

# Comparing Data-Dependent and Data-Independent Embeddings for Classification and Ranking of Internet Images

Yunchao Gong and Svetlana Lazebnik  
Department of Computer Science, UNC Chapel Hill, NC, 27599.  
{yunchao, lazebnik}@cs.unc.edu

## Abstract

*This paper presents a comparative evaluation of feature embeddings for classification and ranking in large-scale Internet image datasets. We follow a popular framework for scalable visual learning, in which the data is first transformed by a nonlinear embedding and then an efficient linear classifier is trained in the resulting space. Our study includes data-dependent embeddings inspired by the semi-supervised learning literature, and data-independent ones based on approximating specific kernels (such as the Gaussian kernel for GIST features and the histogram intersection kernel for bags of words). Perhaps surprisingly, we find that data-dependent embeddings, despite being computed from large amounts of unlabeled data, do not have any advantage over data-independent ones in the regime of scarce labeled data. On the other hand, we find that several data-dependent embeddings are competitive with popular data-independent choices for large-scale classification.*

## 1. Introduction

The recognition community is starting to grapple with Internet datasets containing millions of images and thousands of categories. For example, the ImageNet dataset [5] contains over 10 million images and 10,000 categories with complete ground truth information. To fully take advantage of such data, it is necessary to develop efficient large-scale supervised learning methods. Other datasets, like 80 Million Tiny Images [21], have very sparse ground truth. They pose a different challenge of how to learn good classifiers using very few labeled images and a potentially very large number of unlabeled ones.

State-of-the-art representations for image classification are based on combining powerful features with nonlinear kernels, e.g., GIST features [14] with Gaussian kernels, or bag-of-words (BoW) features with histogram intersection kernels [13]. Unfortunately, the computational complexity of training standard nonlinear support vector machines

(SVMs) can be worse than quadratic in the size of the training set. To make large-scale learning practical, many researchers are beginning to favor a strategy where the data is first transformed by a nonlinear mapping induced by a particular kernel and then an efficient linear classifier is trained in the resulting space [4, 13, 15, 16].

This paper presents a comparative evaluation of scalable nonlinear embedding methods for Internet datasets. We consider two major classes of embeddings: data-dependent and data-independent ones. Data-dependent methods originate from the manifold and semi-supervised learning (SSL) literature [2], and they are computed from unlabeled training samples. A standard manifold learning formulation [1] computes the Laplacian matrix encoding the pairwise similarities between the unlabeled data samples and then performs its eigendecomposition to find a basis for functions that are smooth with respect to the data distribution. Because exact Laplacian eigenvector methods scale cubically in the size of the dataset, we consider instead efficient approximations based on the Nyström method [8, 23] and on separability assumptions [7].

As for data-independent embeddings, the idea behind them is to transform features into a space in which the dot product approximates a specific kernel in the original feature space, so that a linear classifier in the transformed space could approximate a nonlinear one in the original space. Our study includes explicit embeddings for the Gaussian kernel [16], which is appropriate for GIST features, and for histogram intersection and Bhattacharyya kernels [13, 15], which are appropriate for BoW features.

To our knowledge, our study is the only one that encompasses both ends of the supervision spectrum: from very few labeled images (the scenario that SSL methods are aimed at) to thousands of labeled images per class. Perhaps surprisingly, we show that for multi-class classification with very little training data, data-dependent embeddings offer no discernible advantage over data-independent ones. We conjecture this is because Internet image data violates the “cluster assumption” of SSL, according to which points from the same class form a high-density cluster separated

from the other classes by low-density “valleys” [2]. On the other hand, for classification tasks with large amounts of labeled data, we find several data-dependent embeddings that can offer promising alternatives to current popular choices for GIST and BoW features. Through our evaluation on ranking tasks, we also find that data-dependent methods appear to be somewhat better at correctly classifying images closer to the positive training examples, as opposed to the ones closer to the decision boundaries.

## 2. Relationship to Previous Work

Perronnin et al. [15] study explicit embeddings for large-scale supervised learning with BoW features. Our evaluation is broader: we also look at learning with few labeled points, and consider GIST features as well as BoW. Our findings confirm the effectiveness of the SQRT and additive KPCA (AddKPCA) embeddings suggested in [15]. In addition, we compare AddKPCA to the intersection kernel embedding of [13] and show that AddKPCA gives comparable performance at a fraction of the memory cost.

Deng et al. [4] train 10,000 classifiers on 4.5 million ImageNet images using LIBLINEAR SVM [6] with the embedding of [13] applied to BoW and spatial pyramids [12]. They report that training in this setup takes over a CPU-year. Our evaluation demonstrates alternative embedding/classifier setups that can potentially save memory and/or computation.

Recently there has been a lot of interest in improving the scalability of manifold embedding methods through the use of low-rank approximations such as the Nyström method (see, e.g., [3, 11, 20]). However, in much of this work, evaluation is performed on data known to be “friendly” to SSL, such as digits or faces. Even when the methods are evaluated on more diverse datasets, baseline comparisons with data-independent embeddings are usually lacking. We compare the two types of embeddings side by side for classification and ranking of Internet images and critically inquire when the use of unlabeled data is justified.

Fergus et al. [7] propose an approximate manifold embedding method specifically aimed at Internet photo collections and evaluate it on re-ranking tasks. We evaluate this method using different experimental protocols against several other state-of-the-art embeddings, and find a simple variant that can make it more effective for GIST features.

## 3. Components of the Study

### 3.1. Image Features

The first type of feature used in our study is GIST [14], which is a global image descriptor based on convolving the image with a Gabor filter bank at several scales and orientations and aggregating the filter responses over a coarse spatial grid. The GIST feature is typically used in conjunction

with a Gaussian kernel. The second type of feature is a bag of words (BoW), or a histogram of frequencies of visual words obtained by quantizing local image descriptors to a visual vocabulary. BoW and its extensions, such as spatial pyramids [12], are typically used with kernels for comparing probability distributions. We consider two such kernels: histogram intersection [13] and Bhattacharyya [15]. Details of feature extraction will be given in Section 4.1.

## 3.2. Nonlinear Embedding Methods

### 3.2.1 Data-dependent Embeddings

The data-dependent methods considered in this study are motivated by the manifold learning formulation in which the goal is to find a mapping that is smooth with respect to the data distribution, i.e., that maps nearby points in the original space to nearby points in the embedded space. We can think of each coordinate of the embedding as a scalar function defined on the data. For a dataset of  $n$  points, let  $\mathbf{f} \in \mathbb{R}^{n \times 1}$  be the vector of corresponding function values. The smoothness of  $\mathbf{f}$  is measured by  $\sum_{i,j} K_{ij}(f_i - f_j)^2 = \mathbf{f}^T L \mathbf{f}$ , where  $K$  is the  $n \times n$  kernel matrix of pairwise similarities between the original data points and  $L$  is the *graph Laplacian*. We can write  $L = D - K$ , where  $D$  is a diagonal matrix in which each element is sum of rows of  $K$ . By taking the eigenvectors of  $L$  associated with  $d$  smallest (non-zero) eigenvalues, we obtain a  $d$ -dimensional basis spanning the smoothest functions over the data [1]. This gives us the desired embedding over which we can then learn a linear classifier using the available labeled points.

Note that most implementations of spectral methods use the *normalized Laplacian*  $I - D^{-1/2} K D^{-1/2}$ . Consistent with general practice, we use this Laplacian with the Nyström method (see below). Other formulations, like [7], use the *random walk Laplacian*  $I - D^{-1} K$  [22]. To stay consistent with [7], we use this variant for our separable approximations (PCALap and RPLap), which yields the generalized eigenvalue problem in Table 1.

**Nyström approximation.** Since eigendecomposition of the  $n \times n$  Laplacian scales as  $O(n^3)$ , practical methods tend to rely on approximations. The Nyström method [8, 23] samples a subset of *landmark points* from the dataset and approximates the eigenvectors of the whole matrix  $L$  using the eigenvectors of the smaller Laplacian matrix of the landmark points.<sup>1</sup> To embed new test points, Nyström takes a linear combination of the embedded landmark coordinates weighted by their similarity (as defined by the kernel function) to the test point (for details see [8, 23]).

**Separable approximations.** Recently, Fergus et al. [7] have introduced a new method<sup>2</sup> for approximating Laplacian eigenvectors making the assumption that the distribu-

<sup>1</sup>We extended the code from <http://alumni.cs.ucsb.edu/~wychen/sc.html>

<sup>2</sup>Code from [http://cs.nyu.edu/~fergus/code/ssl\\_eigenfunction.zip](http://cs.nyu.edu/~fergus/code/ssl_eigenfunction.zip)

Table 1. Embeddings based on the separability assumption.

---

Given: a set of  $n$   $D$ -dimensional data points,  $X \in R^{n \times D}$ .

---

1. Linearly transform the data:
  - **PCALap**:  $S = XR$  in which  $R$  is PCA directions
  - **RPLap**:  $S = XR$  in which  $R$  is random projections
2. **For each dimension  $s^m$  of  $S$ :**
  - Partition the range of  $s^m$  into  $q$  bins with centers  $b_i$  ( $q = 50$  in the implementation)
  - Construct a kernel matrix  $K$  where  $K_{ij} = k(b_i, b_j)$ :
    - **PCALap**:  $k$  is Gaussian kernel
    - **RPLap**:  $k$  is Gaussian kernel
  - Estimate single dimensional distribution  $P$
  - solve  $(D - PKP)\mathbf{f} = \lambda PD\mathbf{f}$ ,  $D = \text{diag}(\sum_j K_{ij})$

**End For**

5. Choose  $d$  eigenvectors  $\mathbf{f}$  with the smallest eigenvalues to form the embedding
6. Linearly interpolate the eigenvectors along each dimension to compute the embedding for the whole data set

---

tion of the data is separable, i.e., it can be decomposed into a product of the marginal distributions of the individual dimensions. In this case, the eigenvectors of individual dimensions become the eigenvectors of the entire dataset. They use PCA to find the separable directions, estimate the single-dimensional eigenvectors, and embed new points by linear interpolation. The steps of this method, referred to as **PCALap**, are summarized in Table 1.

In the course of analyzing the performance the PCALap method, we have found that using Gaussian random projections instead of PCA results in very good performance for GIST features. This choice lacks theoretical justification, but it parallels the use of random projections in the random Fourier features mapping for Gaussian kernels [16]. This new variant, referred to as **RPLap**, is also summarized in Table 1. To keep our implementation of PCALap consistent with [7], we use histograms to estimate the marginal distributions  $P$  along each dimension and incorporate this information into the eigenvalue problem. However, for RPLap, we simply use uniform marginals ( $P$  is the identity), which works just as well.

**AddKPCA.** Perronnin et al. [15] have proposed a mapping for BoW based on kernel PCA [18] applied along each dimension of the original data. This method follows the outline of Table 1, with a few differences. Instead of computing the kernel matrix  $K$  over a set of uniform bin centers in each dimension, it uses the coordinates of random landmark points as in the Nyström method. Then it performs the eigendecomposition of  $K$  instead of the Laplacian. After processing all the dimensions, it keeps the  $d$  single-dimensional eigenvectors associated with the *largest* eigenvalues. To embed a new test point, it applies the Nyström interpolation method along each dimension separately.

### 3.2.2 Data-Independent Embeddings

The idea of kernel-specific data-independent embeddings is to transform the original feature vectors into a high-

dimensional space so that the dot product of two embedded data points approximates (or exactly equals) the kernel value computed on the original feature points.

**Random Fourier Feature (RFF)** [16] approximates the Gaussian kernel. For a feature point  $\mathbf{x}$ , each coordinate of the embedding is given by  $\sqrt{2} \cos(\mathbf{w}^T \mathbf{x} + b)$ , where  $\mathbf{w}$  and  $b$  are random parameters. To approximate the kernel  $k(\mathbf{x}, \mathbf{z}) = e^{-\gamma \|\mathbf{x} - \mathbf{z}\|^2 / 2}$ , we draw the random vector  $\mathbf{w}$  from  $\text{Normal}(0, \gamma I)$  and  $b$  from  $\text{Unif}[0, 2\pi]$ .

**Histogram Intersection Kernel Embedding (HIKE)** [13] works by discretizing the values of histogram bins into  $B$  levels and representing them in “unary” notation (for details, see [13], where this embedding is referred to as  $\phi_2$ ).<sup>3</sup> HIKE raises the dimensionality of the features by a factor of  $B$ , leading to high memory usage.

**Square Root Embedding (SQRT)** [15] is an extremely simple mapping for BoW feature vectors. It is based on the Bhattacharyya kernel for comparing distributions:  $k(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^d \sqrt{x_j z_j}$ . The corresponding exact mapping is given by square-rooting  $\mathbf{x}$  term by term.

### 3.2.3 Discussion

When might we expect data-dependent embeddings to work better than data-independent ones? Intuitively, when very few labeled training points are available, unlabeled points may be expected to regularize the embedding by incorporating additional information about the data distribution. In the SSL literature, a popular “cluster assumption” [2] gives the conditions under which unlabeled data can improve classification performance: namely, if points belonging to the same class lie in high-density areas and points from different classes are separated by low-density “valleys,” then unlabeled data can help to locate the boundaries between classes more precisely. The cluster assumption holds for relatively “clean” image datasets, such as digits, to which SSL methods have traditionally been applied. As an illustration, Figure 1(a) shows the Nyström embedding of digits 1-4 from the USPS dataset.<sup>4</sup> In this plot, the four different digits form reasonably well-separated clusters. As shown in Figure 1(b), when there are fewer than 10 labeled points per class, learning a classifier in the Nyström-embedded space results in better performance than either in the original feature space or with the data-independent RFF embedding. By contrast, Figure 1(c) shows the Nyström embedding for three ImageNet classes. These classes are much more mixed, and there are no obvious “valleys” separating them. One has reason, therefore, to doubt whether the cluster assumption holds for challenging and diverse Internet photo collections. In the rest of our evaluation, we will seek

<sup>3</sup>Code from <http://www.cs.berkeley.edu/~smaji/projects/add-models/>

<sup>4</sup>Data from <http://www.zjucadcg.cn/dengcai/Data/MLData.html>

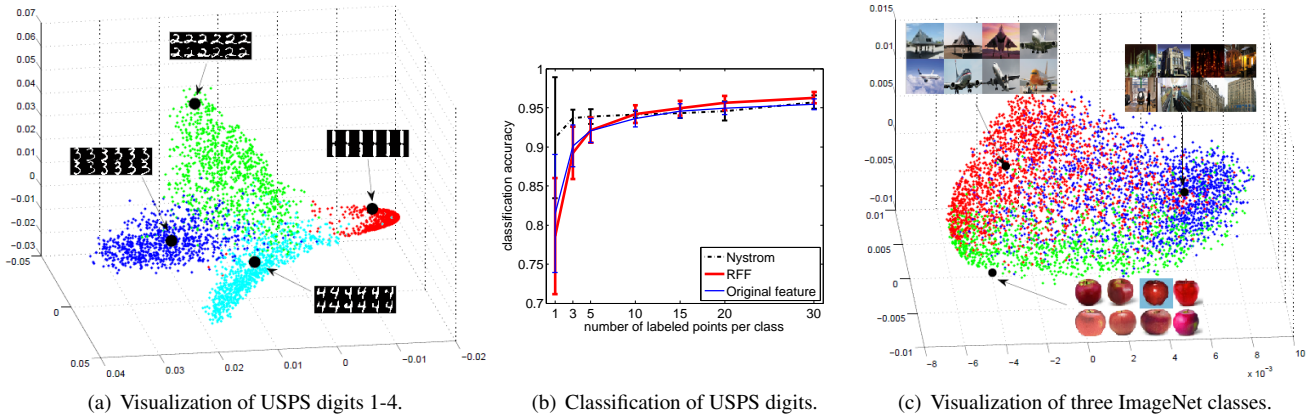


Figure 1. (a) Visualization of the top three dimensions of the Nyström embedding for USPS digits 1-4. (b) Classification performance for USPS digits for three different embeddings and different numbers of training points per class. For Nyström, the number of eigenvectors used for different amounts of training data is [4, 5, 5, 8, 10, 30, 40]. Results are averaged over 100 random trials, and error bars show the standard deviations. (c) Top three dimensions of the Nyström embedding for GIST descriptors of three ImageNet classes: airplane (red), apple (green), and construction (blue). Best viewed in color.

to understand whether, and when, data-dependent embeddings offer a performance advantage on such datasets.

### 3.3. Efficient Linear Classifiers

After computing a nonlinear embedding of the data with one of the methods described in Section 3.2, the next step is to use the available labeled training data to learn a linear classifier in the embedded space.

One of the most efficient linear SVM packages is LIBLINEAR [6]. However, for very large-scale experiments, it can still be very slow [4]. In this evaluation, we use ridge regression [9], which is extremely simple, runs much faster than LIBLINEAR SVM, and has been reported to work well in the machine learning literature [16, 17].

Given  $\tilde{X} \in \mathbb{R}^{l \times d}$ , the matrix of  $l$  training vectors embedded into a  $d$ -dimensional space, and  $Y \in \{0, 1\}^{l \times c}$ , the matrix of label information where  $Y_{ij} = 1$  if the  $i$ th data point has label  $j$  and 0 otherwise, ridge regression solves for the matrix of classifier parameters  $W \in \mathbb{R}^{d \times c}$  as  $W = (\tilde{X}^T \tilde{X} + \rho I)^{-1} \tilde{X}^T Y$ , where  $\rho$  is a regularization parameter (found through validation). In order to obtain the classifier parameters  $W$  for all  $c$  classes, we only need to invert a  $d \times d$  matrix once. Unlike one-vs.-all SVMs, there is no need to train  $c$  separate classifiers.

For both the SVM and ridge regression models, we do not use a bias parameter (i.e., we fit a decision hyperplane that passes through the origin), but prior to the training, we translate the origin to the mean of all points (including the unlabeled ones). Note that centering makes little difference for most setups we have tried, except for ranking with extremely unbalanced labeled data (see Section 4.4 for discussion). Figure 2 shows a comparison of classification accuracy and training time for ridge regression and LIBLINEAR SVM. The accuracy of the two classifiers is almost identical,

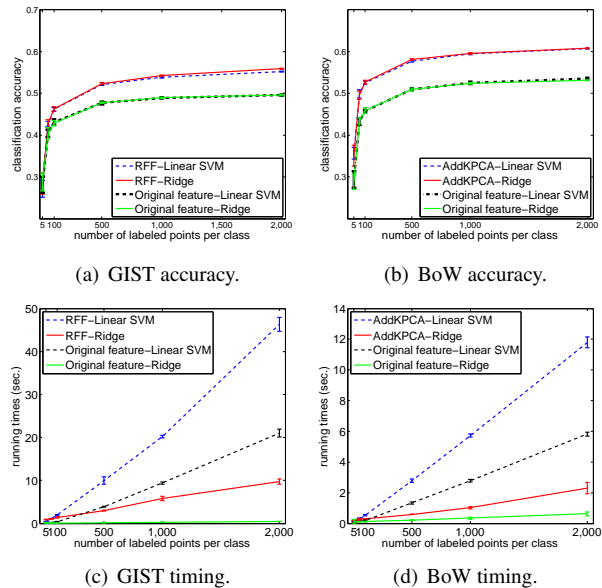


Figure 2. Comparison of ridge regression and linear SVM. Results are averaged over 10 random training/test splits.

cal, and ridge regression is 5-10 times faster than SVM.

## 4. Experimental Evaluation

### 4.1. Implementation Details

**Datasets.** Our first dataset, **CIFAR10** [10], is a manually cleaned subset of 80 Million Tiny Images [21]. There are 10 classes with 6,000 images each, for a total of 60,000 images. For our second dataset, **ImageNet10**, we have selected 10 classes from ImageNet [5], for a total of 331,390 images. The selected classes together with the number of instances are: airplane (25,551), apple (13,993), building (21,446), cat (40,246), fieldgame (32,784), floor-

cover (17,163), legume (25,920), optical (26,996), orchid (96,239), and palm (31,052). For each of these classes, we have downloaded all the images of the corresponding synset and its children. In addition, for experiments that are particularly computation-intensive (such as the ones in Section 4.4) we have constructed a smaller random subset of 20,000 images, referred to as **ImageNet20K**.<sup>5</sup>

**Feature extraction.** For all datasets, we compute grayscale GIST features at 8 orientations and 4 different scales, resulting in 320-dimensional descriptors. BoW features are computed only on the higher-resolution ImageNet datasets. For this, we extract SIFT descriptors over a regular grid of  $16 \times 16$  pixel patches with a spacing of 8 pixels and quantize them to a dictionary of size 400.

**Embeddings.** For the **Nyström** method, the number of eigenvectors needed for optimal performance typically increases with the amount of labeled training data available. For small amounts of labeled data (experiments of Sections 4.2 and 4.4), we have found that using 50 eigenvectors works well. For large amounts of labeled data (Section 4.3), we use 500 eigenvectors. As for the number of landmarks, we use 1,000 randomly sampled points; we have found that increasing the number to 2,000 or 3,000 produced almost no improvement. To apply the Nyström method to GIST features, we compute a Gaussian kernel over the landmark points by setting the bandwidth to the average distance to the 50th nearest neighbor in the dataset. For BoW features, we compute the kernel matrix using the histogram intersection kernel, which does not have any parameters.

**PCALap** behaves similarly to Nyström, in that more eigenvectors are needed to get good performance on larger amounts of labeled data. Accordingly, for the small-scale experiments of Sections 4.2 and 4.4, we set the number of PCA directions to 50, and for the large-scale experiments of Section 4.3, we use all the PCA directions. The number of eigenvectors is set to twice the number of projected directions in each case. To make our PCALap implementation consistent with [7], we alter the regression slightly to regularize the solution by a diagonal matrix of eigenvalues instead of a scalar multiple of the identity (this works 1-2% better for PCALap than standard ridge regression, and has no effect on the other methods). For the single-dimensional Gaussian kernel used with GIST features, we set the bandwidth to 0.2 as in [7].

**RPLap** behaves differently from Nyström and PCALap: its accuracy tends to improve the more random projections one uses, regardless of the amount of labeled data. We use 1,000 projections. In each projected direction, we estimate the Gaussian kernel bandwidth using the method of [19]. As with PCALap, the number of eigenvectors is two per dimension, for a 2,000-dimensional embedding.

<sup>5</sup>Our experimental data, together with MATLAB code, is available at <http://www.unc.edu/~yunchao/ssl.htm>.

Table 2. Approximate timing and memory usage for computing different embeddings on the ImageNet10 dataset on a workstation with a 2-core Xeon 3.33GHZ CPU and 32GB memory. For PCALap, we use different settings for small amounts of training data (Sections 4.2 and 4.4) and for large amounts of training data (Section 4.3). For Nyström, we also use different numbers of eigenvectors, but since we compute the embedding by finding all the eigenvectors, the running time and memory do not change.

Feature	Embedding	Dim.	Time (sec.)	Memory (GB)
(320 dim.)	Nyström	50	92	7.5
	PCALap (small)	100	24	2.4
	PCALap (large)	640	103	3.6
	RPLap	2,000	311	8.5
	RFF	2,000	60	12.5
(400 dim.)	Nyström	50	4,447	7.6
	PCALap (small)	100	32	3.5
	PCALap (large)	800	114	4.0
	SQRT	400	2	1.3
	HIKE	3,200	56	16.5
	AddKPCA	800	2,131	2.0

For **AddKPCA**, we follow the same settings as [15]. We use two eigenvectors per dimension, resulting in an 800-dimensional embedding for our 400-dimensional BoW features. To perform the Nyström interpolation, we randomly sample 128 landmark points per each dimension.

For **RFF**, we use 2,000 random projections. The bandwidth  $\gamma$  of the multi-dimensional Gaussian kernel is estimated in the same way as for Nyström.

For **HIKE**, we partition each of the 400 dimensions of the BoW histogram into 8 bins, resulting in a 3,200-dimensional embedding.

Table 2 summarizes the dimensionalities of all our embeddings, together with the time and memory requirements for computing them. Note that even though the dimensionalities of the different embeddings vary widely, our experiments will demonstrate that the highest-dimensional features are not always the most effective ones. Overall, in our evaluation, we took care to tune the performance of each method, and the settings we use are reasonably close to optimal in each case.

**Evaluation protocol.** We randomly split each dataset into three parts: 60% training, 20% testing, and 20% validation. The data-dependent embeddings are trained on the entire training set. Then, we randomly sample different numbers of labeled points from the training set and train a classifier based on the selected samples. We automatically tune the ridge/SVM regularization parameter over a grid [0.0002, 0.002, 0.02, 0.2, 2, 20, 200, 2000] on the validation set and finally test on the testing set. All the results are averaged over 100 random trials; error bars in the plots show the standard deviations.

## 4.2. Classification with little training data

First, we report multi-class classification results with very few labeled training points (3 to 30 per class). For

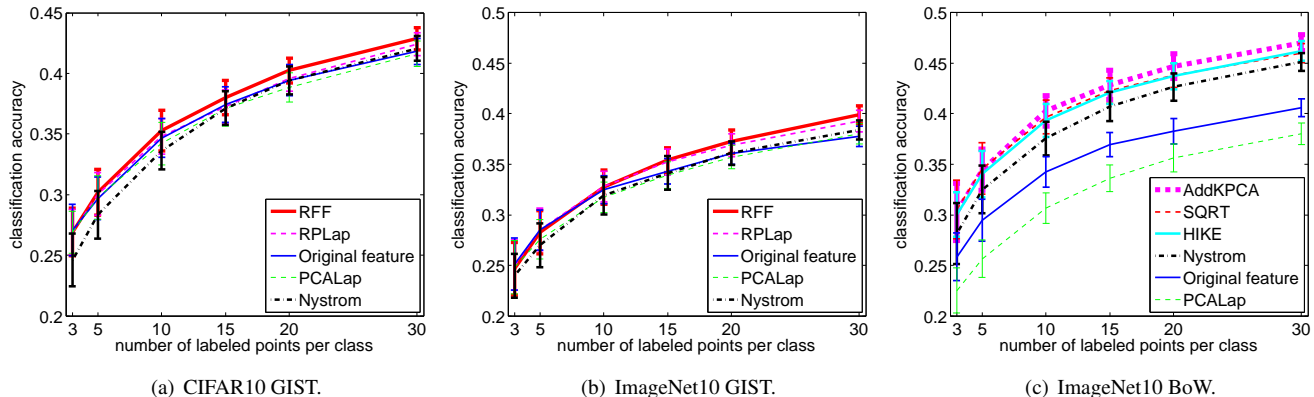


Figure 3. Average per-class classification rates for small amounts of labeled training data (best viewed in color). “Original feature” corresponds to running ridge regression on the original features (i.e., no embedding).

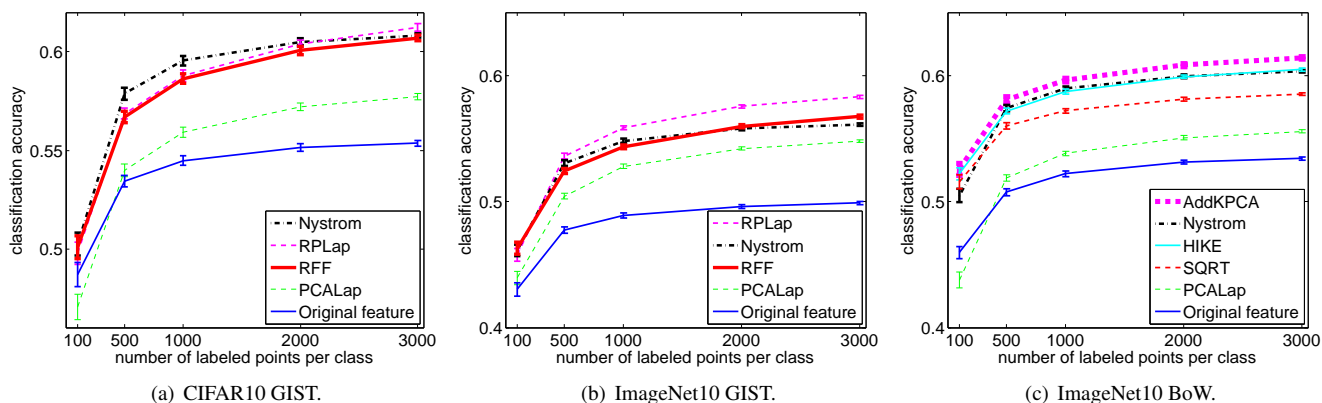


Figure 4. Average per-class classification rates for large training set sizes. Figure best viewed in color.

the data-dependent embeddings (Section 3.2.1), this corresponds to the SSL scenario. Therefore, the goal here is to find out whether unlabeled data can help to improve performance when very little labeled data is available.

Figure 3(a,b) compares the performance of different embeddings on GIST features for CIFAR10 and ImageNet10 datasets. All the methods perform very similarly, with no discernible advantage for the data-dependent embeddings. In fact, none of our nonlinear embeddings can significantly improve performance over the linear baseline. The situation is quite different for BoW features, as shown in Figure 3(c). Here, there is a wider separation between the methods. AddKPCA, HIKE and SQRT all achieve a consistent improvement of at least 5% over the linear baseline. Nyström is not far behind, while PCALap is actually below the linear baseline. It may be that PCA embedding is less well suited for histogram features than methods that are additive along the original feature dimensions.

Based on these results, we cannot see any clear advantage for data-dependent embeddings for very small training sets. This is significantly different from the results on the USPS digits dataset (Figure 1(b)) and from the findings in the traditional SSL literature.

### 4.3. Classification with lots of training data

This section reports multi-class classification results with large amounts of training data (100 to 3000 labeled points per class). As we can see from Figure 4(a,b), for GIST features, all the nonlinear embeddings gain a substantial improvement over the linear baseline. Nyström, RPLap, and RFF methods all get good performance. For BoW features, Figure 4(c) shows that AddKPCA has a small but consistent advantage, closely followed by HIKE and Nyström. The SQRT embedding, despite its simplicity, is not too far behind. PCALap is the weakest of the nonlinear embeddings, although, unlike in Figure 3(c), for large enough dataset sizes it can also improve over the linear baseline.

We can conclude that both data-dependent and data-independent embeddings are capable of strong performance: for GIST, the most competitive methods are RPLap, RFF, and Nyström; for BoW features, they are AddKPCA, HIKE, and once again, Nyström. Because the differences in performance are not very large, the choice of embedding in practice is likely to depend on computation and memory requirements (Table 2), as well as the ease of parameter tuning. While Nyström can achieve very good accu-

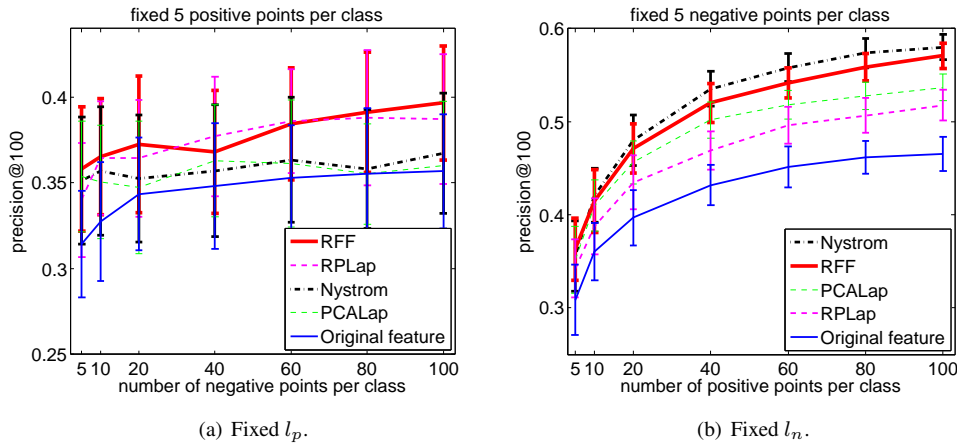


Figure 5. Ranking results using GIST features on ImageNet20K.  $l_p$  and  $l_n$  denote the numbers of positive and negative labeled points, respectively. Note the different vertical scales in (a,b). Best viewed in color.

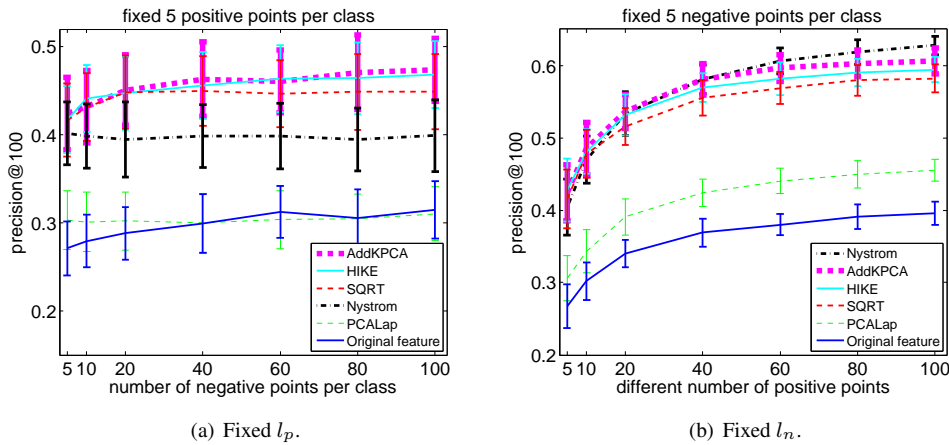


Figure 6. Ranking results using BoW features on ImageNet20K. Note the different vertical scales in (a,b). Best viewed in color.

racy in all cases, it is by far the slowest method, and its parameters (such as the number of eigenvectors and the number of landmark points) are very cumbersome to tune. For GIST features, the data-independent RFF method is by far the simplest. In some cases, RPLap can outperform it by a couple percent, though computing the RPLap embedding takes longer. For BoW features, AddKPCA works slightly better than HIKE and uses a fraction of the memory, which potentially makes it more practical for very large-scale tasks. Even though in our current implementation, AddKPCA takes 40 times longer to compute than HIKE, this computation can be easily parallelized and made more efficient. As for the SQRT embedding, even though it is 2-3% worse than AddKPCA and HIKE, it may still be a competitive choice due to its extreme simplicity and efficiency.

#### 4.4. Ranking

In this section, we take another look at the different embeddings by comparing them on ranking tasks. The idea behind these tasks is to mimic the scenario in which a user labels a few positive and negative images representing their

search preferences. These images are used to train a binary classifier, and the output of the classifier on the test images is used to rank them in decreasing order of their “relevance” to the points labeled by the user. In our case, the positive images are randomly sampled from a single target class, and the negative images are sampled from all the other classes. Performance is evaluated by the average precision of the class label at top 100 returned images. This measure makes for an interesting difference from the multi-class classification scenario, since it highlights how well the classifier does on points closer to the positive examples, as opposed to the ones closer to the decision boundary.

In the multi-class classification experiments of the previous section, we used an equal number of training points *per class*. Thus, for a problem with  $c$  classes, the number of *negative* examples for any given class was  $c - 1$  times the number of positive ones. In this section, we independently vary the numbers of positive and negative examples (denoted  $l_p$  and  $l_n$  respectively). For reasons of speed, we run these experiments on the ImageNet20K subset.

Figure 5(a) shows how ranking performance for GIST

features changes if we fix  $l_p = 5$  and vary  $l_n$ , and Figure 5(b) shows how it changes if we fix  $l_n = 5$  and vary  $l_p$ . By comparing the two figures, we can see that positive examples are generally much more “valuable” than negative ones: performance of all methods increases much more rapidly as we increase  $l_p$ . Beyond this, there are a couple of interesting observations. First, the performance of PCALap relative to the other methods seems to improve in some cases. For example, in Figure 5(b), it is better than RPLap. Second, in Figure 5(b), Nyström becomes slightly better than all the other methods as  $l_p$  grows much larger than  $l_n$ . Somehow, Nyström is able to make more efficient use of positive training examples than the other methods. The flip side of that is that in Figure 5(a), Nyström makes the most inefficient use of additional negative examples: as  $l_n$  increases, its performance hardly improves. Figure 6 shows analogous results for BoW features. The trends are similar to those of Figure 5: Nyström becomes especially effective when  $l_p$  is much larger than  $l_n$ .

Note that we have found data centering to be crucial in getting good results for increasing  $l_p$  (Figures 5(b) and 6(b)). When  $l_p$  becomes much larger than  $l_n$ , the distribution of the *labeled* points becomes very different from the distribution of the dataset as a whole (this is not the case when  $l_n$  is much larger than  $l_p$ , since the negative data is scattered all over the data space). Without translating the data to the mean of *all* the points, classifier performance would actually *decrease* with increasing  $l_p$ . Thus, there is at least one experimental situation (although one might argue it is not a very common or natural one) in which a very simple data-dependent transformation can make a big performance difference (as mentioned in Section 3.3, centering had little effect on performance in all the other cases).

## 5. Summary

Based on our experiments, we can conclude that adapting the nonlinear embedding to the data distribution has the potential to improve performance of large-scale visual learning, though this improvement is subtle and does not necessarily show up where one would expect, i.e., when labeled training data is scarce. In our tests, data-dependent embeddings had no advantage over data-independent ones on multi-class classification tasks with as few as three labeled points per class. We conjecture that due to the high intra-class variability in datasets such as CIFAR and ImageNet, the cluster assumption [2] does not hold, so the usefulness of traditional SSL methods is likely to be limited. On the other hand, in our tests with large amounts of training data, several data-dependent embeddings (Nyström, RPLap, and AddKPCA) have shown themselves capable of outperforming the best data-independent ones. But in general, the relative performance of different methods depends on many factors, including the dataset, the task (classifi-

cation vs. ranking), the feature representation (GIST vs. BoW), and the performance measure used (classification rate vs. precision of top-ranked images). Sometimes, even very simple data transformations such as centering can have a big effect on performance. This points to the need for more extensive theoretical and experimental investigations in order to gain a better understanding of the issues.

**Acknowledgements:** We thank Rob Fergus, Alex Berg, Florent Perronnin, and Joe Tighe for helpful discussions. This research was supported in part by NSF CAREER Award IIS 0845629, Microsoft Research Faculty Fellowship, Xerox, and ARO.

## References

- [1] M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. *NIPS*, 2002.
- [2] O. Chapelle, B. Scholkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.
- [3] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *PAMI*, 2010.
- [4] J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? *ECCV*, 2010.
- [5] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. *CVPR*, 2009.
- [6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 2008.
- [7] R. Fergus, A. Torralba, and Y. Weiss. Semi-supervised learning in gigantic image collections. *NIPS*, 2009.
- [8] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *PAMI*, 26(2):214–225, 2004.
- [9] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
- [10] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report. University of Toronto*, 2009.
- [11] S. Kumar, M. Mohri, and A. Talwalkar. Ensemble Nyström method. In *NIPS*, 2009.
- [12] S. Lazebnik, S. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing. *CVPR*, 2006.
- [13] S. Maji and A. Berg. Max-margin additive classifiers for detection. *CVPR*, 2009.
- [14] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 2001.
- [15] F. Perronnin, J. Sanchez, , and Y. Liu. Large-scale image categorization with explicit data embedding. *CVPR*, 2010.
- [16] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *NIPS*, 2007.
- [17] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.
- [18] B. Schölkopf, A. Smola, and K.-R. Müller. Non-linear component analysis as a kernel eigenvalue problem. *Neural Computation*, 1998.
- [19] T. Shi and M. Belkin. Data spectroscopy: Eigenspaces of convolution operators and clustering. *Annal of Statistics*, 2010.
- [20] A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. *CVPR*, 2008.
- [21] A. Torralba, R. Fergus, and W. Freenman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *PAMI*, 2008.
- [22] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.
- [23] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, pages 682–688, 2000.