Unsupervised Feature Learning by Deep Sparse Coding

Yunlong He* Koray Kavukcuoglu† Yun Wang‡ Arthur Szlam § Yanjun Qi¶

Abstract

In this paper, we propose a new unsupervised feature learning framework, namely Deep Sparse Coding (DeepSC), that extends sparse coding to a multi-layer architecture for visual object recognition tasks. The main innovation of the framework is that it connects the sparse-encoders from different layers by a sparse-to-dense module. The sparse-to-dense module is a composition of a local spatial pooling step and a low-dimensional embedding process, which takes advantage of the spatial smoothness information in the image. As a result, the new method is able to learn multiple layers of sparse representations of the image which capture features at a variety of abstraction levels and simultaneously preserve the spatial smoothness between the neighboring image patches. Combining the feature representations from multiple layers, DeepSC achieves the state-of-the-art performance on multiple object recognition tasks.

1 Introduction

Visual object recognition is a major topic in computer vision and machine learning. In the past decade, people have realized that the central problem of object recognition is to learn meaningful representations (features) of the image/videos. A large amount of focus has been put on constructing effective learning architecture that combines modern machine learning methods and in the meantime considers the characteristics of image data and vision problems.

One thread of the state-of-the-art systems for visual object recognition is the bag-of-visual-words (BoV) framework. The classic BoV framework utilizes a pipeline of multiple stages, i.e., "local descriptor extraction", "coding", "spatial pooling" [20] and "Support Vector Machine" (SVM) classifier, and has achieved excellent performance on multiple benchmarked image classification datasets [6]. In this pipeline, feature learning is performed by the unsupervised "coding" stage, and the image labels are only used in the final step of BoV pipeline to train the classifier. Recent literature has been focused on refining the unsupervised "coding" step, which encodes the (hand-crafted) feature descriptor of local image patches by a set of visual codewords. Sparse coding (SC) [25], one of the most successful signal processing paradigms, was proposed as the "coding" step in BoV framework by [27] to replace vector quantization. By utilizing just a linear SPM kernel based on the sparse codes, their approach reduced the training and testing complexities of SVM, and has achieved the state-of-the-art performance on several benchmarks.

Another thread of research consist of a number of deep learning methods that aim at extracting feature hierarchies from data. Convolutional Neural Network [21] and Deep Belief networks [16] are the early works in this area. Most of the architecture in deep learning(see [2]) is constructed by a stack of feature extractors, such as Restricted Boltzman Machine(RBM) and auto-encoder. Each layer in the architecture encodes features at different level of abstraction, defined as a composition of lower-level features. More recently, deep learning based systems have won a number of high-profile competitions, e.g. [19] and have gained great popularity.

In this work, we combine the power of deep learning architecture and the sparse coding BoV pipeline to construct a new unsupervised feature learning architecture for learning image representations.

- Compared to the single-layer sparse coding framework, our method can extract feature hierarchies at the different levels of abstraction. The sparse codes at the same layer keeps the spatial smoothness across image patches and different SC hierarchies also capture different spatial scopes of the representation abstraction. As a result, the method has richer representation power and hence has better performance on object recognition tasks.

- Compared to deep learning methods, our method benefits from effective hand-crafted features, such as SIFT features, as the input. Each module of our architecture has sound explanation and can be formulated as explicit optimization problems with promising computational performance. The method shows superior performance over the state-of-the-art methods in multiple experiments.

*heyunlong@gatech.edu, Georgia Institute of Technology
†koray@deepmind.com, DeepMind Technologies
‡yunwang@princeton.edu, Princeton University
§aszlam@ccny.cuny.edu ,The City College of New York
¶yanjun@virginia.edu, University of Virginia

2 Background

In this section, we review the technical background of the new framework. In Subsection 2.1, we go over the pipeline of using bag-of-visual-words for object recognition. In Subsection 2.2, we revisit dimensionality reduction methods and concentrate on a low-dimensional embedding method called DRLIM.

2.1 Bag-of-visual-words pipeline for object recognition We now review the bag-of-visual-words pipeline consisting of hand-crafted descriptor computing, bag-of-visual-words representation learning, spatial pyramid pooling and finally a classifier.

The first step of the pipeline is to exact a set of overlapped image patches from each image with fixed patch size, while the spacing between the centers of two adjacent image patches is also fixed. Then a D-dimensional hand-crafted feature descriptor (e.g. 128-dimensional SIFT descriptor) is computed from each image patch. Now let $X^{(i)}$ denote the set of M_i feature descriptors, which are converted from M_i overlapped image patches extracted from the i-th image (e.g. size 300×300), i.e.,

$$X^{(i)} = [x_1^{(i)}, \cdots, x_{M_i}^{(i)}] \in \mathbb{R}^{D \times M_i},$$

where $x_j^{(i)}$ is the feature descriptor of the j-th patch in the i-th image.

Let $X = [X^{(1)}, X^{(2)} \cdots, X^{(N)}] \in \mathbb{R}^{D \times M}$, where $M = M_1 + M_2 + \cdots + M_N$, denote the set of all feature descriptors from all N training images. The second step of the pipeline consists of a dictionary learning process and a bag-of-visual-words representation learning process. In the case of using sparse coding to learn the bag-of-visual-words representation, the two processes can be unified as the following problem.

$$\begin{aligned} (2.1) \qquad & \min_{V,Y} \|X - VY\|_F^2 + \alpha \|Y\|_{1,1} \\ & = \sum_{m=1}^{M} \|x_m - V y_m\|_2^2 + \alpha \|y_m\|_1 \\ & s.t. \|v_k\| \leq 1, \quad \forall k = 1, \cdots, K, \end{aligned}$$

where $V = [v_1, \cdots, v_K] \in \mathbb{R}^{D \times K}$ denote the dictionary of visual-words, and columns of $Y = [y_1, \cdots, y_M] \in \mathbb{R}^{K \times M}$ are the learned sparse codes, and α is the parameter that controls sparsity of the code. We should note, however, other sparse encoding methods such as vector quantization and LLC could be used to learn the sparse representations (see [8] for review and comparisons). Moreover, the dictionary learning process of finding V in (2.1) is often conducted in an online style [23] and then the feature descriptors of the i-th image stored $X^{(i)}$ is encoded as the bag-of-visual-words representation $Y^{(i)} = [y_1^{(i)}, \cdots, y_{M_i}^{(i)}]$ in K-dimensional space ($K >> D$). Intuitively speaking, the components of the bag-of-visual-words representation are less correlated compared to the components of dense descriptors. Therefore, compared to the dense feature descriptors, the high-dimensional sparse representations are more favorable for the classification tasks.

In the third stage of the pipeline, the sparse bag-of-visual-words representations associated with the image patches from each image are pooled over some hierarchical image neighborhoods to obtain a single feature vector for the image. To achieve this, each image is divided into three levels of pooling regions as suggested by the spatial pyramid matching (SPM) technique [20]. The first level of pooling region is the whole image. The second level is consist of 4 pooling regions which are 4 quadrants of the whole image. The third level consist of 16 pool regions which are quadrants of the second level pooling regions. In this way, we obtain 21 overlapped pooling regions. Then for each pooling region, a max-pooling operator is applied to all the sparse codes whose associating image patch center locates in this pooling region, and we obtain a single feature vector for each pooling region. The max-pooling operator maps any number of vectors that have the same dimensionality to a single vector, whose components are the maximum value of the corresponding components in the mapped vectors. Formally, given the descriptors $y_1, \cdots, y_n \in \mathbb{R}^K$ that are in the same pooling region, we calculate

$$(2.2) \qquad y = op_{max}(y_1, \cdots, y_n) := max\{y_1, \cdots, y_n\} \in \mathbb{R}^K,$$

where max is operated component-wisely. Finally, the pooled bag-of-visual-words representations from 21 pooling regions are concatenated to obtain a single feature vector, which is regarded as the representation for the image and linear SVM is then used for training and testing on top of this representation. Since the labels of the training images are not used until the final training of SVM, the whole pipeline of learning the image representation is regarded as an unsupervised method. For the rest of this paper, we focus on the version of the pipeline where the feature (bag-of-visual-words representation) learning part is performed by a sparse coding (2.1) step.

2.2 Dimensionality reduction by learning an invariant mapping Dimensionality reduction is a very common technique in machine learning, which embeds high-dimensional data instances into a low-dimensional space. Popular dimensionality reduction methods include encoders such as NMF, linear transformations

Figure 1: The bag-of-visual-words pipeline.

such as PCA and many other manifold learning techniques. Most of the dimensionality reduction methods are unsupervised methods which learn the transformation to preserve the relationship in the original space/manifold. In this paper, however, we are interested in preserving the spatial smoothness of image patches when learning a low-dimensional embedding. More specifically, if two image patches are largely overlapped, we have the prior knowledge that they are likely to capture the same object even though their representations could be far apart in the original space. Therefore a dimensionality reduction method that is able to make use of the prior knowledge is of our main interest. We now review a method called dimensionality reduction by learning an invariant mapping (DRLIM, see [15]), which is the base model for our new method in Subsection 3.3.

Different from traditional unsupervised dimensionality reduction methods, DRLIM relies not only on a set of training instances $y_1, y_2, \cdots, y_n \in \mathbb{R}^K$, but also on a set of binary labels $\{l_{ij} : (i,j) \in I\}$, where I is the set of index pairs such that $(i,j) \in I$ if the label for the corresponding instance pair (y_i, y_j) is available. The binary label $l_{ij} = 0$ if the pair of training instances y_i and y_j are similar instances, and $l_{ij} = 1$ if y_i and y_j are known to be dissimilar. Notice that the similarity indicated by l_{ij} is usually from extra resource instead of the knowledge that can be learned from data instances $y_1, y_2, \cdots, y_n$ directly. DRLIM learns a parametric mapping

$$A : y \in \mathbb{R}^K \mapsto z \in \mathbb{R}^D,$$

such that the embeddings of similar instances attract each other in the low-dimensional space while the embeddings of dissimilar instances push each other away in the low-dimensional space. In this spirit, the exact loss function of DRLIM is as follows:

$$L(A) = \sum_{(i,j)\in I} (1 - l_{ij}) \frac{1}{2} \|A(y_i) - A(y_j)\|^2 + l_{ij} \frac{1}{2} max(0, \beta - \|A(y_i) - A(y_j)\|)^2, \tag{2.3}$$

where $\beta > 0$ is the parameter for the contrastive loss term which decides the extent to which we want to push the dissimilar pairs apart. Since the parametric mapping A is assumed to be decided by some parameter. DRLIM learns the mapping A by minimizing the loss function in (2.3) with respect to the parameters of

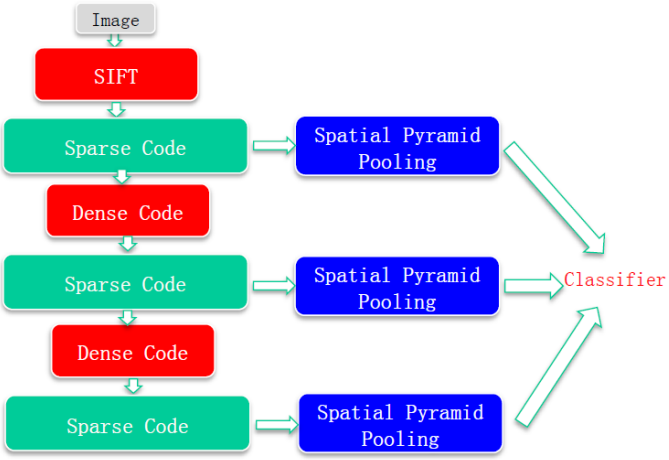

Figure 2: A three-layer deep sparse coding framework. Each of the three layers contains three modules. The first module converts the input (image patches at the first layer and sparse codes at other layers) to dense codes. The second module is a sparse encoder converting the dense codes to sparse codes. The sparse codes are then sent to the next layer, and simultaneously to a spatial pyramid pooling module. The outputs of the spatial pyramid pooling modules can be used for further tasks such as classification.

A mapping. The mapping A can be either linear or nonlinear. For example, we can assume A is a two-layer fully connected neural network and then minimize the loss function (2.3) with respect to the weight. Finally, for any new data instance y_{new}, its low-dimensional embedding is represented by $A(y_{new})$ without knowing its relationship to the training instances.

3 Deep sparse learning framework

3.1 Overview Recent progress in deep learning [2] has shown that the multi-layer architecture of deep learning system, such as that of deep belief networks, is helpful for learning feature hierarchies from data, where different layers of feature extractors are able to learn feature representations of different scopes. This results in more effective representations of data and benefits a lot of further tasks. The rich representation power of deep learning methods motivate us to combine deep learning with the bag-of-visual-words pipeline to achieve better performance on object recognition tasks.

In this section, we introduce a new learning framework, named as deep sparse coding (DeepSC), which is built of multiple layers of sparse coding.

Before we introduce the details of the DeepSC framework, we first identify two difficulties in designing such a multi-layer sparse coding architecture.

- First of all, to build the feature hierarchies from bottom-level features, it is important to take advantage of the spatial information of image patches such that a higher-level feature is a composition of lower-level features. However, this issue is hardly addressed by simply stacking sparse encoders.
- Second, it is well-known (see [26, 13]) that sparse coding is not "smooth", which means a small variation in the original space might lead to a huge difference in the code space. For instance, if two overlapped image patches have similar SIFT descriptors, their associated sparse codes can be very different. If another sparse encoder were applied to the two sparse codes, they would lost the affinity which was available in the SIFT descriptor stage. Therefore, stacking sparse encoders would only make the dimensionality of the feature higher and higher without gaining new informations.

Based on the two observations above, we propose the deep sparse coding (DeepSC) framework as follows. The first layer of DeepSC framework is exactly the same as the bag-of-visual-words pipeline introduced in Subsection 2.1. Then in each of the following layer of the framework, there is a sparse-to-dense module which converts the sparse codes obtained from the last layer to dense codes, which is then followed by a sparse coding module. The output sparse code of the sparse coding module is the input of the next layer. Furthermore, the spatial pyramid pooling step is conducted at every layer such that the sparse codes of current layer are converted to a single feature vector for that layer. Finally, the feature vectors from all layers are concatenated as the input to the classifier. We summarize the DeepSC framework in Figure 2. It is important to emphasis that the whole framework is unsupervised until the final classifier.

The sparse-to-dense module is the key innovation of the DeepSC framework, where a "pooling function" is proposed to tackle the aforementioned two concerns. The pooling function is the composition of a local spatial pooling step and a low-dimensional embedding step, which are introduced in Subsection 3.2 and Subsection 3.3 respectively. On one hand, the local spatial pooling step ensures the higher-level features are learned from a collection of nearby lower-level features and hence cover larger scopes. On the other hand, the low-dimensional embedding process is designed to take into account the spatial affinities between neighboring image patches such that the spatial smoothness information is not lost during the dimension reduction process. As the combination of the two steps, the pooling function fills the gaps between the sparse coding modules, such that the power of sparse coding and spatial pyramid pooling can be fully expressed in a multi-layer fashion.

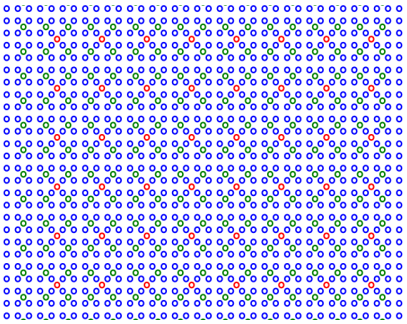

Figure 3: The blue points form the first level sampling grid. The green points form the second level sampling grid. The red points for the third level sampling grid. The local spatial pooling step is performed on the local 4×4 grid.

3.2 Learning the pooling function In this subsection, we introduce the details of designing the local spatial pooling step, which performs as the first part of the pooling function. First of all, we define the pooling function as a map from a set of sparse codes on a sampling grid to a set of dense codes on a new sampling grid. Assume that G is the sampling grid that includes M sampling points on an image, where any two adjacent sampling points have fixed spacing (number of pixels) between them. As described in Subsection 2.1, each sampling point corresponds to the center of an image patch. Let $Y = [y_1, \cdots, y_M] \in \mathbb{R}^{K \times M}$ be the sparse codes on the sampling grid G, where each y_i is associated with a sampling point on G according to its associated image patch. Mathematically, the pooling function is defined as the map:

$$f : (Y, G) \mapsto (Z, G'),$$

where G' is the new sampling grid with M' sampling points and $Z = [z_1, \cdots, z_{M'}] \in \mathbb{R}^{D \times M'}$ stores the D-dimensional dense codes ($D < K$ [1]) associated with the sampling points on the new sampling grid G'.

As the feature representations learned in the new layer are expected to cover larger scopes than those

[1] For simplicity, we let D be the same as the dimensionality of SIFT features.

in the previous layer, we enforce each of the sampling points on the new grid G' to cover a larger area in the image. To achieve this, we take the center of 4×4 neighboring sampling points in G and let it be the new sampling points in G'. By taking the center of every other 4×4 neighboring sampling points, the spacing between neighboring sampling points in G' is twice of that in G. As a result, we map G to a coarser grid G' such that $M' \approx M/4$ (see Figure 3).

Once the new sampling grid G' is determined, we finish the local spatial pooling step by applying the max-pooling operator (defined in (2.2)) to the subsets of M sparse codes $\{y_1, \cdots, y_M\}$ and obtain M' pooled sparse codes associated with the new sampling grid G'. More specifically, let $\bar{y}_i$ denote the pooled sparse codes associated with the i-th sampling point in G', where $i \in \{1, \cdots, M'\}$. We have

$$\bar{y}_i := op_{max}(y_{i_1}, y_{i_2}, \cdots, y_{i_{16}}), \tag{3.4}$$

where $\{i_1, i_2, \cdots, i_{16}\}$ are the indices of the 16 sampling points in G that are closest to the i-th sampling point in G'.

3.3 Dimensionality reduction with spatial information In this subsection, we introduce the details of combining the DRLIM method [15] with the spatial information of image patches to learn a low-dimensional embedding A such that

$$z_i := A(\bar{y}_i). \tag{3.5}$$

As the feature vector is transformed by A to lower-dimensional space, part of its information is discarded while some is preserved. As introduced in Subsection 2.2, DRLIM is trained on a collection of data instance pairs $(\bar{y}_i, \bar{y}_j)$, each of which is associated with a binary label indicating their relationship. Therefore, it provides the option to incorporate prior knowledge in the dimensionality reduction process by determining the binary labels of training pairs based on the prior knowledge.

In the case of object recognition, the prior knowledge that we want to impose on the system is that if an image patch is shifted by a few pixels, it still contains the same object. Therefore, we constructed the collection of training pairs for DRLIM as follows. We extract training pairs such that there always exist overlapped pixels between the two corresponding patches. Let $\bar{y}_i$ and $\bar{y}_j$ be the pooled sparse codes corresponding to two image patches that have overlapped pixels and d_{ij} be the distance (in terms of pixels) between them, which is calculated based on the coordinate of the image patch centers. Given a thresholding σ, we set

$$l_{ij} = \begin{cases} 0 & d_{ij} < \sigma \\ 1 & d_{ij} > \sigma \end{cases} \tag{3.6}$$

Generated this way, $l_{ij} = 0$ indicates the two image patches are mostly overlapped, while $l_{ij} = 1$ indicates that the two image patch are only partially overlapped. This process of generating training pairs ensures that the training of the transformation A is focused on the most difficult pairs. Experiments shows that if we instead take the pooled sparse codes of far-apart image patches as the negative pairs ($l_{ij} = 1$), DRLIM suffers downgrading in performance. The sensitivity of the system to the thresholding parameter σ is demonstrated in Table 7.

Let the linear transformation A be defined by the transformation matrix $W \in \mathbb{R}^{D \times K}$ such that

$$A(\bar{y}_i) = W\bar{y}_i.$$

Then, the loss function with respect to the pair $(\bar{y}_i, \bar{y}_j)$ is

$$\begin{aligned} L_{ij}(W) = (1 - l_{ij})\frac{1}{2}\|W\bar{y}_i - W\bar{y}_j\|^2 \\ + l_{ij} max(0, \beta - \|W\bar{y}_i - W\bar{y}_j\|)^2. \end{aligned} \tag{3.7}$$

Let I be the set of index pairs for training pairs collected from all training images, W is then obtained by minimizing the loss with respect to all training pairs, i.e., solving

$$\begin{aligned} &\min_W \sum_{(i,j)\in I} L_{ij} \\ &s.t. \quad \|w_k\| \le 1, \quad \forall k = 1, \cdots, K. \end{aligned}$$

To accelerate the training of DRLIM, we apply stochastic gradient descent method with mini-batch, which is described as follows. At each iteration, we randomly pick a subset $I_m \subset I$ of size m, and then update using the following stochastic gradient descent step:

$$W \leftarrow W - \frac{\gamma}{m} \sum_{(i,j)\in I_m} \frac{dL_{ij}(W)}{dW}, \tag{3.8}$$

where γ is the step size and the gradient $\frac{dL_{ij}(W)}{dW}$ is defined as

$$\begin{aligned} \frac{dL_{ij}(W)}{dW} &= \mathcal{I}(l_{ij} = 0) D_{ij}(\bar{y}_i - \bar{y}_j)^\top \\ &- \mathcal{I}(l_{ij} = 1, 0 < D_{ij} < \beta) \frac{\beta - \|D_{ij}\|}{\|D_{ij}\|} D_{ij}(\bar{y}_i - \bar{y}_j)^\top, \end{aligned}$$

where $\mathcal{I}(\cdot)$ is the binary indicator function and

$$D_{ij} := W\bar{y}_i - W\bar{y}_j.$$

After the gradient step, each column of W is then projected to the unit L_2 ball.

4 Related Work

One of the drawbacks of single-layer sparse coding BoV system is that it encodes local patches independently, ignoring the spatial neighborhoods structure of the image. Our DeepSC framework uses DRLIM to remedy this issue through enforcing spatial smoothness. A closely related work "Hierarchical Sparse Coding" from [28], presented a two-layer sparse coding scheme, where the first layer encodes the local patches of an image, and the second layer of sparse coding gets the pooling of the first layer codes and models the higher-order dependency of patches in the same local region. Their system achieved 74.0% on Caltech-101 dataset using the first-layer SC dictionary size as 8 and the second-layer SC dictionary as 2048 dictionary size. It is very difficult to extend this two-layer framework to more layers, due to the high dimensionality of sparse codes (the more layers, the higher dimension of dictionary the deeper-layer SC needs). Different from this work, our DeepSC makes use of the low-dimensional projection DRLIM to connect multiple layers of SC together, which enables SC to be applicable on deeper layers.

In the recent decade, most of the literature on multi-stage object recognition system can be seen as variants of the BoV pipeline. For example, HoG features[9] or even the original image patches[8] can be used as the first level image descriptors to replace SIFT features[22]. While most of the previous works use SIFT features of image patches, TFM [18] proposed new locally-invariant feature descriptors that are learned from raw images automatically in an unsupervised fashion. The new descriptors give comparable performance to SIFT on Caltech 101 and better performance than SIFT on MNIST and Tiny Images. The proposed DeepSC utilizes SIFT feature descriptors to describe local image patches. Thanks to the flexible and extendable multiple-layer structure, it should be very easy to extend DeepSC using the raw pixels descriptors as inputs.

The pooling step performed by Spatial pyramid matching (SPM) [20, 27] is regarded a huge extension over bag-of-visual-words method by considering the spatial order of local descriptors and significantly improve the performance over the bag-of-visual-words representation on many databases. Some recent works [17, 12] focus on improving the pooling step by designing the pooling regions more adaptively [17] or by learning a pooling operator instead of using the max-pooling operator. In our DeepSC framework, spatial pyramid pooling with 21 pooling regions and max-pooling operator plays an important role as the third module in every layer. However, we want to point out that the results in our experiment are possible to be further improved if the above alternative techniques are applied.

The recent literature focuses more on refining the "coding" steps in the BoV pipeline. The k-means based algorithm [20] has been shown to be outperformed by sparse-coding [27], which is then outperformed by locality-constrained linear coding (LLC) [26], Laplacian sparse coding [13] and smooth sparse coding [1]. The later three works address the "non-smoothness" problem of sparse coding at the stage of coding. Instead, the DeepSC framework addresses this problem in the stage of dimension reduction. Moreover, compared to DeepSC, the three works mentioned above didn't consider the extension to multi-layer framework so that they are not able to learn feature hierarchies. In the future work, it will be interesting to see if the DeepSC framework can be combined with these recently proposed sparse encoders to achieve better performance.

In addition to the improved coding methods, other works successfully approached the object recognition task using multiple patch sizes. Applying KSVD and batch orthogonal matching pursuit (OMP) as the building blocks recursively at varying layers and scales, the Multipath Hierarchical Matching Pursuit (M-HMP) [3] learns features through multiple paths where different patch size is used in different path. [7] demonstrated that combining several deep convolutional neural networks (DNN) columns into a Multi-column DNN (MCDNN) can achieve near-human performance on MNIST handwriting benchmark, and outperform humans on GTSRB traffic sign dataset. It is worth noticing that DeepSC is different from these two works because the input image patches have only one fixed size. We speculate, however, that the idea of multi-path also works with the DeepSC framework.

5 Experiments

In this section, we evaluate the performance of DeepSC framework for image classification on three data sets: Caltech-101 [10] , Caltech-256 [14] and 15-Scene.

- Caltech-101: this data set [10] contains 9144 images belonging to 101 classes, with about 40 to 800 images per class. Most images of Caltech-101 are with medium resolution, i.e., about 300×300.
- Caltech-256: this data set [14] contains $29,780$ images from 256 categories. The collection has higher intra-class variability and object location variability than Caltech-101. The images are of similar size to Caltech-101.
- 15-Scene: this data set, compiled by several researchers [11, 20, 24], contains a total of 4485 images falling into 15 categories, with the number of images per category ranging from 200 to 400. The

categories include living room, bedroom, kitchen, highway, mountain, street and et al.

For each data set, the average per-class recognition accuracy is reported. Each reported number is the average of 10 repeated evaluations with random selected training and testing images. For each image, following [4], we sample 16×16 image patches with 4-pixel spacing and use 128 dimensional SIFT feature as the basic dense feature descriptors. The final step of classification is performed using one-vs-all SVM through LibSVM toolkit [5]. The parameters of DRLIM and the parameter to control sparsity in the sparse coding are selected layer by layer through cross-validation. In the following, we present a comprehensive set of experimental results, and discuss the influence of each of the parameters independently. In the rest of this paper, DeepSC-2 indicates two-layer DeepSC system; DeepSC-3 represents three-layer DeepSC system, and SPM-SC means the one layer baseline, i.e. the BoV pipeline with sparse coding plus spatial pyramid pooling.

5.1 Effects of Number of DeepSC Layers As shown in Figure 2, the DeepSC framework utilizes multiple-layers of feature abstraction to get a better representation for images. Here we first check the effect of varying the number of layers utilized in our framework. Table 1 shows the average per-class recognition accuracy on three data sets when all using 1024 as dictionary size. The number of training images per class for the three data sets is set as 30 for Caltech-101, 60 for Caltech-256, and 100 for 15-Scene respectively. The second row shows the results when we have only one layer of the sparse coding, while the third row and the fourth row describe the results when we have two layers in DeepSC or three layers in DeepSC. Clearly the multi-layer structured DeepSC framework has superior performance on all three data sets compared to the single-layer SPM-SC system. Moreover, the classification accuracy improves as the number of layers increases.

5.2 Effects of SC Dictionary Size We examine how performance of the proposed DeepSC framework changes when varying the dictionary size of the sparse coding. On each of the three data sets, we consider three settings where the dimension of the sparse codes K is $1024, 2048$ and 4096. The number of training images per class for these experiments is set as 30 for Caltech-101, 60 for Caltech-256, and 100 for 15-Scene respectively. We report the results for the three data sets in Table 2, Table 3 and Table 4 respectively. Clearly, when increasing the dictionary size of sparse coding K from 1024 to 4096, the accuracy of the system improves for all three data sets. We can observe

	Caltech-101	Caltech-256	15-Scene
SPM-SC	75.66±0.59	43.04±0.34	80.83±0.59
DeepSC-2	77.41±1.06	46.02±0.57	82.57±0.72
DeepSC-3	78.24±0.76	47.00±0.45	82.71±0.68

Table 1: Average per-class recognition accuracy (shown as percentage) on three data sets using 1024 as dictionary size. The number of training images per class for the three data sets are 30 for Caltech-101, 60 for Caltech-256, and 100 for 15-Scene respectively. DeepSC-2/3: two/three layers of deep sparse coding. SPM-SC: the normal BoV pipeline with one layer of sparse coding plus spatial pyramid pooling.

that the performance of DeepSC is always improved with more layers, while in the case of $K = 4096$ the performance boost in term of accuracy is not so significant. This probably is due to that the parameter space in this case is already very large for the limited training data size. Another observation we made from Table 2, Table 3 and Table 4 is that DeepSC-2 (K=1024) always performs better than SPM-SC (K=2048), and DeepSC-2 (K=2048) always performs better than SPM-SC (K=4096). These two comparisons demonstrate that simply increasing the dimension of sparse codes doesn't give the same performance boost as increasing the number of layers, and therefore DeepSC framework indeed benefits from the feature hierarchies learned from the image.

Caltech-101	$K=1024$	$K=2048$	$K=4096$
SPM-SC	75.66±0.59	76.34±0.58	77.21±0.7
DeepSC-2	77.41±1.06	78.27±0.6	78.3±0.9
DeepSC-3	78.24±0.76	78.43±0.72	78.41±0.74

Table 2: Effect of dictionary size used in sparse coding on recognition accuracy (shown as percentage). data set: Caltech-101; number of training images per class: 30

Caltech-256	$K=1024$	$K=2048$	$K=4096$
SPM-SC	43.04±0.34	45.66±0.53	47.8±0.63
DeepSC-2	46.02±0.57	48.04±0.44	49.29±0.50
DeepSC-3	47.0±0.45	48.85±0.42	49.91±0.39

Table 3: Effect of dictionary size used in sparse coding on recognition accuracy (shown as percentage). data set: Caltech-256; number of training images per class: 60

15-Scene	$K = 1024$	$K = 2048$	$K = 4096$
SPM-SC	80.83±0.59	82.11±0.61	82.88±0.82
DeepSC-2	82.57±0.72	83.58±0.71	83.76±0.72
DeepSC-3	82.71±0.68	83.58±0.61	83.8±0.73

Table 4: Effect of varying sparse coding dictionary size on recognition accuracy (shown as percentage). data set: 15-Scene; number of training images per class: 100

5.3 Effects of Varying Training Set Size Furthermore, we check the performance change when varying the number of training images per class on two Caltech data sets. Here we fix the dimension of the sparse codes K as 2048. On Caltech-101, we compare two cases: randomly select 15 or 30 images per category respectively as training images and test on the rest. On Caltech-256, we randomly select 60, 30 and 15 images per category respectively as training images and test on the rest. Table 5 and Table 6 show that with the smaller set of training images, DeepSC framework still continues to improve the accuracy with more layers.

Caltech-101	30	15
SPM-SC	76.34±0.58	69.94±0.61
DeepSC-2	78.27±0.6	71.53±0.53
DeepSC-3	78.43±0.72	71.86±0.55

Table 5: Effect of varying training set size on averaged recognition accuracy. data set: Caltech-101; Dictionary Size: 2048

Caltech-256	60	30	15
SPM-SC	45.66±0.53	39.86±0.24	33.44±0.15
DeepSC-2	48.04±0.44	41.86±0.28	35.10±0.19
DeepSC-3	48.80±0.42	42.33±0.29	35.28±0.27

Table 6: Effect of varying training set size on averaged recognition accuracy. data set: Caltech-256; Dictionary Size: 2048

5.4 Effects of varying parameters of DRLIM In table 7, we report the performance variations when tuning the parameters for DRLIM. The parameter σ is the threshold for selecting positive and negative training pairs (see (3.6)) and the parameter β in the hinge loss (see (3.7)) of DRLIM model is for controlling penalization for negative pairs. We can see that it is important to choose the proper thresholding parameter σ such that the transformation learned by DRLIM can differentiate mostly overlapped image pairs and partially overlapped image pairs.

$\sigma \setminus \beta$	1	2	3	4	5	6
8	76.5	77.41	77.07	76.71	76.24	75.81
16	74.93	76.55	76.87	76.97	76.43	75.83
24	73.95	75.43	76.18	76.42	76.53	76.45

Table 7: The effect of tuning DRLIM parameters on recognition accuracy for DeepSC-2. data set: Caltech-101; dictionary size: 1024; the number of training images per class: 30.

	Caltech-101	Caltech-256	15-Scene
ScSPM	73.2±0.54	40.14±0.91	80.28±0.93
SSC	77.54±2.59	–	84.53±2.57
DeepSC-3	78.24±0.76	47.04±0.45	82.71±0.68

Table 8: Comparison of results with other image recognition algorithms: ScSPM[27], LLC[26], and SSC[1]. Dictionary size $K = 1024$. Number of training images are 30, 60, and 100 for Caltech-101, Caltech-256 and 15-Scene respectively.

5.5 Comparison with other methods We then compare our results with other algorithms in Table 8. The most direct baselines [2] for DeepSC to compare are the sparse coding plus SPM framework (ScSPM) [27], LLC[26], and SSC[1]. Table 8 shows the comparison of our DeepSC versus the ScSPM and SSC. We can see that our results are comparable to SSC, with a bit lower accuracy on the 15-Scene data (the std of SSC is much higher than ours). For the LLC method proposed from [26], it reported to achieve 73.44% for Caltech-101 when using $K = 2048$ and 47.68% when using $K = 4096$. Our DeepSC-3 has achieved 78.43% for Caltech-101 when using $K = 2048$ and 49.91% when using $K = 4096$. Overall our system achieves the state-of-the-art performance on all the three data sets.

6 Conclusion

Feature learning has been the core problem of many machine learning problems such that object recognition, natural language processing and speech recognition. In this paper, we propose a new method, namely Deep Sparse Coding (DeepSC), that extends sparse coding to a deep feature learning framework. The multi-layer framework connects the sparse-encoders from different levels by a pooling function, which consists of a local spatial pooling step and a dimensionality reduction step. This new method is able to learn sparse representations of the images at different levels of abstraction and of

[2] We are also aware of that some works achieve very high accuracy based on adaptive pooling step [12] or multiple-path system that utilizes image patches of multiple sizes [3].

different spatial scopes. We test DeepSC on multiple visual object recognition data sets and achieves the state-of-the-art performance. In the future, we plan to improve DeepSP by extending the classic sparse coding step we used currently to the newly proposed Smooth Sparse Coding [1], which considers spatial smoothness in the "coding step". It is also promising to extend the current framework on more types of data, e.g. audio.

References

[1] K. Balasubramanian, K. Yu, and G. Lebanon. Smooth sparse coding via marginal regression for learning sparse representations. In *ICML*, 2013.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *arXiv preprint arXiv:1206.5538*, 2012.

[3] L. Bo, X. Ren, and D. Fox. Multipath sparse coding using hierarchical matching pursuit. CVPR, 2013.

[4] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, pages 2559–2566. IEEE, 2010.

[5] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

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

[7] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. CVPR, 2012.

[8] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, volume 8, page 10, 2011.

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR 2005*, volume 1, pages 886–893. IEEE, 2005.

[10] 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 *CVPR, Workshop on Generative-Model Based Vision.*, 2004.

[11] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

[12] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric p-norm feature pooling for image classification. In *CVPR*, pages 2609–2704. IEEE, 2011.

[13] S. Gao, I. W. Tsang, L.-T. Chia, and P. Zhao. Local features are not lonely–laplacian sparse coding for image classification. In *CVPR*, pages 3555–3561. IEEE, 2010.

[14] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[15] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, volume 2, pages 1735–1742. IEEE, 2006.

[16] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[17] Y. Jia, C. Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *CVPR, 2012*, pages 3370–3377. IEEE, 2012.

[18] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *CVPR*, 2009.

[19] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[20] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178. IEEE, 2006.

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

[22] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[23] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696. ACM, 2009.

[24] A. Oliva and A. Torraba. Modeling the shape of the scene: A holistic representation of the spatial envelop. In *IJCV*, 2001.

[25] B. A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[26] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367. IEEE, 2010.

[27] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801. IEEE, 2009.

[28] K. Yu, Y. Lin, R. Fergus, and Y. LeCun. Learning image representations from the pixel level via hierachical sparse coding. In *CVPR*, 2011.