arXiv:1701.08349v3 [cs.CV] 23 May 2017

Supervised Deep Sparse Coding Networks

Xiaoxia Sun[1], **Nasser M. Nasrabadi**[2], **and Trac D. Tran**[1]
The Johns Hopkins University[1]
West Virginia University[2]
{xxsun, trac}@jhu.edu, nasser.nasrabadi@mail.wvu.edu

Abstract

In this paper, we describe the deep sparse coding network (SCN), a novel deep network that encodes intermediate representations with nonnegative sparse coding. The SCN is built upon a number of cascading *bottleneck modules*, where each module consists of two sparse coding layers with relatively wide and slim dictionaries that are specialized to produce high dimensional discriminative features and low dimensional representations for clustering, respectively. During training, both the dictionaries and regularization parameters are optimized with an end-to-end supervised learning algorithm based on multilevel optimization. Effectiveness of an SCN with seven bottleneck modules [1] is verified on several popular benchmark datasets [2]. Remarkably, with few parameters to learn, our SCN achieves 5.81% and 19.93% classification error rate on CIFAR-10 and CIFAR-100, respectively.

1 Introduction

Representing a signal with a linear combination of few dictionary atoms, sparse coding has shown promising performance on a range of computer vision tasks including image classification and object detection [1, 2, 3]. Even when given only a small amount of training samples, sparse coding models can become exceptionally resilient against severely corrupted or noisy data. However, when the noise in the data is an expression of the natural variation of objects, such as those caused by changes in orientation, the linear representation of sparse coding is not able to catpure these nonlinear variations and contribute to the degradation of the performance. As such, sparse coding models exhibit disappointing performance on large datasets where variability is broad and anomalies are common.

Conversely, deep neural networks thrive on large volume of data. Their success derives from an ability to distill the core essence of a subject from abundant diverse examples [4], which has encouraged researchers to try and augment the learning capacity of traditionally shallow sparse coding methods by adding layers [5, 6, 7]. Multilayer sparse coding networks are expected to combine the best of both strategies. For instance, the imperative for sparse codes to adequately reconstruct an input signal ameliorates information degeneracy issues within deep architectures. Furthermore, with parsimoneous representations, sparse coding networks can lead to more intuitive intepretation of the learned features. From the view point of neural network, sparse coding enjoys a much stronger 'explain away' effect [8] compared to the feedforward network and is closely related to recurrent network through unfolding the sparse recovery algorithm [5, 9]. To date, however, endeavors to marry the two techniques have not achieved significant improvements over their individual counterparts [6, 7].

In this paper, we present a novel deep sparse coding network based on nonnegative sparse coding and multilevel optimization. The goal of our work is to efficiently extend the conventional sparse coding

[1]Consisting 14 sparse coding layers.

[2]Our codes and models are available on https://github.com/XiaoxiaSun/supervised-deep-sparse-coding-networks.

to multilayer architectures in order to expand its learning capacity. We present a bottleneck module, the core building block of our SCN, which employs two specialized sparse coding layers: one is equiped with a relatively wide dictionary, while the other one has a slim dictionary to significantly reduce the number of learnable parameters without incurring the penalty on the performance of the network. The width of the sparse coding layer is defined as the number of dictionary atoms. We also propose to optimize both the dictionaries and regularization parameters of SCN using an end-to-end supervised learning algorithm based on multilevel optimization.

We demonstrate the effectiveness of the proposed SCN on four benchmark data sets, including CIFAR-10, CIFAR-100, STL-10 and MNIST. The proposed network exhibits competitive performance using only a few learnable parameters. Remarkably, our 15-layer [3] SCN achieve higher accuracy on CIFAR-10 and CIFAR-100 compared to a 110-layer [10] and 1001-layer [11] residual network, respectively. We emphasize here that in this paper we do not aim at pursuing highest accuracy, but rather effectively extending sparse coding to multilayer architectures to achieve competitive performance.

2 Related Works

The proposed SCN is mostly related but substantially differs from the following approaches:

Deep neural networks with wide shape. Wide networks [12, 13, 14] have been recently proposed to exploit the high dimensional latent features, demonstrating competitive performance compared to slim deep network [10, 11, 15] while using much fewer layers. The wide networks consist of a much larger number of learnable parameters compared to their slim counterparts, therefore requiring much stronger regularization to reduce overfitting. In this paper, we focus on regularizating the wide SCN via dimensionality reduction and clustering using the nonnegative sparse coding-based approach.

Dimensionality reduction and clustering in deep neural networks. Bottleneck shaped neural network [10, 14] applies dimensionality reduction in order to reduce the overfitting of residual network. In contrast to neural network, dimensionality reduction with nonnegative sparse coding is equivalent to clustering [16] and therefore the low dimensional hidden features act as weighted cluster indicators which is discussed in Section 3.2. Alternate approach related to our work is the deep semi-nonnegative matrix factorization (semi-NMF) [17] that trains a hierarchical network with the reconstruction loss. Our approach differs from the aforementioned works since we simoultaneously learn high dimensional discriminative representations and low dimensional clustered features in a single network architecture with end-to-end supervised learning.

Multilayer sparse coding. Most common approach for multilayer sparse coding is to train the reconstructive dictionaries of each layer in a greedily layer-wise fashion [7, 18], where the nonlinearity is usually enforced with a ReLU layer. An alternative approach is to unfold and approximate the sparse coding process with deep neural networks [5, 9], in which the sparse coding parameters are trained end-to-end by minimizing reconstruction loss. Our approach differs in that we extend the conventional nonnegative sparse coding to a multilayer architecture without applying unfolding during training, therefore the model size would not increase with the number of iteration. Moreover, our approach allows a much broader choice among the off-the-shelf sparse recovery algorithms without changing the network architecture. In addition, the proposed network also shares a high level motivation with the stacked autoencoder [19] and CNN-based model with auxiliary reconstruction loss [20, 21], which trains the network in an unsupervised, semi-supervised or supervised fashion by manually balancing the discriminative and reconstruction loss. In contrast we employ conventional sparse coding instead of neural network to encode latent features and train the network supervisedly in an end-to-end fashion.

Supervised dictionary learning. Supervised dictionary learning strengthens the discriminative power of the sparse codes by exploiting the labeled samples. Thorough study on task-driven dictionary learning algorithms for various applications is reported in [22]. Applying fixed point differntiation and bilevel optimization, a supervised dictionary learning scheme for the shallow sparse coding model is proposed in [23, 24]. In this paper, we generalize the single-layer supervised dictionary learning to multilayer network based on multilevel optimization.

[3] Including 14 sparse coding layers and one linear classifier.

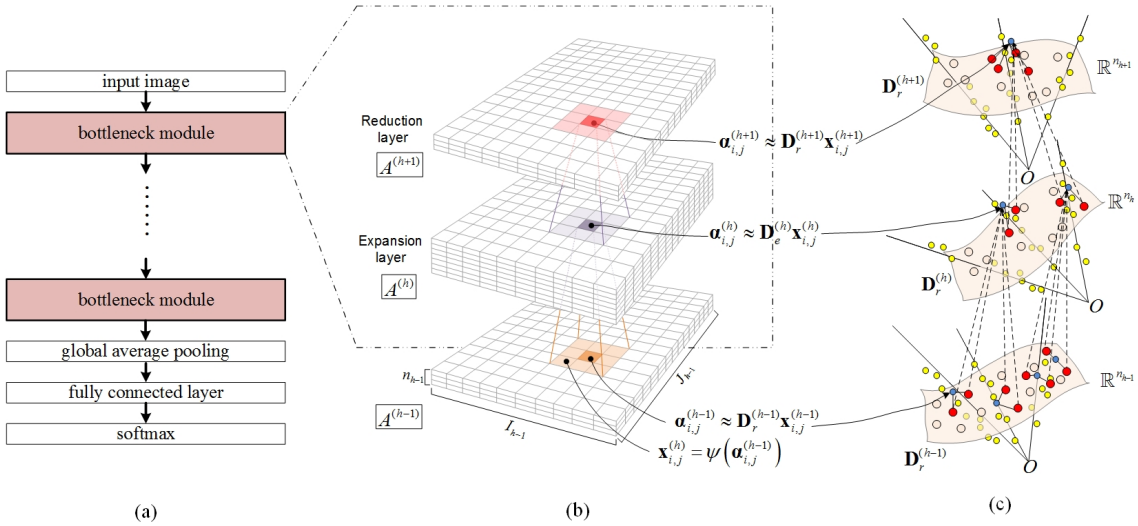


Figure 1: Architecture of our multilayer sparse coding network: (a) The proposed network is constructed by repeatedly stacking multiple bottleneck modules. The network does not contain any pooling operation and subsampling is conducted with a stride of 2. (b) Bottleneck module consists of one expansion layer and one reduction layer, which is used to expand or reduce the dimensionality of the local features of the previous layer, respectively. (c) Intepretation of SCN. Red and hollow circles on the manifolds are the active and inactive atoms, respectively. Yellow circles are represents all the local features of a hidden layer and blue circles denotes the neighboring local features centered at (i, j).

3 Extension of Sparse Coding to Deep Architectures

In this section, we introduce the SCN model by extending the conventional sparse coding to deep architectures. We start with the illustration of inference with nonnegative sparse coding which can be applied to every sparse coding layer in SCN. We then introduce the bottleneck module, where the sparse coding layers are specialized to encourage discriminative or clustered representations. Throughout this section, we assume that the dictionaries in SCN are given and we describe the dictionary learning algorithm later in Section 4. The architecture of the SCN is illustrated in Fig. 1.

3.1 Inference with Nonnegative Sparse Coding

We now introduce a general formulation of sparse coding layer for SCN. Let the representation of the layer h in SCN be a 3D-tensor $\mathcal{A}^{(h)} \in \mathbb{R}^{n_h \times I_h \times J_h}$, $h \in \{0, \ldots, H\}$ and denote each local feature vector at (i, j) of layer h as $\boldsymbol{\alpha}_{i,j}^{(h)} \triangleq \mathcal{A}_{:,i,j}^{(h)} \in \mathbb{R}^{n_h}$, where n_h, I_h and J_h are the number of channels, height and width of the layer representation. For instance, $\boldsymbol{\alpha}_{i,j}^{(0)}$ of a color image represents a 3-channel pixel of red, green and blue. In deeper layers where $h > 0$, $\boldsymbol{\alpha}_{i,j}^{(h)}$ represent a local sparse code. To recover the local sparse code $\boldsymbol{\alpha}_{i,j}^{(h)}$, we construct an intermediate local feature $\mathbf{x}_{i,j}^{(h)} = \psi(\boldsymbol{\alpha}_{i,j}^{(h-1)}) \in \mathbb{R}^{m_h}$ by concatenating all the neighboring features centered at (i, j) within a window of size $k_{h-1} \times k_{h-1}$ from the previous layer $h-1$, For illustrative purpose, we assume the neighboring window is square. $\psi(\cdot)$ denotes the concatenation operation and $m_h = n_{h-1}k_{h-1}^2$. We constrain the sparse codes to be nonnegative in order to introduce nonlinearity to the deep network. Given a dictionary $\mathbf{D}^{(h)} \in \mathbb{R}^{m_h \times n_h}$ of layer h, the nonnegative sparse code is recovered by solving the following constrained elastic net problem:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha} > \mathbf{0}} F(\boldsymbol{\alpha}) \triangleq \arg\min_{\boldsymbol{\alpha} > \mathbf{0}} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda_1\|\boldsymbol{\alpha}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}\|_2^2, \tag{1}$$

where we have omitted the coordinate and layer indices for simplicity. $\|\boldsymbol{\alpha}\|_1 = \sum_{n=1}^{N} |\alpha_n|$ is the ℓ_1-norm and $\lambda_1, \lambda_2 > 0$ are the regularization parameters. Importance of the parameter λ_2 is to stabilize the training procedure [22]. *In this paper, we directly solve (1) using conventional sparse recovery algorithm for inference instead of applying unfolding on sparse coding process with deep*

neural network [5, 9]. Number of sparse recovery algorithms such as LISTA [5, 9], FISTA [25] and LARS [26] can efficiently solve problem 1. In this paper, we adopt FISTA mainly for the ease of coding in GPUs. The nonnegativity is enforced by using nonnegative soft-thresholding during the optimization. For the purpose of clarity, sparse recovery algorithm for solving problem 1 is shown in Appendix A.

3.2 Bottleneck Module with Specialized Sparse Coding Layer

We now describe the *bottleneck module* which is the core building block of our SCN. Each bottleneck module consists of a cascade of two specialized sparse coding layers, which are referred to as *expansion layer* and *reduction layer*. The expansion layer is equiped with a relatively wide dictionary in order to reach a fine-grained partition of the input feature space, whereas the reduction layer has a relatively slim dictionary which focuses more on dimensionality reduction and clustering in order to extract more abstract representations. We denote the dictionaries in expansion and reduction layer as $\mathbf{D}_e \in \mathbb{R}^{m_e \times n_e}$ and $\mathbf{D}_c \in \mathbb{R}^{m_r \times n_r}$, respectively, where $n_e \gg n_r$. We note that the order of expansion and reduction layer in a bottleneck module does not matter much in the multilayer environment. For illustrative purpose, we sequentially employ expansion layer and reduction layer in a single bottleneck module. We illustrate the two specialized sparse coding layers and describe the motivations of proposing the bottleneck module in more details:

Expansion layer focuses on partitioning feature space. Nonnegative sparse coding functions as a robust and stable partition of the input feature space [23], where the 'resolution' of the partition depends on the dictionary width. With a relatively wide or even overcomplete dictionary, we are able to achieve a high resolution fine-grained partition of the feature space and therefore recover highly discriminative sparse codes. Behavior of sparse coding with a wide or even overcomplete dictionary in single layer environment has been thoroughly exploited through number of studies [2, 22, 23].

Reduction layer focuses on clustering features. Reduction layer is designed to produce abstract compact sparse codes using a much narrower dictionary compared to that of the expansion layer. Output of the expansion layer has a high dimensionality and is intensely sparse, which poses two issues in SCN. First, neighboring features from similar subspaces could be represented by highly distinctive sparse codes when the dictionary is overcomplete. Second, high dimensional outputs require the dictionaries in deeper layers to be excessively wide in order to maintain the redundancy and overcompleteness, which makes the network computationally infeasible. In this paper, inspired by the SqueezeNet [15], we propose to compress the high dimensional latent sparse codes using a reduction layer, which can be interpreted as conducting clustering on the high dimensional sparse codes.

Nonnegative sparse coding with slim dictionary functions as clustering, which can be illustrated based on semi-NMF [16]. Several inspirational works [9, 27, 28] illustrate the relations between sparse coding, dictionary learning and matrix factorization: In a reduction layer, when the given dictionary is slim, the nonnegative sparse coding is equivalent with sparsity-regularized semi-NMF algorithm, which is strongly related to the K-means clustering. Hence, the slim dictionary atoms in reduction layer can be interpreted as the cluster centroids of the high dimensional inputs, whereas the corresponding low dimensional nonnegative sparse code is the weighted cluster indicator.

3.3 Interpreting Sparse Coding Network as Deep Subspace Learning

For illustrative purposes, we consider the simplified case where all the local features of layer h lie on a union of disjoint subspaces, i.e., every pair of these subspaces only intersect at origin. As is shown in Fig. 1 (c), each atom of the learned dictionary is the cluster center of a large number of local features in $\mathbb{R}^{n_h}$ and every nonnegative sparse code $\boldsymbol{\alpha}_{i,j}^{(h)}$ in layer h describes how strong it is connected to a certain cluster center. We note that in the case of supervised learning, the distance between each local feature and their related cluster centers, i.e., dictionary atoms, are not only measured by the reconstructive loss but also described by the discriminative loss as shown in (2).

In the case of SCN, large number of subspaces in $\mathbb{R}^{n_h}$ are related to each other through the local sparse code $\boldsymbol{\alpha}_{i,j}^{(h)}$, which itself lies on another subspace in $\mathbb{R}^{n_{h+1}}$ of the deeper layer $h+1$. Similarly, as the network goes deeper, each point in $\mathbb{R}^{n_k}$ of layer k relates to a more complex union of subspaces in $\mathbb{R}^{n_j}$ of the shallower layer j, where $k \gg j$, i.e., local sparse codes in deeper layers are more

expressive compared to those from shallower layers. Driven by the discriminative loss function, the local features of two different classes are gradually mapped to different subspaces of each layer and eventually become linearly separable with respect to the hyperplane defined by the classifier.

3.4 Sparse Coding Network Structure

Our SCN is designed to stack multiple bottleneck modules in order to perform dimensionality expansion and reduction repeatedly. Batch normalization layer [29] is added after each sparse coding layer in order to obtain a faster convergence. The last bottleneck module lies on top of a global average pooling layer, which is followed by a fully connected layer which functions as the linear classifier.

4 End-to-end Supervised Learning for Sparse Coding Networks

In previous section, we described the architecture of SCN with the bottleneck module. In this section, we develop the end-to-end supervised learning algorithm for training both the dictionaries and regularization parameters in the deep SCN based on multilevel optimization.

4.1 Problem Formulation with Multilevel Optimization

Without loss of generality, we consider a prediction task for binary class given a set of training pairs $\{\mathcal{A}_s^{(0)}, y_s\}_{s=1}^S$, where $y_s \in \{0,1\}$ is the label for the image sample $\mathcal{A}_s^{(0)}$. Given an SCN with H sparse coding layers, our goal is to fit the network prediction to the label through minimizing a smooth and convex loss function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with respect to the network parameters, including dictionaries, regularization parameters and the linear classifier. Suppose the network maps the input image $\mathcal{A}_s^{(0)}$ to the corresponding label y_s with a nonlinear function $f : \mathbb{R}^{n_0} \times \mathbb{R}^{I_0} \times \mathbb{R}^{J_0} \to \mathbb{R}$, the optimization procedure of SCN is formulated as an empirical risk minimization problem based on multilevel optimization:

$$
\begin{aligned}
&\min_{\theta} \frac{1}{S}\sum_{s=1}^{S} L(y_s, f(\mathcal{A}_s^{(h)}, \mathbf{w})) + \frac{\mu}{2}R(\theta),\\
&s.t. \quad \boldsymbol{\alpha}_s^{(H)^*} = \arg\min_{\boldsymbol{\alpha}_s^{(H)} \geq \mathbf{0}} F(\mathbf{D}^{(H)}, \lambda^{(H)}, \mathbf{x}^{(H)_s}, \boldsymbol{\alpha}_s^{(H)}),\\
&\qquad \vdots\\
&\quad s.t. \quad \boldsymbol{\alpha}_s^{(1)^*} = \arg\min_{\boldsymbol{\alpha}^{(1)} \geq \mathbf{0}} F(\mathbf{D}^{(1)}, \lambda^{(1)}, \mathbf{x}_s^{(1)}, \boldsymbol{\alpha}_s^{(1)}),\\
&\qquad s.t. \quad \lambda^{(h)} > 0,\ \mathbf{x}_s^{(h)} = \psi(\boldsymbol{\alpha}_s^{(h-1)^*}), \quad \forall h = 1, \ldots, H,
\end{aligned} \tag{2}
$$

where $\theta = \{\mathbf{D}^{(h)}, \lambda^{(h)}\}_{h=1}^H$ is the learnable parameter set including both dictionaris and regularization parameters. $\mathbf{w} \in \mathbb{R}^{n_H}$ is the linear classifier. In this paper, we adaptively optimize the regularization parameters at each layer, which has a similar effect as training the bias in deep neural networks [9]. The motivation for training regularization parameters is that cross-validation becomes formidable as the network becomes deeper.

To prevent the ℓ_2-norm of dictionary to be arbitrarily large and recovering trivial sparse codes, we introduce regularizer $R(\mathbf{D}) \triangleq \|\mathbf{D}\|_F^2$, or usually referred to as weight decay in deep neural network, on the dictionary to reduce the overfitting. We note that constraining every dictionary atom with $\|\mathbf{d}_j\|_2 \leq c$, where $c > 0$ is a chosen constant, is the most common choice for regularizing dictionary atoms in a single layer model. However, during experiment, we found that such constraint is too stringent for the network to converge due to the projection on descent gradient.

4.2 Updating Dictionary and Regularization Parameter

Every sparse code $\boldsymbol{\alpha}$ is parameterized by the dictionary and regularization parameters, it is therefore natural to solve the multilevel optimization problem (2) with gradient descent method based on error backpropagation [30]. The derivation of the updating rules is based on the fixed point differentiation

[22, 23, 24]. Due to the page limitation, we state the first order optimality condition of the nonnegative elastic net, which is the core building block of the derivation, and we leave the rest parts in Appendix B.

Lemma 1 (Optimality conditions of nonnegative elastic net.) *The optimal sparse code $\boldsymbol{\alpha}^*$ of (1) solves the following system:*

$$\mathbf{d}_j^\top(\mathbf{D}\boldsymbol{\alpha}^* - \mathbf{x}) + \lambda_2\alpha_j^* = -\lambda_1, \text{ if } \alpha_j^* > 0 \tag{3}$$

$$\mathbf{d}_j^\top(\mathbf{D}\boldsymbol{\alpha}^* - \mathbf{x}) + \lambda_2\alpha_j^* \geq -\lambda_1, \text{ otherwise.} \tag{4}$$

The nonnegative part of the sparse code $\boldsymbol{\alpha}^*$ can be described as $\boldsymbol{\alpha}_\Lambda^* = (\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda + \lambda_2\mathbf{I})^{-1}(\mathbf{D}_\Lambda^\top\mathbf{x} - \lambda_1\mathbf{1}_\Lambda)$, where $\mathbf{1}_\Lambda \in \mathbb{R}^{|\Lambda|}$ is an all one vector, Λ is the active set of $\boldsymbol{\alpha}^*$ and $|\Lambda|$ is the cardinality of the active set Λ.

Proof. Let $\partial\|\boldsymbol{\alpha}\|_1$ be the subgradient of $\|\boldsymbol{\alpha}\|_1$. Since $\boldsymbol{\alpha}^*$ is a the optimum of (1), $\forall j \in [N], \exists \mathbf{z} \in \partial\|\boldsymbol{\alpha}\|_1$, such that $\boldsymbol{\alpha}^*$ solves the nonlinear Karush–Kuhn–Tucker (KKT) system $(\mathbf{d}_j^\top(\mathbf{D}\boldsymbol{\alpha}^* - \mathbf{x}) + \lambda_2\alpha_j^* + \lambda_1 z_j) \cdot \alpha_j = 0$ and z_j is the jth element of $\mathbf{z}$. Followed by the classical result of Elastic Net [22] and the complementary slackness of KKT condition, when $\alpha_j = 0$, we have $z_j \leq 1$ and reach (3). Representation of $\boldsymbol{\alpha}_\Lambda^*$ can be achieved by applying algebraic simplification on (3) for all atoms j. When $\alpha_j > 0$, we have $z_j = 1$ and reach (4). □

5 Experiments

We conduct extensive experiments on CIFAR-10, CIFAR-100, STL-10 and MNIST. We demonstrate that the proposed SCN exhibits competitive performance while using much smaller number of parameters and layers compared to numerous deep neural network approaches. Notably, a 14-layer SCN model exceeds the performance of a 164-layer and 1001-layer deep residual network, respectively. The proposed SCN is implemented using Matlab with C++ and GPU backend based on the framework of MatConvNet [31]. Experiments are conducted on a server with 4 Nvidia Tesla P40 GPUs.

Model. The architecture of the network is similar to the ResNet [10]. The SCN is configured with 7 bottleneck module which includes 14 sparse coding layers that are divided into 3 sections, i.e., $(16, 16K) \times 3 - (32, 32K) \times 2 - (64, 64K) \times 2$, where each $(M, MK) \times P$ denotes a bottleneck module that is repeatedly stacked for P times, the output dimensions of reduction and expansion layers are M and MK. For CIFAR-10 and CIFAR-100, we exploit the performance of the network with different width, i.e., $K \in \{1, 2, 4\}$. For MNIST and STL-10, we set $K = 4$. We denote an SCN with width of MK as SCN-K. The window size k_h at each sparse coding layer has a size of 3×3. We use the same network configurations for CIFAR-10, CIFAR-100 and STL-10, and for MNIST we set the number of filters at the first layer to be 8 due to the simplicity of the dataset. Following the architecture of ResNet, we apply spatial subsampling with a factor of 2 at the last two bottleneck modules. The last sparse coding layer is followed by one global spatial average pooling layer [32] and one fully connected layer which is the linear classifier. Batch normalization is added after each sparse coding layer to facilitate the convergence.

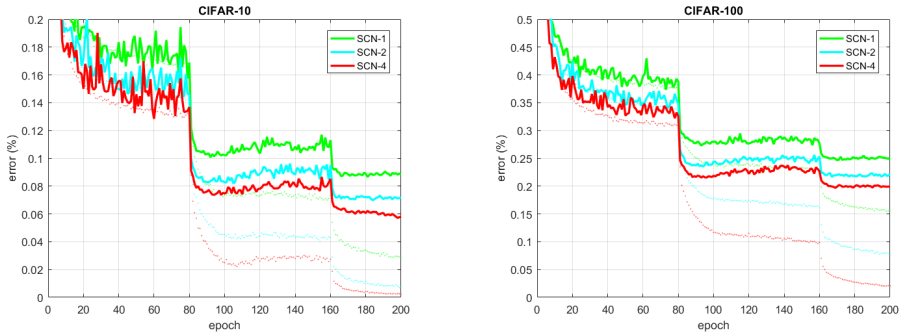
Training. At the training stage, we apply data augmentation and preprocessing for all datasets except for MNIST with random horizontal flipping and random translation. The image is translated up to 4 pixels in each direction for CIFAR-10 and CIFAR-100, and up to 12 pixels for STL-10. Images in the same batch share the same augmentation parameters. Both training and testing images are preprocessed with per-pixel-mean subtraction, which is a common procedure for preprocessing these datasets [10, 18, 32, 33]. We use a minibatch size of 128 for MNIST, CIFAR-10, and CIFAR-100. For STL-10, we use a batch size of 16 in order to have more iterations per epoch on the small training set. For all dataset, the initial learning rate is set to 0.1 and SCN is trained with a total of 200 epochs. For CIFAR-10, CIFAR-100 and STL-10, we follow a similar learning rate schedule with [11], where the learning rate decreases twice at 80 and 160 epochs by a factor of 10. For MNIST, the network is trained with 25 epochs, where the learning rate decreases at 10 and 20 epochs by a factor of 10. The weight decay is set to 0.0005 for all the dataset with cross-validation.

Baseline comparison methods. We compare our proposed SCN with numerous multilayer sparse coding-based approaches, including multilayer sparsity regularized coding (OMP) [18] and nonnegative multilayer sparse coding (NOMP) [7]. We also compare with supervised convolutional kernel

networks (SCKN) [34] and scattering network (ScatNet) [35]. For deep neural network baseline, we mainly compare with residual network (ResNet) [10, 11], wide residual network (WRN) [12] and swapout networks (SwapOut) [13].

Computation time. For CIFAR-10 and CIFAR-100, training SCN-4 model with 200 epochs takes about 26 hours and inference of all the $10,000$ testing images takes about 9 seconds. Training and inference with SCN-4 on STL-10 dataset takes about 21 hours and 65 seconds, respectively. For MNIST, training and testing takes about 3 hours and 7 seconds, respectively.

5.1 CIFAR-10 and CIFAR-100



(a) (b)

Figure 2: Learning curve of SCN on CIFAR-10 and CIFAR-100. Dotted and solid lines denote the learning curves of training and testing stage, respectively.

Our most extensive experiment is conducted on the CIFAR-10 and CIFAR-100 datasets, which consists of $60,000$ color images that are evenly splitted into 10 classes. The database is split into $50,000$ training samples and $10,000$ test samples. Each class has $5,000$ training images and $1,000$ testing images with size 32×32. CIFAR-100 has exactly the same set of images as CIFAR-10 but are split into 10 times more classes, therefore each class has much fewer training samples compared with CIFAR-10, making it a more challenging dataset for the task of classification.

Table 1: Classification Error (%) on CIFAR-10 and CIFAR-100.

Method	# Params	# Layers	CIFAR-10	CIFAR-100
SCKN [34]	10.50M	10	10.20	-
OMP [18]	0.70M	2	18.50	-
PCANet [36]	0.28B	3	21.33	-
NOMP [7]	1.09B	4	18.60	39.92
NiN [32]	-	-	8.81	35.68
DSN [33]	1.34M	7	7.97	36.54
WRN [12]	36.5M	28	4.00	19.25
ResNet-110 [10]	0.85M	110	6.41	27.22
ResNet-1001 v2 [11]	10.2M	1001	4.92	27.21
ResNext-29 [14]	**68.10M**	**29**	**3.58**	**17.31**
SwapOut-20 [13]	1.10M	20	5.68	25.86
SwapOut-32 [13]	7.43M	32	4.76	22.72
SCN-1	0.17M	15	8.86	25.08
SCN-2	0.35M	15	7.18	22.17
SCN-4	**0.69M**	**15**	**5.81**	**19.93**

SCN outperforms previous deep sparse coding models. As shown in Table 1, for CIFAR-10, previous sparse coding-based models of OMP [18] and NOMP [7] networks reported a classification error of 18.50% and 18.60%, respectively. With the aid of bottleneck module and end-to-end supervised learning algorithm, the proposed SCN demonstrates an improvement of 13% on accuracy while only using 0.69M parameters.

Wider dictionary in expansion layer significantly improves performance. Table 1 and Fig. 2 show that the classification performance of SCN increases with the width of the dictionary of the expansion layer, gaining 3% and 6% on CIFAR-10 and CIFAR-100, respectively. In the case when

$K = 4$, our 15-layer SCN exhibits competitive performance compared to 20-layer SwapOut network on CIFAR-10 while using twice fewer parameters.

SCN with bottleneck module uses parameters efficiently. From Table 1, we can see that the proposed SCN uses fewest learnable parameters compared to *all* baseline models and contain fewest number of layers compared to all deep neural network-based baselines. Compared to the state-of-the-art approach of ResNext, our model uses almost $100\times$ fewer parameters and almost one half of layers while still reaching a competitive performance. Moreover, the SCN-4 outperforms other approaches with similar model size such as ResNet-1001 on CIFAR-100.

SCN exhibits strongly competitive performance compared to baselines models. The proposed SCN achieves classification error of 5.81% and 19.93% on CIFAR-10 and CIFAR-100, respectively, which is shown in Table 1. Consider the small size of our model, the performance of SCN is rather strong and competitive. In addition, our model has not yet exploited numerous useful tools in deep neural network such as dropout, shortcut connection and swapout, we strongly believe that the performance of SCN can be further improved.

5.2 STL-10

The dataset STL-10 is originally designed for unsupervised learning, which contains a total number of $5,000$ labeled training images and $8,000$ testing images with size of 96×96. For this dataset, we follow the evaluation protocol used in [37]. The training samples in STL-10 is highly limited and SCN is supposed to generate more competitive performance due to the regularization from the bottleneck modules. We directly apply the 14 sparse coding layer model on STL-10 and replace the 8×8 average pooling with 24×24. We compare our network with the baseline of DeepTEN and previous state-of-the-art approach [20]. From table 2, we can see SCN with bottleneck module (SCN-4) outperforms Deep-TEN under fair comparison by a large margin of 7%. SCN also exceeds previous state-of-the-art performance [20] by almost 9%.

Table 2: Classification Accuracy (%) on STL-10.

Method	#Params	#Layers	Accuracy
SWWAE [20]	10.50M	10	74.33
Deep-TEN	25.60M	50	76.29
SCN-4	**0.69M**	**15**	**83.11**

5.3 MNIST

The MNIST dataset consists of $70,000$ images of digits, of which $60,000$ are the training set and the remaining $10,000$ are the test set. Each digit is centered and normalized to a 28×28 field. The classification error on this dataset is reported in Table 3. The purpose of evaluating this relatively simple dataset is to compare with previous sparse coding-based models, as most of which are evaluated on MNIST. With a deep architecture, the SCN achieves a classification error of 0.36%, which outperforms all previous sparse coding-based models.

Table 3: Classification Error (%) on MNIST.

Method	#Params	#Layers	Accuracy
CKN [38]		2	0.39
ScatNet [35]	-	3	0.43
PCANet [36]	-	3	0.62
S-SC [23]	-	1	0.84
TDDL [22]	-	1	0.54
SCN-4	**0.69M**	**15**	**0.36**

6 Conclusion and Discussion

In this paper, we have developed a novel multilayer sparse coding network based on nonnegative sparse coding and multilevel optimization. We propose applying bottleneck module to dramatically reduce the overfitting and computational costs of SCN. Moreover, we also show that our SCN is compatible with other powerful deep learning tools such as batch normalization. We have demonstrated that our network produces results competitive with deep neural networks but uses significantly fewer parameters and layers.

References

[1] A Wagner, J. Wright, A Ganesh, Z. Zhou, H. Mobahi, and Y. Ma. Toward a practical face recognition system: Robust alignment and illumination by sparse representation. *IEEE TPAMI*, 34(2):372–386, 2012.

[2] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE TPAMI*, 31(2):210–227, 2009.

[3] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[4] A. Krizhevsky, Ilya S., and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[5] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, 2010.

[6] Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi. Unsupervised feature learning by deep sparse coding. In *ICDM*, 2014.

[7] T. Lin and H. T. Kung. Stable and efficient representation learning with nonnegativity constraints. In *ICML*, 2014.

[8] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI*, 35(8):1798–1828, 2013.

[9] T. Moreau and J. Bruna. Understanding trainable sparse coding via matrix factorization. In *ICLR*, 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[12] S. Zagoruyko and N. Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[13] S. Singh, D. Hoiem, and D. A. Forsyth. Swapout: Learning an ensemble of deep architectures. In *NIPS*, 2016.

[14] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[15] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[16] C. H. Q. Ding, T. Li, and M. I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE TPAMI*, 32(1):45–55, 2010.

[17] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller. A deep semi-NMF model for learning hidden representations. In *ICML*, 2014.

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

[19] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. In *JMLR*, 2010.

[20] J Zhao, M. Mathieu, R. Goroshin, and Y. LeCun. Stacked what-where auto-encoders. *CoRR*, abs/1506.02351, 2015.

[21] Y. Zhang, K. Lee, and H. Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. *CoRR*, abs/1606.06582, 2016.

[22] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE TPAMI*, 34(4):791–804, 2012.

[23] J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. *in CVPR*, Jun. 2010.

[24] J. Yang, Z. Wang, Z. Lin, X. Shu, and T. Huang. Bilevel sparse coding for coupled feature spaces. In *CVPR*, 2012.

[25] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM JIS*, 2(1):183–202, 2009.

[26] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *AS*, 32:407–499, 2004.

[27] J. Mairal, F. R. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. In *JMLR*, 2010.

[28] P. O. Hoyer. Non-negative sparse coding. In *NNSP*, 2002.

[29] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[30] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *AOR*, 153(1):235–256, 2007.

[31] A. Vedaldi and K. Lenc. Matconvnet - convolutional neural networks for matlab. In *ICM*, 2015.

[32] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[33] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[34] J. Mairal. End-to-end kernel learning with supervised convolutional kernel networks. *CoRR*, abs/1605.06265, 2016.

[35] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE TPAMI*, 35(8):1872–1886, 2013.

[36] T. H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. PCANet: A simple deep learning baseline for image classification? *IEEE TIP*, 24(12):5017–5032, 2015.

[37] H. Zhang, J. Xue, and K. Dana. Deep ten: Texture encoding network. *arXiv preprint arXiv:1612.02844*, 2016.

[38] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *NIPS*, 2014.

A Solving Constrained Elastic Net using FISTA

For the purpose of clarification, we describe the nonnegative FISTA in Algorithm 1, which is used for inference during training and testing. We denote $(\mathbf{A})_+$ as the element-wise nonnegative thresholding on $\mathbf{A}$.

Algorithm 1 FISTA for nonnegative Elastic Net

Require: Dictionary $\mathbf{D} \in \mathbb{R}^{M \times N}$, κ is the largest eigenvalue of $(\mathbf{D}^\top \mathbf{D} + \lambda_2 \mathbf{I})$, precompute $\mathbf{A} = \mathbf{I} - \frac{1}{\kappa}(\mathbf{D}^\top \mathbf{D} + \lambda_2 \mathbf{I})$, $\mathbf{b} = \frac{1}{\kappa}(\mathbf{D}^\top \mathbf{x} - \lambda_1)$, iterator $t = 0$, $\boldsymbol{\alpha}_t = \mathbf{0} \in \mathbb{R}^N$, $\boldsymbol{\gamma}_t = \mathbf{0} \in \mathbb{R}^N$, $s_0 = 1$.

while stopping criterion not satisfied **do**
  $\boldsymbol{\alpha}_{t+1} \leftarrow (\mathbf{A}\boldsymbol{\gamma}_t + \mathbf{b})_+$.
  $s_{t+1} \leftarrow (1 + (1 + 4s_t^2))/2$.
  $\boldsymbol{\gamma}_{t+1} \leftarrow \boldsymbol{\alpha}_{t+1} + (s_t - 1)(\boldsymbol{\alpha}_{t+1} - \mathbf{x}_t)/s_{t+1}$.
  $t \leftarrow t + 1$.
end while
return Nonnegative sparse code $\boldsymbol{\alpha}_t$.

B Dictionary and parameter update

We now derive the backpropagation rule for solving problem 2. In this paper, we derive the updating rule for the case of holistic sparse coding since extension to the convolutional local sparse coding is trivial. Optimizing dictionaries and regulerization parameters with stochastic gradient descent is shown in Algorithm 2. We start by differentiating the empirical loss function with respect to every element of the dictionaries and regularization parameters:

$$\frac{\partial L}{\partial d_{jk}^{(h)}} = \frac{\partial L}{\partial \boldsymbol{\alpha}^{(H)}} \cdot \left(\prod_{i=H}^{h+1} \frac{\partial \boldsymbol{\alpha}^{(i)}}{\partial \boldsymbol{\alpha}^{(i-1)}} \right) \cdot \frac{\partial \boldsymbol{\alpha}^{(h)}}{\partial d_{jk}^{(h)}}, \tag{5}$$

$$\frac{\partial L}{\partial \lambda_1^{(h)}} = \frac{\partial L}{\partial \boldsymbol{\alpha}^{(H)}} \cdot \left(\prod_{i=H}^{h+1} \frac{\partial \boldsymbol{\alpha}^{(i)}}{\partial \boldsymbol{\alpha}^{(i-1)}} \right) \cdot \frac{\partial \boldsymbol{\alpha}^{(h)}}{\partial \lambda_1^{(h)}}, \quad \text{s.t.} \quad \lambda_1^{(h)} > 0, \tag{6}$$

where $d_{jk}^{(h)}$ is the (j,k)-element of the dictionary $\mathbf{D}^{(h)}$. To solve for (5) and (6), we need to derive $\partial \boldsymbol{\alpha}^{(h)} / \partial \boldsymbol{\alpha}^{(h-1)}$, $\partial \boldsymbol{\alpha}^{(h)} / \partial d_{jk}^{(h)}$ and $\partial \boldsymbol{\alpha}^{(h)} / \partial \lambda_1^{(h)}$. We employ fixed point differentiation for deriving the required derivatives, which is based on the previous works of dictionary learning for one-layer sparse coding model [22, 23]. Let $\boldsymbol{\alpha} \in \mathbb{R}^N$ [4] be the optimal point of Lasso problem, it then satisfies the optimality condition based on (3) and for all $\boldsymbol{\alpha}_\Lambda > 0$:

$$(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})\boldsymbol{\alpha}_\Lambda - \mathbf{D}_\Lambda^\top \mathbf{x} + \lambda_1 \mathbf{1}_{|\Lambda|} = \mathbf{0}, \tag{7}$$

where we have omitted the layer indices for simplicity. Λ denotes the active set of the sparse code $\boldsymbol{\alpha}$ and $|\Lambda|$ is the cardinality of the active set. $\mathbf{D}_\Lambda \in \mathbb{R}^{m \times |\Lambda|}$ is the subset of dictionary consists of the active atoms. $\mathbf{I}_{|\Lambda|} \in \mathbb{R}^{|\Lambda| \times |\Lambda|}$ is identity matrix and $\mathbf{1}_{|\Lambda|} \in \mathbb{R}^{|\Lambda|}$ is an all one vector.

Differentiation of $\partial L / \partial \mathbf{D}$. We first derive the differentiation $\partial \boldsymbol{\alpha} / \partial d_{jk}$ for a single dictionary element d_{jk}. The inactive atoms are not updated since the desired gradient on which $\alpha_j = 0$ is not well defined [22, 23] and $\partial \boldsymbol{\alpha}_{\Lambda^c} / \partial d_{jk} = \mathbf{0}$, where Λ^c is the complementary of Λ. Differentiate both sides of (7) with respect to d_{jk} for all $j \in \Lambda$:

$$\left(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|}\right) \frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial d_{jk}} + \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial d_{jk}} \boldsymbol{\alpha}_\Lambda - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial d_{jk}} = \mathbf{0}, \tag{8}$$

which is equivalent with

$$\frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial d_{jk}} = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1} \left(\frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial d_{jk}} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial d_{jk}} \boldsymbol{\alpha}_\Lambda \right). \tag{9}$$

[4] We have omitted the superscript '*' for simplicity.

Algorithm 2 Dictionary and parameter update for deep sparse coding network

Require: $\{\mathbf{D}^{(h)}\}_{h=1}^{H}$ dictionary initialized with Gaussian random noise, initial $\{\lambda_1^{(h)}\}_{h=1}^{H}$. $\{\mathbf{x}_i, y_i\}$ training pairs. $t = 1$.

while stopping criterion not satisfied **do**
  Randomly choose a sample pair $\{\mathbf{x}_i, y_i\}$ and let $\boldsymbol{\alpha}^{(0)} = \psi(\mathbf{x}_i)$.
  for layer $h = 1$ to H **do**
    $\boldsymbol{\alpha}^{(h)^*} \leftarrow \arg\min_{\boldsymbol{\alpha}^{(h)} > \mathbf{0}} \frac{1}{2}\|\mathbf{x}^{(h)} - \mathbf{D}^{(h)}\boldsymbol{\alpha}^{(h)}\|_2^2 + \lambda_1^{(h)}\|\boldsymbol{\alpha}^{(h)}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}^{(h)}\|_2^2$.
    where $\mathbf{x}^{(h)} = \psi(\boldsymbol{\alpha}^{(h-1)})$.
  end for
  for layer $h = H$ down to 1 **do**
    Update the dictionary/regularization parameters with a gradient descent/projection step

$$\mathbf{D}^{(h)} \leftarrow \mathbf{D}^{(h)} - \rho_t(\partial L/\partial \mathbf{D}^{(h)} + \mu \mathbf{D}^{(h)}),$$
$$\lambda_1^{(h)} \leftarrow \left(\lambda_1^{(h)} - \rho_t(\partial L/\partial \lambda^{(h)} + \mu \lambda_1^{(h)})\right)_+,$$

    where ρ_t is the learning rate at time t.
  end for
  $t \leftarrow t + 1$.
end while
return $\{\mathbf{D}^{(h)}, \lambda^{(h)}\}_{h=1}^{H}$.

We reach the updating rule for a single dictionary element:

$$\frac{\partial L}{\partial d_{jk}} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1} \left(\frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial d_{jk}} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial d_{jk}} \boldsymbol{\alpha}_\Lambda\right). \tag{10}$$

Stacking all elements $\partial L/\partial d_{jk}$ into $\partial L/\partial \mathbf{D}$ and applying algebraic simplification:

$$\frac{\partial L}{\partial \mathbf{D}} = -\mathbf{D}\boldsymbol{\gamma}\boldsymbol{\alpha}^\top + (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})\,\boldsymbol{\gamma}^\top, \tag{11}$$

where $\boldsymbol{\gamma}_\Lambda = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1} \cdot \partial L/\partial \boldsymbol{\alpha}_\Lambda$ and $\boldsymbol{\gamma}_{\Lambda^c} = \mathbf{0}$. Due to the sparsity constraint, only few atoms are activated in each layer and $|\Lambda|$ is small enough for efficiently implementation (17) on modern GPUs.

Differentiation of $\partial L/\partial \lambda$. Differentiating both sides of Eq. (7) with respect to λ_1:

$$\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda \frac{\partial \boldsymbol{\alpha}}{\partial \lambda_1} = -1, \tag{12}$$

and we reach at

$$\frac{\partial L}{\partial \lambda_1} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot -(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1} = -\boldsymbol{\gamma}. \tag{13}$$

Differentiation of $\partial L/\partial \mathbf{x}$. The gradient of sparse code $\boldsymbol{\alpha}$ with respect to each input signal element $\mathbf{x}$ can be reached by differentiating both sides of (7) with respect to x_i:

$$(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|}) \frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial x_i} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i} = \mathbf{0}, \tag{14}$$

where x_i is the ith element of $\mathbf{x}$. (14) is equivalent with

$$\frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial x_i} = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1} \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i}. \tag{15}$$

Therefore we have

$$\frac{\partial L}{\partial x_i} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1} \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i}, \tag{16}$$

which can be further simplified as

$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{D}\boldsymbol{\gamma}. \tag{17}$$