SCaLE: Supervised and Cascaded Laplacian Eigenmaps for Visual Object Recognition Based on Nearest Neighbors

Ruobing Wu Yizhou Yu Wenping Wang
Department of Computer Science, The University of Hong Kong
{rbwu,yzyu,wenping}@cs.hku.hk

Abstract

Recognizing the category of a visual object remains a challenging computer vision problem. In this paper we develop a novel deep learning method that facilitates example-based visual object category recognition. Our deep learning architecture consists of multiple stacked layers and computes an intermediate representation that can be fed to a nearest-neighbor classifier. This intermediate representation is discriminative and structure-preserving. It is also capable of extracting essential characteristics shared by objects in the same category while filtering out nonessential differences among them. Each layer in our model is a nonlinear mapping, whose parameters are learned through two sequential steps that are designed to achieve the aforementioned properties. The first step computes a discrete mapping called supervised Laplacian Eigenmap. The second step computes a continuous mapping from the discrete version through nonlinear regression. We have extensively tested our method and it achieves state-of-the-art recognition rates on a number of benchmark datasets.

1. Introduction

Recognizing the category of a visual object is an important and challenging aspect of automatic object recognition. Visual objects from the same category may exhibit a wide range of shape and appearance variations due to many reasons, including inherent within-class shape and appearance diversity as well as the presence of deformation and illumination changes. Humans have a remarkable ability in recognizing object categories despite such within-class shape and appearance variations. It is a common practice in computer vision to develop algorithms that are inspired by mechanisms in the human visual system. Deep learning methods, such as convolutional neural networks [16], are a good example of this approach. They are biologically inspired by complex multilayer hierarchies in the human visual cortex.

It is well known that learning plays a significant role in the development of human brain functionalities. Learned knowledge is often represented as a collection of cases and examples. In the event of visual object recognition, when presented with a new object, humans often draw conclusions from their past experiences [7]. That is, we consciously or subconsciously compare the new object against examples of objects we have seen in the past. Nevertheless, such comparisons are not always performed on the original visual impressions where the directly observed shape and appearance of the objects may be very different even when they belong to the same category. Therefore, correctly recognizing visual object categories with large within-class diversity requires the brain to form an intermediate representation that grasps the essence of those characteristics shared by objects in the same category while filtering out nonessential differences. Objects with similar intermediate representations in this nature are more likely to belong to the same category.

Our goal is to develop a deep learning method that facilitates example-based reasoning for visual object category recognition. Inspired by the important role played by examples in human visual recognition, we would like to determine the category of an object by computing its similarities with a set of examples with known category labels. As discussed above, such similarities need to be computed with the assistance of a proper intermediate representation, which is crucial in achieving high recognition rates. Since this intermediate representation needs to grasp the essence of an object's attributes, it is likely to be the result of a complicated transformation applied to the features extracted from original input images. Since multilayer deep learning architectures have a reputation for learning such transformations, we devise a novel deep learning architecture for computing a desired intermediate representation.

There exist a few requirements on the intermediate representation we learn. First, it needs to be discriminative. The intermediate representation of a visual object is in fact a point in a new multidimensional feature space, and the dissimilarity between two objects can be measured by the Euclidean distance between their corresponding points. In

this context, being discriminative means points with different category labels should be further away from each other than those with same category labels. Second, nonessential differences between objects with the same category label should be suppressed, which has been much discussed earlier. Third, essential structures in image feature distribution should be retained. It is important for supervised learning to capture intrinsic connections between the input (image features) and the output of the trained system. Otherwise, the trained system would have poor generalization capability.

In summary, we propose a supervised layered visual object category classification framework inspired by recent deep learning methods. Our framework is simple and powerful. It consists of three stages. The first stage is usual feature extraction. In the second stage, a layered model is applied to the extracted features to obtain a desired intermediate representation. In the last stage, the intermediate representation is fed to a classifier based on nearest neighbors. Our main contribution is the layered model used in the second stage. This model is capable of obtaining discriminative and structure-preserving intermediate representations supporting nearest-neighbor classification. At each layer, the output representation from the previous layer is taken as the input and is nonlinearly mapped. The result of this mapping defines the output representation at the current layer. This nonlinear continuous mapping approximates (through regression) a discrete mapping called *supervised Laplacian Eigenmap*, which is the optimal solution of an energy function designed to address the aforementioned three requirements. We have tested our framework on a number of benchmark datasets. It achieves recognition rates higher than other state-of-the-art techniques.

1.1. Related Work

Many image classification and object category recognition approaches [26, 31, 17, 28] have been developed to process medium-scale benchmark datasets such as Caltech 101 [14], Caltech 256 [18] and PASCAL VOC [13]. There are also attempts made on the large-scale ImageNet [12] dataset, which includes millions of images distributed in more than 20,000 classes.

Deep learning has received much attention recently and been employed to further improve image classification accuracy. It relies on multiple levels of transformation and abstraction to extract the essence of the input data. Thus, deep learning simulates the subconscious process undertaken by humans when they make classification decisions. Deep belief networks (DBN) are a type of deep learning models where lower-level features are progressively combined into more compact high-level representations. Stacked denoising autoencoders [27] represent another deep learning architecture where each layer serves as a compact encoder of its previous layer. In recent years, deep convolutional neural networks (DNN) [16], have demonstrated their capability of automatically extracting spatial and spatial-temporal features from raw images while being resistant to object appearance variations in the images. Recently, a multi-column DNN (MCDNN) framework [10] has much improved the state-of-the-art recognition rates on several benchmark datasets, including MNIST [20] and NORB [21]. Nevertheless, training such DNNs is very time-consuming even with the help of GPUs. It is also hard to comprehend the internal structures of a DNN.

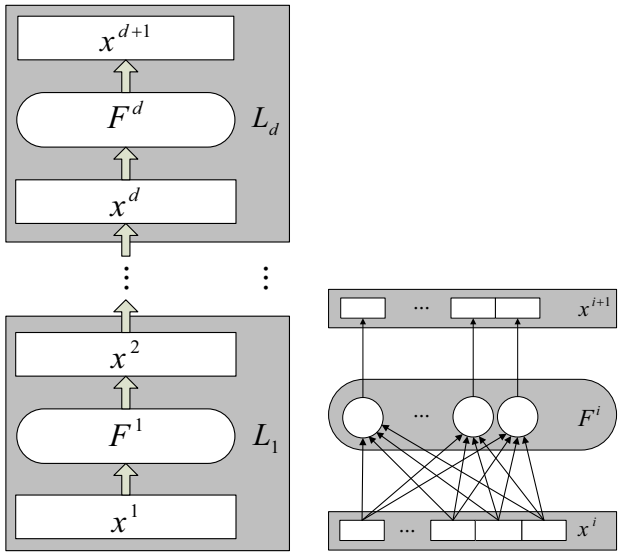


Figure 1. LEFT: A diagram of our layered model. RIGHT: A detailed illustration of one layer, where an input feature vector $\boldsymbol{x}^i$ is mapped to an output feature vector $\boldsymbol{x}^{i+1}$ through F^i, which itself has multiple components. Each component maps $\boldsymbol{x}^i$ to one of the elements in $\boldsymbol{x}^{i+1}$.

2. Architecture

The architecture of our learning system, as demonstrated in Figure 1, is a layered model which aims at constructing a discriminative and structure-preserving image feature space. Images from the same category have inherent connections and similarities even though they may differ visually in geometric deformation, color, texture, illumination, etc. Our model is designed to dig out hidden similarities and force together images with the same category label in a new feature space. Meanwhile, the new feature space needs to be sufficiently discriminative and structure-preserving. That means feature vectors with different category labels should be further away from each other than those with same category labels, and feature distribution in the new space should be regularized. Since it would be hard to meet these goals at once for challenging datasets, in the spirit of deep learning, we envision compositing multiple layers of nonlinear transformations together would be a more feasible solution.

Suppose there exist d layers in our model. Each layer performs a (nonlinear) mapping between an input feature

space and an output feature space. Let us first focus on a single layer $L_i (1 \leq i \leq d)$. The training data for L_i is denoted as $\{(\boldsymbol{x}_j^i, l_j)\}_{j=1}^N$, where $\boldsymbol{x}_j^i (\in R^{s_i})$ is a training sample, l_j is its category label, which is independent of the layer index, N is the number of training samples, and s_i denotes the dimensionality of the training samples. There are two types of training samples. For the first layer, they are features extracted from raw training images. For any of the subsequent layers, they are actually transformed features produced by the previous layer $L_{i-1} (i > 1)$. In this context, we define the mapping at L_i as $F^i : R^{s_i} \rightarrow R^{s_{i+1}}$. Let $S^i(j)$ be the set of n nearest neighbors of $\boldsymbol{x}_j^i$, and $W_{jk}^i (\geq 0)$ be a similarity measure between $\boldsymbol{x}_j^i$ and $\boldsymbol{x}_k^i$.

Fulfilling the previously mentioned objectives amounts to the following optimization problem.

$$F^i = \arg\min \lambda^i E_{pull} + \mu^i E_{push} + (1 - \lambda^i - \mu^i) E_{inner}, \tag{1}$$

where λ^i and μ^i are weights for different energy terms.

Specific formulations of the three energy terms in (1) are discussed as follows. The first term,

$$E_{pull} = \sum_j \sum_{\substack{k \in S^i(j), \\ l_j = l_k}} ||F^i(\boldsymbol{x}_j^i) - F^i(\boldsymbol{x}_k^i)||^2, \tag{2}$$

is a pulling energy that attempts to pull closer a training sample and its neighbors with the same category label in the output feature space $R^{s_{i+1}}$.

The second term,

$$E_{push} = \sum_j \sum_{\substack{k \in S^i(j), \\ l_j = l_k}} \sum_{\substack{q \in S^i(j), \\ l_j \neq l_q}} [1 + {d_{jk}^i}^2 - {d_{jq}^i}^2]_+, \tag{3}$$

where $d_{jk}^i = ||F^i(\boldsymbol{x}_j^i) - F^i(\boldsymbol{x}_k^i)||$, is a pushing energy that attempts to push neighbors with different category labels at least a unit margin further away in the output feature space than neighbors with same labels. The term $[z]_+ = \max(z, 0)$ is the standard hinge loss, which makes this term disappear once the unit margin has been satisfied.

The third term,

$$E_{inner} = \sum_j \sum_{k \in S^i(j)} ||F^i(\boldsymbol{x}_j^i) - F^i(\boldsymbol{x}_k^i)||^2 W_{jk}^i, \tag{4}$$

is a standard structure-preserving energy that keeps training samples close in the output feature space if they have high similarity in the input feature space. Unlike previous two energy terms, this one does not take labels into consideration.

This last energy term is very similar to the energy used for deriving Laplacian Eigenmap [1] except that we seek a continuous mapping F^i instead of a discrete embedding and there is no normalization constraint on the magnitude of the mapped training samples. Furthermore, the first two energy terms have a supervised nature since they make use of labels while the third one has an unsupervised nature. These are the reasons we call the mapping we learn at each layer a supervised Laplacian Eigenmap.

The three energy terms in our optimization have competing effects at the same time being complementary to each other. For example, without a normalization constraint, minimizing E_{inner} in (4) alone would make all input samples collapse to a single point in the output feature space. However, the unit margin in (3) can effectively prevent that from happening. On the other hand, minimizing E_{pull} in (2) alone would also have the potential of collapsing input samples with the same category label to a single point in the output feature space. E_{inner} has the effect of a regularization term. Its structure-preserving functionality can make such samples more spread out according to their mutual similarities. Results in Figure 2 show that minimizing the sum of all three energy terms with proper weights can yield much better (7% to 9% in accuracy) recognition performance than minimizing $E_{push} + E_{pull}$ only or E_{inner} only.

Our complete model consists of multiple layers of continuous mappings stacked on top of each other, i.e. $F^i(\boldsymbol{x}_j^i) = \boldsymbol{x}_j^{i+1}$. Each layer is constructed and learned in the same way while using potentially different parameters, which include the dimensionality s_i of input features, and the weights λ_i and μ_i for different energy terms. Note that our model only takes care of the middle portion of the entire pipeline. It neither extracts features from a raw input image nor performs final classification to determine the category label of the input image. It only performs nonlinear transformations that eventually define a final feature space through multiple intermediate feature spaces defined at intermediate layers. Given an input feature vector, $\boldsymbol{x}^{\mathbf{1}} (\in R^{s_1})$, to the first layer, the output feature vector, $F^d(F^{d-1}(...F^1(\boldsymbol{x}^{\mathbf{1}})...)) (\in R^{s_d})$, of the last layer is fed to a final multi-category classifier.

Our general strategy to solve the optimization in (1) is to decompose it into two sequential subproblems. In the first subproblem, we seek a discrete mapping between an input feature space and an output feature space for the training samples only. This discrete mapping minimizes the same energy as defined in (1). In the second subproblem, we seek a continuous mapping by performing regression on the discrete results obtained in the first subproblem. This general strategy is motivated by the fact that directly minimizing (1) can be easily stuck with local minima while solving the two simpler subproblems may actually gives rise to better solutions. The actual implementation of these two steps will be elaborated in the following section.

3. Implementation

3.1. Feature Extraction

We use concatenated combinations of PHOG [3], SIFT [22], and CENTRIST [30] as the initial feature vector from raw input images. Such feature vectors are used as the input to the first layer of our learned model. Detailed feature settings with respect to individual datasets can be found in Section 4.

PHOG [3] partitions an image into a pyramid of nested subregions, and computes a histogram of oriented gradients (20 bins or 40 bins) over each subregion at each pyramid level. In our experiments, we typically use oriented (Shp_{360}) PHOG, with 40 bins and 3 levels. The total dimensionality is $40\sum_{l=0}^{3} 4^l = 3400$. We also use SIFT [22] to model appearance and adopt a codebook with 1024 to 4096 grey-scale keypoint descriptors in our implementation. CENTRIST [30] is used for datasets where the contour of foreground objects plays a crucial role in classification. CENTRIST is performed on the Sobel filtered images, and fully takes advantage of local sign information.

Since the visual objects that need to be recognized typically belong to a salient foreground region of an image, we first perform saliency detection as in [9] and then threshold the resulting saliency map. We further obtain a bounding box for the pixels with saliency above the threshold. Feature descriptors are only computed for the part of the image inside the bounding box. Most of the feature descriptors we use involve histograms. During the computation of a histogram, the feature value at a pixel is cast into a bin after being multiplied by the saliency value of that pixel. Then the summation of all the bins in the histogram is normalized. By doing so, foreground pixels within the bounding box have a greater influence on the feature vector we construct. Compared with region-of-interest (ROI) selection in [4], we do not need to run an optimization over the entire training set. Instead, it is only necessary to precompute and save the saliency map and its associated bounding box for each training image, once and for all.

3.2. Discrete Mapping

As discussed earlier, we take two sequential steps to solve the optimization in (1). The first of these two steps is to compute a discrete mapping that minimizes (1). Let $Y_j^i = F^i(\boldsymbol{x}_j^i)(j = 1, \ldots, N)(\in R^{s_{i+1}})$. Instead of taking the continuous mapping F_i as the unknown in the minimization, we take $\{Y_j^i\}_{j=1}^N$ as a set of mutually independent unknowns and replace all $F_i(\boldsymbol{x}_j^i)$'s in (2)-(4) with Y_j^i. Note that the input feature vectors of the discrete mapping are known since we solve the mapping for one layer at a time. The resulting problem is still a challenging nonconvex optimization with local minima. Nevertheless, converting from a continuous mapping to a discrete mapping makes it easier to compute a good initial solution, which is often crucial in obtaining a high-quality final solution for nonconvex problems.

Because of the similarity between the energy term in (4) and the energy for Laplacian Eigenmap [1], we decided to take the Laplacian Eigenmap as the initial solution to our discrete mapping. Laplacian Eigenmap is capable of producing a nonlinear embedding that preserves important local structures while removing non-significant features. With the set of input vectors $\{Y_j^{i-1}\}_{j=1}^N$ at layer L_i in mind, we take the following steps. The adjacency graph among these input vectors is defined by the set of nearest neighbors, $S^i(j)$, of Y_j^{i-1}. An edge in the graph is associated with a heat kernel weight, $W_{jk}^i = e^{-||Y_j^{i-1} - Y_k^{i-1}||^2/t_i}$. For each connected component of the adjacent graph, the eigenvalues and eigenvectors of the following problem is solved: $\tilde{L}Y = \lambda DY$, where D is a diagonal matrix with $D_{jj} = \sum_k W_{kj}^i$ and $\tilde{L} = D - W$ is the Laplacian matrix. This is equivalent to minimizing the E_{inner} term along with the constraint $Y^T DY = 1$ to avoid arbitrary scaling [1].

Although Laplacian Eigenmap generates a starting point for our optimization, it does not take category label information into consideration. We further minimize the complete energy function defined in (1) from the initial solution provided by Laplacian Eigenmap using gradient descent, which typically result in a high-quality solution to the discrete mapping we seek in our experiments.

Note that energy terms similar to E_{pull} and E_{push} in a discrete setting have been previously exploited in metric learning literature, including the Large Margin Nearest Neighbor (LMNN) approach in [29], which learns a Mahanalobis distance metric for k-NN classification. Compared with LMNN, we directly optimize the coordinates of transformed feature vectors in a new s_{i+1}-dimensional space instead of just training a linear transformation for the original feature space. The total number of unknowns in our problem is $N \times s_{i+1}$, which is much larger than s_{i+1}^2 (for LMNN) when $N >> s_{i+1}$. Furthermore, the new feature vectors learned from our problem formulation is typically the result of a more powerful nonlinear transformation.

3.3. Continuous Mapping

So far we have obtained a discrete mapping which redistributes training samples in a new feature space according to the three requirements discussed in the Introduction. A continuous mapping is then learned from the discrete mapping through nonlinear regression, which essentially solves another optimization problem that seeks a continuous mapping that can approximately satisfy the constraints defined by the discrete mapping, i.e. $\{F^i(\boldsymbol{x}_j^i) = Y_j^i\}_{j=1}^N$, where $\boldsymbol{x}_j^i = F^{i-1}(\boldsymbol{x}_j^{i-1})$, which means we fit a continuous mapping at layer L_i immediately after computing the discrete mapping, and when we work on layer L_{i+1}, we take the

output from the continuous mapping at L_i as the input when solving both the discrete and continuous mappings there.

We have experimented with multiple nonlinear regression techniques from the literature, including kernel SVM regression, random forests [5], and radial basis functions (RBFs) [6]. In our context, Gaussian RBFs achieve the highest generalization accuracy. Since the output feature space of layer L_i is s_{i+1}-dimensional, we actually perform RBF regression s_{i+1} times once for each dimension. The Gaussian RBF approximation of the k-th element of the output feature vector has the following form:

$$f_k^i(\boldsymbol{x}) = \sum_{j=1}^{N_{rbf}^i} \theta_{k,j}^i \phi_{k,j}^i(\boldsymbol{x}), \tag{5}$$

where $\phi_{k,j}^i(\boldsymbol{x}) = e^{-||\boldsymbol{x}-\boldsymbol{x}_{k,j}^i||^2/2{\sigma_i}^2}$. σ_i is the standard deviation which is estimated according to the average distance between pairs of closest kernel centers in the input feature space. It has been shown that any continuous function on a compact interval can in principle be approximated to an arbitrary accuracy with a sum in this form if a sufficiently large number of radial basis functions are used. In practice, however, too many RBF kernels would eventually overfit noise and outliers, and consequently lower the generalization capability of the learned model. In our experiments, the centers of Gaussian kernels, $\{\boldsymbol{x}_{k,j}^i\}$ are chosen to be a random subset of training samples, whose size is between 1/50 to 1/10 of the training set size. The weights, $\{\theta_{k,j}^i\}_{j=1}^{N_{rbf}}$ are optimized to minimize the following summed squared errors (SSE):

$$SSE_{i,k} = \sum_{j=1}^{N} ||f_k^i(\boldsymbol{x}_j^i) - Y_j^i(k)||^2, \tag{6}$$

where $Y_j^i(k)$ denotes the k-th element of Y_j^i.

3.4. Classification

The energy terms E_{push} and E_{pull} in our optimization make it naturally suitable (slightly better than using linear and kernel SVMs) to apply k-NN classification (voting among k nearest neighbors) on the final feature vector $R_d(R_{d-1}(...R_1(\boldsymbol{x})...))$ in our layered model. The final feature vectors of all training samples are stored in a k-d tree for fast nearest neighbor lookup.

4. Experiments

In this section, we report experimental results on four widely used datasets: Caltech 101 [14], Caltech 256 [18], PASCAL VOC 2007 [13] and NORB [21]. Our method is compared against other state-of-the-art techniques over each dataset. All experiments were conducted on an Intel Xeon E5-2690 2.90GHz CPU with 128GB RAM. Each reported result is an average of 10 different runs.

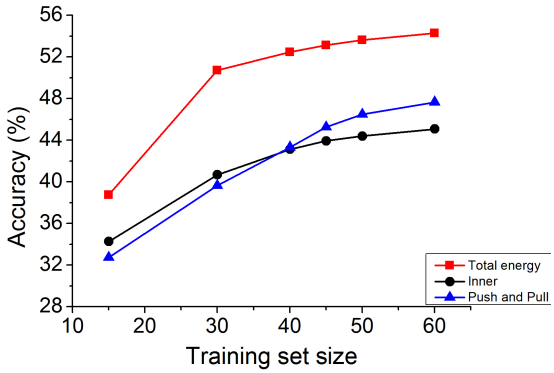
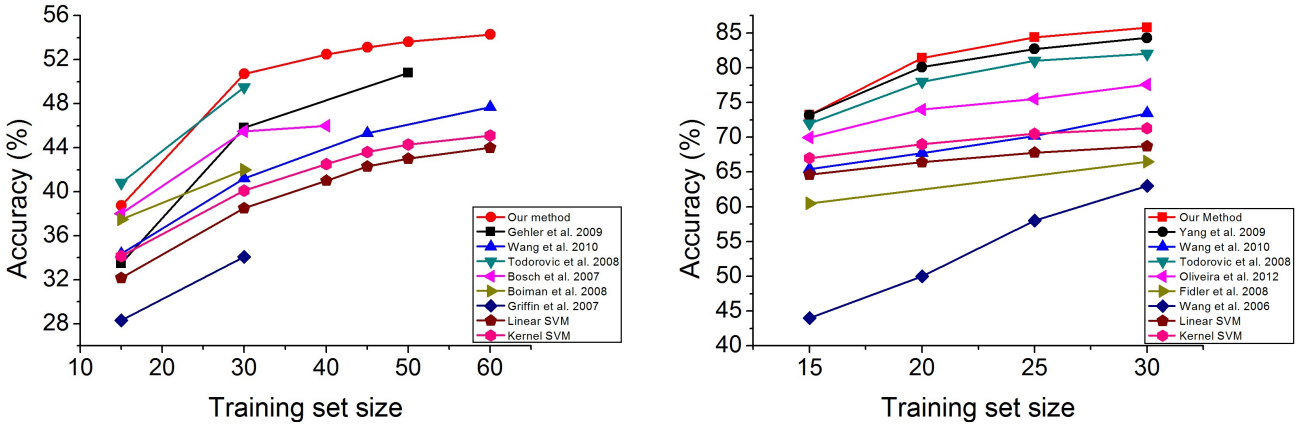


Figure 2. Results on Caltech 256 by optimizing different energy terms. The "Total energy" curve was achieved by optimizing the sum of all three terms in (1). The "Inner" curve was achieved by merely minimizing the E_{inner} term without pushing and pulling. The "Push and Pull" curve was achieved by minimizing $E_{push} + E_{pull}$ (equal weights) only. Note that for fair comparison, we use random initializations for the "Push and Pull" curve instead of the way introduced in Section 3.2.

4.1. Caltech Datasets

The Caltech 256 dataset [18] contains 30,607 images in 256 categories with large variance in foreground object size, pose, and texture. Each class contains no less than 80 images. We randomly chose 15, 30, 40, 45, 50, 60 training images per class and no more than 100 test images. SIFT (codebook size 1024) was concatenated to PHOG (Shp_{360}, 3 levels) to construct the initial feature vector. 7 layers were used, at the first 4 of which the dimensionality was reduced by half with $\lambda = \mu = 0.35$ while in the others the dimensionality was reduced by a third with $\lambda = \mu = 0.45$. The size of a nearest-neighbor set during training was 31 (for 60 training images) for all layers. 580 RBF kernels were used in the regression stage. Results and comparisons with state-of-the-art techniques can be found in Figure 3(a). When using 15, 30, 40, 45, 50, 60 training images, our recognition rates are: 38.74%, 50.69%, 52.48%, 53.12%, 53.63% and 54.19% ($\pm$0.24%). Our method gives competitive results with 15 training images per category, and significantly leads the performance when the training set becomes larger. For example, with 50 training images per category, our recognition rate (53.63%) is 2.83% higher than the second best result (50.8%) achieved in [17]. Note that even using a single feature descriptor, such as PHOG or SIFT, our method still outperforms all existing methods. With 50 training images per category, our recognition rates are 53.13% (PHOG) and 53.49% (SIFT).

Our method also exhibits similar advantages over the Caltech 101 dataset [14], which contains 9,144 images with significant shape variance in 102 classes. Similar to the setting for Caltech 256, PHOG and SIFT were used as the initial features. 5 layers were used. At each layer the dimen-



(a) Comparison results on Caltech 256

(b) Comparison results on Caltech 101

Figure 3. (a) Comparison on Caltech 256 between our method and [17, 28, 26, 4, 2, 18] as well as baseline linear and kernel SVMs with the same input features; (b) Comparison on Caltech 101 between our method and [31, 28, 26, 23, 15, 19] as well as baseline linear and kernel SVMs with the same input features.

sionality is reduced by half. $\lambda = 0.2$ and $\mu = 0.3$ for the first 4 layers, and at the last layer $\lambda = 0.4$ and $\mu = 0.5$. The nearest neighbor set size was 16 (for 30 training images) for all layers. 190 RBF kernels were used for regression. Results and comparisons are shown in Figure 3(b). When using 15, 20, 25, 30 training images, our accuracy rates are: 73.19%, 81.42%, 84,39% and 85.77% ($\pm$0.35%). Our method works extremely well for 23 categories, such as car, human face, pyramid, and etc., with 100% accuracy. Like other deep learning methods, our method requires a sufficient amount of training data to perform well, and when compared to other approaches, yields better performance once the training set becomes sufficiently large.

4.2. PASCAL VOC 2007

The PASCAL VOC 2007 dataset [13] has 9,963 images (5011 for training and validation, 4952 for testing) of 20 classes from Flicker. Classification here is challenging due to casual styles of daily photos. PHOG and SIFT (4096 codebook size) were used as the initial feature. There were 10 layers in our model. At each layer the dimensionality was reduced by a third with $\lambda = \mu = 0.15$. The number of nearest neighbors used for training was 26 for all layers. Note that since we are confronted with multiple labels here, a training sample is considered to belong to a category as long as they share a certain label. 480 RBF kernels were used in the regression stage for all layers. Evaluations were performed using Average Precision (AP) introduced in [13]. Results and comparisons are given in Table 1. Our method delivers an overall best mean score (64.1), and also achieves the best recognition rate in 6 of the 20 categories.

4.3. NORB

NORB [21] is a collection of stereo images of 3D models for object shape recognition. These objects are centered on randomly chosen backgrounds cluttered with other objects. This dataset has images of 50 toys in 6 classes (5 genuine categories and 1 background category), taken by 2 cameras under 6 lighting conditions, 9 elevations (30 to 70 degrees every 5 degrees), and 18 azimuths (0 to 340 every 20 degrees). There are 291600 training images (in 10 folds) in total with 58320 test images.

We chose CENTRIST as the initial feature because object contours are the most important features for NORB. There were 8 layers in our model. At each layer the dimensionality is reduced by a quarter with $\lambda = \mu = 0.45$. The size of a nearest-neighbor set during training was 600 for all layers. 360 RBF kernels were used in the regression stage for all layers. Results using 2 folds (58320) of training data and comparisons with state-of-the-art techniques can be found in Table 2. We are further encouraged to observe that with 10 folds of training images, our method achieves 97.67% accuracy which outperforms another recent deep learning architecture [10] (97.3% accuracy), which has set records on a few benchmark datasets including NORB. With large weights assigned to E_{push} and E_{pull}, our method is capable of well separating images from different categories especially when each category has a relatively large number of training images.

Method	[25]	[11]	[10]	Our method
Accuracy	94.4%	95%	96.43%	96.87$\pm$0.18%

Table 2. Comparison on NORB between our method and those in [25, 11, 10] using 2 folds of training images.

Class	aero	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	
Chatfield et al. 2011	79.0	67.4	51.9	70.9	30.8	72.2	79.9	61.4	56.0	49.6	
Yang et al. 2009	79.4	62.4	**58.5**	70.2	**46.6**	62.3	75.6	54.9	**64.8**	40.7	
Zhou et al. 2010	79.4	**72.5**	55.6	**73.8**	34	72.4	**83.4**	**63.6**	56.6	**52.8**	
Ours	**79.8**	66.1	57.9	72.5	46.4	**72.8**	82.6	57.5	63.1	47.6	
Class	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Mean
Chatfield et al. 2011	58.4	44.8	78.8	70.8	85.0	31.7	**51.0**	56.4	80.2	57.5	61.7
Yang et al. 2009	58.3	**51.6**	79.2	68.1	**87.1**	**49.5**	48.8	56.4	75.9	54.4	62.2
Zhou et al. 2010	63.2	49.5	**80.9**	71.9	85.1	36.4	46.5	59.8	83.3	**58.9**	64.0
Ours	**63.7**	47.2	78.2	**72.1**	85.9	46.6	45.4	**61.2**	**83.6**	51.8	**64.1**

Table 1. Comparison on PASCAL VOC 2007 between our method and those in [8, 31, 32].

4.4. Discussions

In this section, we explore the effects of model depth d, the number of nearest neighbors n during training and the number of RBF kernels N_{rbf} on final classification performance. We fix other parameters and observe how recognition accuracy varies when we change the value of only one of these three parameters. Related curves on Caltech 101 can be found in Figure. 4.

If we reach a certain feature dimensionality through different models with different numbers of layers, we obtain the result in Figure. 4(a). Note that the dimension reduction rate is different in models with different numbers of layers. The final dimensionality is 1/32 of the original. When the number of layers is too small, we may lose key feature dimensions from the beginning. However, adopting too many layers does not necessarily increase the recognition rate after a certain value (5 in this case) since regression introduces errors which gradually decrease the quality of the mapped feature vectors. A proper choice of model depth contributes to a high accuracy with less computation.

The curve of accuracy with respect to the size of the nearest-neighbor set is interesting. It shows that with only a small number of neighbors the adjacency information is not sufficient. On the other hand, too many neighbors would make an entire category almost shrink to a single point in the new feature space. Some value in the middle of these two extremes can deliver the best performance, and this value is related to the number of training samples in each category.

A similar phenomenon has also been observed on the curve for the number of RBF kernels. Too few kernels cannot produce sufficiently accurate results while too many of them gives rise to overfitting. The proper number of RBF kernels depends on the underlying structures of the dataset. For example, a dataset with a small number of tight clusters likely needs a relatively small number of kernels only while a dataset with a more spread out distribution needs a larger number of kernels.

5. Conclusions and Future Work

We have presented a supervised deep learning method for visual object category recognition based on nearest neighbors. In a flexible layered architecture, discriminative and structure-preserving intermediate data representations are obtained for modeling complex nonlinear feature spaces. The performance of our method on widely used benchmark datasets exceeds other state-of-the-art techniques. Such performance demonstrates that classification based on nearest neighbors can become at least as powerful as other classification approaches if a suitable feature space is learned. In the extreme case, if we could learn a continuous mapping from the raw input data to a feature space identical to the label space where objects in the same category share the same value, a nearest-neighbor classifier would be able to achieve perfect accuracy.

Nevertheless, our method is by no means fully optimized. In future, we aim at further boosting recognition rates by exploring more effective feature descriptors used as the initial feature fed to the first layer in our model. For example, Fisher Vector [24, 32], which has been proved to perform well on many datasets including ImageNet, is a potential choice of dense descriptors. Domain transformation techniques other than the Laplacian Eigenmap could be explored to remove the restriction that the dimensionality of intermediate feature vectors cannot be increased from layer to layer. Parallel or cluster computing techniques will be considered to speed up the computation so that our framework can be scaled to much larger image datasets.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. 3, 4

[2] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008. 6

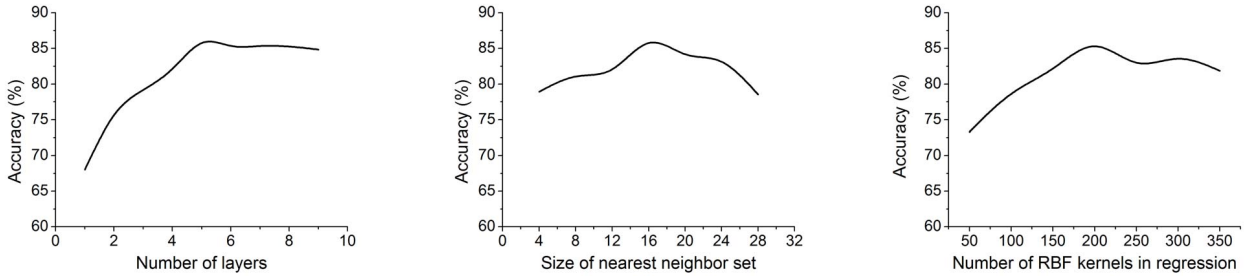


Figure 4. Recognition accuracy with respect to varying parameters on Caltech 101. 30 training images were used.

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

[4] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *ICCV*, 2007. 4, 6

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 5

[6] M. Buhmann. *Radial basis functions: theory and implementations*. Cambridge university press, 2003. 5

[7] G. A. Carpenter and S. Grossberg. Neural dynamics of category learning and recognition: Attention, memory consolidation, and amnesia. *Advances in Psychology*, 42:239–286, 1987. 1

[8] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011. 7

[9] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu. Global contrast based salient region detection. In *CVPR*, 2011. 4

[10] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR*, 2012. 2, 6

[11] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, 2011. 6

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2

[13] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2007 (voc 2007) results (2007), 2008. 2, 5, 6

[14] 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. *Computer Vision and Image Understanding*, 106(1):59–70, 2007. 2, 5

[15] S. Fidler, M. Boben, and A. Leonardis. Similarity-based cross-layered hierarchical representation for object categorization. In *CVPR*, 2008. 6

[16] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. 1, 2

[17] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009. 2, 5, 6

[18] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007. 2, 5, 6

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

[20] Y. LeCun and C. Cortes. The mnist database of handwritten digits, 1998. 2

[21] Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004. 2, 5, 6

[22] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 4

[23] G. Oliveira, E. Nascimento, A. Vieira, and M. Campos. Sparse spatial coding: A novel approach for efficient and accurate object recognition. In *ICRA*, 2012. 6

[24] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 7

[25] D. Scherer, A. Müller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. *ICANN 2010*, pages 92–101, 2010. 6

[26] S. Todorovic and N. Ahuja. Learning subcategory relevances for category recognition. In *CVPR*, 2008. 2, 6

[27] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pages 1096–1103, 2008. 2

[28] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 2, 6

[29] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006. 4

[30] J. Wu and J. Rehg. Centrist: A visual descriptor for scene categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1489–1501, 2011. 4

[31] J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao. Group-sensitive multiple kernel learning for object categorization. In *ICCV*, 2009. 2, 6, 7

[32] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010. 7