on Information Theory*, 14(1):50–55, 1968.

[9] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

[10] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 18(1):1–34, 1999.

[11] C. Domeniconi and D. Gunopulos. Adaptive nearest neighbor classification using support vector machines. In *NIPS*, pages 665–672, 2001.

[12] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*, 2004.

[13] E. Goldmeier. Similarity in visually perceived forms. *Psychological Issues*, 8(1):1–134, 1972.

[14] K. Grauman and T. Darrell. Discriminative classification with sets of image features. In *ICCV*, 2005.

[15] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features (version 2). Technical Report CSAIL-TR-2006-020, MIT, 2006.

[16] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(6):607–616, 1996.

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

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

[19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Winter 1989.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[21] Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99:67 – 81, 2004.

[22] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 43(1):29–44, 2001.

[23] L. Levina. *Statistical Issues in Texture Analysis*. PhD thesis, Department of Statistics, University of California, Berkeley, 2002.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[25] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *CVPR*, volume 1, pages 723–730, 2001.

[26] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *European Conference on Computer Vision LNCS 2352*, volume 3, pages 666–680, 2002.

[27] J. Mutch and D. Lowe. Multiclass object recognition using sparse, localized features. In *CVPR*, 2006.

[28] B. Ommer and J. M. Buhmann. Learning compositional categorization models. In *ECCV*, 2006.

[29] E. Pekalska, P. Paclik, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *J. Mach. Learn. Res.*, 2:175–211, 2002.

[30] J. C. Platt. Using analytic qp and sparseness to speed training of support vector machines. In *NIPS*, pages 557–563, Cambridge, MA, USA, 1999. MIT Press.

[31] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *NIPS*, pages 547–553, 1999.

[32] E. Rosch. Natural categories. *Cognitive Psychology*, 4:328–350, 1973.

[33] B. Schölkopf. The kernel trick for distances. In *NIPS*, pages 301–307, 2000.

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

[35] P. Simard, Y. LeCun, and J. S. Denker. Efficient pattern recognition using a new transformation distance. In *NIPS*, pages 50–58, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[36] P. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*, pages 239–274, London, UK, 1998. Springer-Verlag.

[37] D. W. Thompson. *On Growth and Form*. Cambridge University Press, 1917.

[38] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, June 1996.

[39] A. B. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR*, pages 762–769, 2004.

[40] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1–2):61–81, Apr. 2005.

[41] P. Vincent and Y. Bengio. K-local hyperplane and convex distance nearest neighbor algorithms. In *NIPS*, pages 985–992, 2001.

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