Supervised Deep Sparse Coding Networks for Image Classification

Xiaoxia Sun, Nasser M. Nasrabadi, *Fellow, IEEE*, and Trac D. Tran, *Fellow, IEEE*

***Abstract*— In this paper, we propose a novel deep sparse coding network (SCN) capable of efficiently adapting its own regularization parameters for a given application. The network is trained end-to-end with a supervised task-driven learning algorithm via error backpropagation. During training, the network learns both the dictionaries and the regularization parameters of each sparse coding layer so that the reconstructive dictionaries are smoothly transformed into increasingly discriminative representations. In addition, the adaptive regularization also offers the network more flexibility to adjust sparsity levels. Furthermore, we have devised a sparse coding layer utilizing a "skinny" dictionary. Integral to computational efficiency, these skinny dictionaries compress the high-dimensional sparse codes into lower dimensional structures. The adaptivity and discriminability of our 15-layer SCN are demonstrated on six benchmark datasets, namely Cifar-10, Cifar-100, STL-10, SVHN, MNIST, and ImageNet, most of which are considered difficult for sparse coding models. Experimental results show that our architecture overwhelmingly outperforms traditional one-layer sparse coding architectures while using much fewer parameters. Moreover, our multilayer architecture exploits the benefits of depth with sparse coding's characteristic ability to operate on smaller datasets. In such data-constrained scenarios, our technique demonstrates a highly competitive performance compared with the deep neural networks.**

***Index Terms*— Image classification, sparse representation, dictionary learning, image analysis, image recognition.**

I. Introduction

SPARSE coding has shown promising performance on a range of computer vision tasks including image classification and target detection [42], [43], [51], [54], [60], [62], [62]. Even when given only a small amount of training samples, sparse coding models can become exceptionally resilient against severely corrupted or noisy data. Consequently, sparse coding is well suited to real-life image recognition tasks in which images are often degraded by sensor static or when objects in the image are occluded. However, when the noise in the data is actually an expression of the natural variation of objects, such as those caused by changes in illumination or orientation, the linear representation of sparse coding becomes a liability [51], [61]. 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 bountiful data. Their success derives from an ability to distill the core essence of a subject from abundant diverse examples [15], [18], [27], [46], [58]. This feat has encouraged researchers to try and augment the learning capacity of traditionally shallow sparse coding methods by adding layers [16], [20], [33]. Theoretically, 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 [6] ameliorates information degeneracy issues within deep architectures [17], [21]. Furthermore, multilayer sparse coding networks demand less training data as compared to deep neural networks. To date, however, endeavors to marry the two techniques have not achieved significant improvements over their individual counterparts [20], [33].

The realization of a successful multilayer sparse coding architecture is obstructed by three critical challenges:

- Efficiently learning dictionaries with sufficient discriminative power.
- Avoiding the growth of overly fat dictionaries.
- Calibrating large quantities of regularization parameters.

Supervised dictionary learning with labeled data provides an opportunity to overcome the first challenge. However, the difficulty lies in computing the gradient with respect to each dictionary element. As covered in Section II-B, there has been inspiring breakthroughs in adapting supervised dictionary learning algorithms for use in shallow sparse coding frameworks [35], [61]. We attempt to build on past achievements by training a multilayer sparse coding network using an end-to-end supervised dictionary learning algorithm.

The second challenge arises during the sparse recovery procedure. The dictionary must grow fat with reference data if it is to perform a satisfactory reconstruction of the input signal from a sparse code. In a multilayer environment, dictionaries deeper in the network bear a greater burden, for they must convey crucial information with increasing austerity. This is particularly problematic for unsupervised dictionary learning. The unsupervised learning algorithm cannot judge what information to retain or discard based on reconstructive feedback. As the dictionaries grow more fatter, the sparse codes become further attenuated. Processing such structures is computationally prohibitive. We apply supervised dictionary learning and signal compression algorithms to address this issue.

Manuscript received October 19, 2017; revised December 9, 2018 and April 7, 2019; accepted June 27, 2019. Date of publication July 17, 2019; date of current version September 23, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Gang Hua. *(Corresponding author: Xiaoxia Sun.)*

X. Sun was with Johns Hopkins University, Baltimore, MD 21218 USA. He is now with Apple Inc., Cupertino, CA 95014 USA (e-mail: xsun9@jhu.edu).

N. M. Nasrabadi is with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506 USA (e-mail: nasser.nasrabadi@mail.wvu.edu).

T. D. Tran is with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: trac@jhu.edu).

Digital Object Identifier 10.1109/TIP.2019.2928121

Inspired by the Network in Network [32] and SqueezeNet [24] architectures, we propose a *dimension reduction layer* that balances discriminative power with reconstructive potential. In contrast to the fat dictionary, the reduction layer uses a much skinnier dictionary for lossy compression of the high-dimensional sparse codes while also introducing an additional nonlinearity to the network.

The third obstruction is inflicted by the large parameter space of the multilayer sparse coding network. Traditionally, the sparsity level in a sparse coding model is chosen manually by cross-validation and remains fixed throughout training. As the network gains layers, the manual selection of regularization parameters quickly becomes daunting. Hence, we propose automatically adapting the sparsity level via *task-driven regularization*.

To summarize, this paper makes the following contributions to sparse coding networks:

- Reduction of sparse code dimensionality by employing 'skinny' dictionaries to create *reduction layers*.
- Dynamic adaptation of ℓ_1 regularization parameters with *task-driven regularization*.
- Supervised, end-to-end training of a multilayer sparse coding network with the aforementioned features.
- The code for training and testing our SCN is available online.[1]

In Section II, we briefly review the works related to multilayer sparse coding, supervised dictionary learning and dimensionality reduction. In Section III, we elaborate on our network design and adaptive regularization technique. We develop and discuss an end-to-end supervised training procedure for SCN in Section IV. In order to clearly perceive the efficiency of supervised learning, we do not apply any unsupervised learning schemes to pretrain the dictionary. In Section V, we evaluate our multilayer sparse coding network on six benchmark datasets, including Cifar-10, Cifar-100, STL-10, SVHN, MNIST and ImageNet. The first four datasets are considered to be highly challenging for sparse coding. Of particular interest is the Cifar-100 which poses formidable challenges to sparse coding. In our evaluation, we show our network to decisively outperform shallow sparse coding architectures. Moreover, we demonstrate our network attains highly competitive results with state-of-the-art models such as deep residual learning [18] in terms of both classification accuracy and the model size.

II. Related Work

A. Deep Sparse Coding Network

Sparse coding with overcomplete dictionary was first proposed in [1]. To augment the learning capacity of sparse coding model, several researchers have attempted to extend the one-layer sparse coding model to multilayer hierarchical architectures. Early approach is applied on contour detection in [23] by sparsely representing activations of complex cells hiearchically. Maire *et al.* [38] train a two-layer sparse coding model using unsupervised dictionary learning for the purpose of contour detection and semantic labeling. Zhou *et al.* [68] employ sparse coding model with a multilayer architecture to generate the linear features for image classification. Similarly, Zhang *et al.* [65] construct multilayer sparse coding network by repetitively stacking sparse coding layer, max pooling layer and contrast normalization layer and train the dictionaries using unsupervised learning. Lin and Kung [33] enforce nonnegativity constraints on orthogonal matching pursuit (OMP) to improve the stability of the sparse activations for a three layer sparse coding network. Yu *et al.* [62] have developed an unsupervised dictionary learning algorithm for jointly training the dictionaries in a two-layer sparse coding network. To deal with the dimensionality explosion of the hidden sparse codes, He *et al.* [20] propose to compress the sparse codes into low-dimensional dense features. Multipath sparse coding [3] combines a collection of hierarchical sparse codes to capture various aspects of discriminative structures for image classification. Wang *et al.* [53] have enforced the sparsity priors on deep network for the purpose of image super-resolution.

From the perspective of dictionary learning, most common approach for training multilayer sparse coding network is to optimize the reconstructive dictionaries of each layer in a greedily layer-wise fashion [8], [33], 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 [16], [39], in which the sparse coding parameters are trained end-to-end by minimizing reconstruction loss. The strategy of unfolding the sparse recovery algorithm is also exploited and developed for training the deep sparse coding networks in an end-to-end fashion [52], [57]. For instance, the deep ℓ_0 encoder [52] and the maximal sparsity networks [57] unrolls the iterative sparse recovery algorithm of ℓ_0 pursuit into an equivalent deep neural network. As such, the training of the dictionaries in sparse coding is reformulated into an equivalent deep neural network training problem, which can be efficiently optimized in an end-to-end fashion via by error backpropagation. These approaches demonstrate promising performance and provide an efficient way for training sparse coding dictionaries by taking advantage of modern GPU architecture. Our approach differs in two ways: First, each sparse coding layer of the proposed networks recovers the sparse codes by solving ℓ_1-minimization problem instead of ℓ_0-minimization problem, which enables us to train the regularization parameters via error backpropagation. Second, we propose to employ a dimension reduction layer before each sparse coding layer in order to avoid the explosion of hidden feature dimensionality and to improve the scalability of the proposed network.

In addition, the proposed network also shares a high level motivation with the stacked autoencoder [49] and CNN-based model with auxiliary reconstruction loss [66], [67], 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.

[1]https://github.com/XiaoxiaSun/supervised-deep-sparse-coding-networks

B. Supervised Dictionary Learning for One-Layer Sparse Coding Model

Supervised dictionary learning strengthens the discriminative power of the sparse codes by exploiting the labeled samples. Due to the nonsmoothness of the ℓ_1-regularizer, computing the gradient with respect to the dictionary is a tricky task. Overcomplete independent component analysis [28] is proposed to orthogonalize the dictionary and approximate the sparse coding with a linear function such that the differentiation of the implicit sparse coding function can be avoided. Fast approximation of sparse coding is proposed in [16] to train the dictionary of each layer in a greedy, unsupervised fashion and initialize a corresponding multilayer neural network with the pretrained sparse coding dictionaries. Bradley and Bagnell [4] propose to directly compute the gradient of the dictionary by switching the ℓ_1 regularizor with the smoothed Kullback-Leibler divergence. Applying fixed point differentiation and error backpropagation, a supervised dictionary learning scheme for the shallow sparse coding model is proposed in [61]. Thorough study on task-driven dictionary learning algorithms with various applications are carried out in [35]. In this paper, we generalize the single-layer supervised dictionary learning to multilayer network based on multilevel optimization.

C. Dimensionality Reduction and Clustering in Deep Neural Networks

Bottleneck shaped neural network [18], [56] 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 [10] and therefore the low dimensional hidden features act as weighted cluster indicators which is discussed in Section III-B. Alternate approach related to our work is the deep semi-nonnegative matrix factorization (semi-NMF) [47] that trains a hierarchical network with the reconstruction loss. Our approach differs from the aforementioned works since we simultaneously learn high dimensional discriminative representations and low dimensional clustered features in a single network architecture with end-to-end supervised learning.

D. Adaptive Regularization

In sparse coding, by adapting the sparsity level we can achieve a better approximation of the underlying model for a given training data with lower estimation bias. The adaptive Lasso is proposed in [69] and has been proved to satisfy the oracle property [14]. Do *et al.* [11] propose to substitute the sparsity level of orthogonal matching pursuit (OMP) with a predefined halting criterion. In low-level feature representation, a nonparametric method based on expectation minimization algorithm [41] is proposed to automatically adjust the sparsity level for the soft thresholding operator. In the case of image deblurring and superresolution, the regularization parameters are proposed to be estimated by assuming the distribution of sparse codes follow a zero-mean Laplacian distribution [12]. To be noted, all these methods are carried out for the purpose of low-level feature extraction and are based on shallow structures with unsupervised learning, while we optimize the regularization parameters using end-to-end supervised learning.

III. Multilayer Sparse Coding networks

In this section, we first describe the inference of SCN, which is based on nonnegative sparse coding. We then introduce and discuss our proposed *bottleneck* module for improving the performance of the network, which consists a cascade of an expansion layer followed by a reduction layer.

A. Inference With Nonnegative Sparse Coding

We now introduce a general formulation of sparse coding layer for SCN as shown in Fig. 1. 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:

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

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 [35]. 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 [16], [39]. Number of sparse recovery algorithms such as Learned Iterative Shrinkage and Thresholding Algorithm (LISTA) [16], [39], Fast ISTA (FISTA) [2] and Least Angle Regression (LARS) [13] 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.

B. Constructing Multilayer Architecture With Bottleneck Modules

We formulate a generalized, multilayer sparse coding architecture as illustrated Fig. 1. Following (1), we denote the

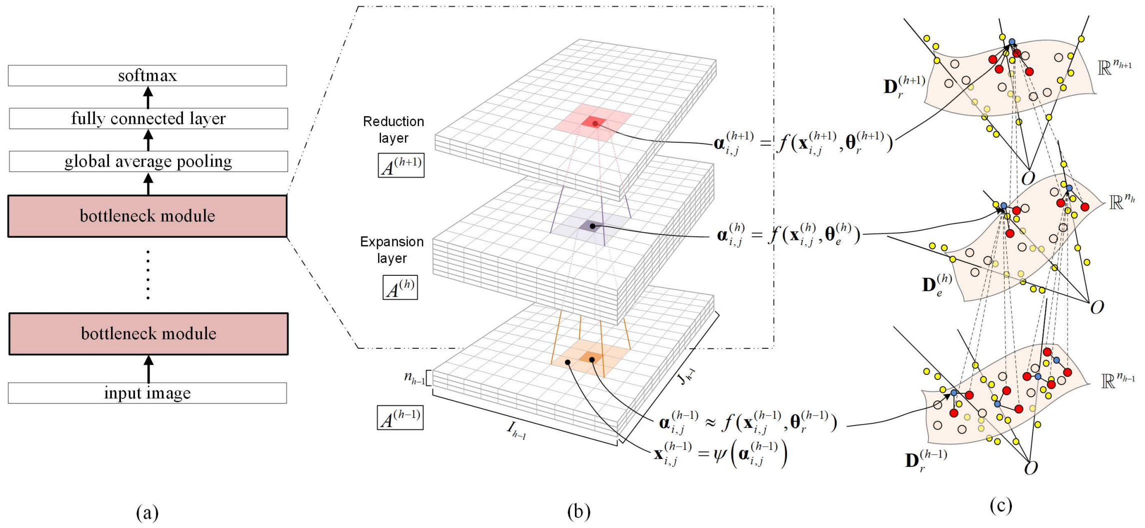

softmax
fully connected layer
global average pooling
bottleneck module
bottleneck module
input image
Reduction layer
$A^{(h+1)}$
Expansion layer
$A^{(h)}$
$A^{(h-1)}$
n_{h-1}
I_{h-1}
J_{h-1}
$\boldsymbol{\alpha}_{i,j}^{(h+1)} = f(\mathbf{x}_{i,j}^{(h+1)}, \boldsymbol{\theta}_r^{(h+1)})$
$\boldsymbol{\alpha}_{i,j}^{(h)} = f(\mathbf{x}_{i,j}^{(h)}, \boldsymbol{\theta}_e^{(h)})$
$\boldsymbol{\alpha}_{i,j}^{(h-1)} \approx f(\mathbf{x}_{i,j}^{(h-1)}, \boldsymbol{\theta}_r^{(h-1)})$
$\mathbf{x}_{i,j}^{(h-1)} = \psi\left(\boldsymbol{\alpha}_{i,j}^{(h-1)}\right)$
$\mathbf{D}_r^{(h+1)}$
$\mathbb{R}^{n_{h+1}}$
$\mathbf{D}_e^{(h)}$
$\mathbb{R}^{n_h}$
$\mathbf{D}_r^{(h-1)}$
$\mathbb{R}^{n_{h-1}}$
O
O
O
(a)
(b)
(c)

Fig. 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) Interpretation of SCN. Red and hollow circles on the manifolds are the active and inactive atoms, respectively. Yellow circles represent all the local features of a hidden layer and blue circles denotes the neighboring local features centered at (i, j).

nonnegative sparse coding as a nonlinear function $f : \mathbb{R}^M \to \mathbb{R}^N$ such that the sparse code at a given location can be recovered as

$$\boldsymbol{\alpha}^* = f(\mathbf{x}, \Theta), \tag{2}$$

where Θ represents the parameters for a given sparse coding layer including the dictionary and the regularization parameters. In sparse coding, the sparse coefficient $\boldsymbol{\alpha}^* \in \mathbb{R}^N$ is generally of much higher dimension than the input signal. Thus, if output sparse codes are naively and repeatedly fed into successive sparse coding layers, computational complexity quickly explodes. Inspired by Network in Network [32] and SqueezeNet [24], we introduce a sparse coding layer with an excessively skinny dictionary to reduce the dimensions of the sparse codes while also forcing sparsity of the low-dimension outputs, as shown in Fig. 1a. Unlike compression with linear projection, such as random projection or PCA, reducing the signal dimension with a sparse coding scheme achieves a good preservation of prior layer information while infusing more nonlinearity into the network.

The core building block of the SCN is the *bottleneck modules*, as depicted in Fig. 1b. 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 equipped 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 skinny dictionary which focuses more on dimensionality reduction and clustering in order to extract more abstract representations. More specifically, we have

$$\boldsymbol{\alpha}^* = f(f(\mathbf{x}, \Theta_e), \Theta_r), \tag{3}$$

where we have dropped the subscript indices for simplicity. Θ_e, Θ_r are the parameter sets of the expansion and reduction layers, respectively. We note that the order of the two specialized layers 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:

1) Expansion Layer Focuses on Partitioning Feature Space: Nonnegative sparse coding functions as a robust and stable partition of the input feature space [61], 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 [35], [54], [61].

2) 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. Nonnegative sparse coding with skinny dictionary functions as clustering, which can be illustrated based on semi-NMF [10]. Several inspirational works [22], [36], [39] illustrate the relations between sparse coding, dictionary learning and matrix factorization: In a reduction layer, when the given dictionary is skinny, the nonnegative sparse coding is equivalent with sparsity-regularized semi-NMF algorithm, which is strongly related to the K-means clustering. Hence, the skinny 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.

Empirically, there are three main operations within a bottleneck module. First is *i)* hyperpixel construction within 3×3 receptive fields of low dimensional inputs. Next, *ii)* an expansion layer transforms the input coefficients into a feature map of high dimension sparse codes. Finally, with *iii)* reduction layer, our skinny dictionary compresses the high-dimensional sparse codes into a low-dimensional space. In this paper, all dictionaries have 3×3 receptive fields. Unlike multilayer neural networks, there is no need to implement nonlinear activation functions after the sparse coding layer because of the enforcement of nonnegative constraint on the sparse code.

Our SCN is designed to stack multiple bottleneck modules in order to perform dimensionality expansion and reduction repeatedly. Batch normalization layer [25] 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.

C. Interpreting SCN 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. 1c, 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 (4).

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.

IV. Supervised Dictionary Learning for Multilayer Sparse Coding Network

In this section, we first describe the discriminative loss function of SCN. Then we introduce the adaptive regularization scheme, which allows each layer to automatically adapt its own ℓ_1-regularization parameters. Finally, we develop the dictionary updating rule for the multilayer sparse coding network by extending the task-driven dictionary learning [35], [61] to a multilevel case.

A. 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 learns to map the input image $\mathcal{A}_s^{(0)}$ to the corresponding label y_s, the optimization procedure of SCN is formulated as an empirical risk minimization problem based on multilevel optimization:

$$\begin{aligned}
&\min_{\theta} \frac{1}{S}\sum_{s=1}^{S} L(y_s, g(\mathcal{A}_s^{(H)}, \mathbf{w})) + \frac{\mu}{2}R(\theta),\\
&s.t.\ \boldsymbol{\alpha}_s^{(H)^*} = \arg\min_{\boldsymbol{\alpha}_s^{(H)}\geq \mathbf{0}} f(\mathbf{D}^{(H)}, \lambda^{(H)}, \mathbf{x}^{(H)_s}, \boldsymbol{\alpha}_s^{(H)}),\\
&\quad\vdots\\
&s.t.\ \ \boldsymbol{\alpha}_s^{(1)^*} = \arg\min_{\boldsymbol{\alpha}^{(1)}\geq \mathbf{0}} f(\mathbf{D}^{(1)}, \lambda^{(1)}, \mathbf{x}_s^{(1)}, \boldsymbol{\alpha}_s^{(1)}),\\
&s.t.\ \ \lambda^{(h)} > 0,\ \mathbf{x}_s^{(h)} = \psi(\boldsymbol{\alpha}_s^{(h-1)^*}), \quad \forall h = 1,\ldots,H, \qquad (4)
\end{aligned}$$

where $\theta = \{\mathbf{D}^{(h)}, \lambda^{(h)}, \mathbf{w}\}_{h=1}^H$ is the learnable parameter set including both dictionaries and regularization parameters. $g : \mathbb{R}^{n_H} \to \mathbb{R}$ is a linear classifier parameterized by $\mathbf{w}\in\mathbb{R}^{n_H}$. f is the nonnegative sparse coding operation defined in Eq. (3). 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 [39]. The motivation for training regularization parameters is that the use of cross-validation for parameter searching becomes a formidable task as the network becomes deeper, which is further discussed in Section IV-B.

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 gradient projection. Besides, enforcing normalization on the dictionary atom is dangerous when the task-driven regularization is employed. During training, some atoms could always remain inactivate if the regularization parameters increase beyond a large threshold. Hence, we only enforce a relatively weak ℓ_2-norm regularizor on the dictionary atoms.

B. Adaptive Regularization

Previous works on sparse coding usually select the regularization parameters manually by cross-validation. However, this scheme is infeasible when we extend the sparse coding to multilayer architectures. Tuning regularization parameters by hand would introduce two major issues in the case of multilayer architectures. First and obviously, manually searching for the optimal parameters of the underlying model would become onerous since the parameter space grows exponentially larger when the model becomes deeper. Second, during experimentation, we found that our multilayer sparse coding network with fixed regularization parameters suffers from low convergence rate and low classification performance.

To begin training, we initialize the ℓ_1-regularization parameter with some small value to avoid numerical issues (set to be 10^{-5} in our paper) and then optimize the underlying sparsity level of the network with the given training data. Applying error backpropagation with the projected gradient descent algorithm, we have

$$\lambda_1 \leftarrow \left(\lambda_1 - \rho \frac{\partial L}{\partial \boldsymbol{\alpha}^*}\frac{\partial \boldsymbol{\alpha}^*}{\partial \lambda_1}\right)_+, \tag{5}$$

where $\rho > 0$ is the learning rate, L is the total task-driven loss function defined in Eq. (4). The detailed updating rule for regularization parameters will be discussed in the next section. As we shall see in the experiment, Eq. (5) causes the regularization parameters to adjust during training in order to render sparse outputs.

C. Updating Dictionary and Regularization Parameter

Every sparse code is parameterized by the dictionary and regularization parameters, it is therefore natural to solve the multilevel optimization problem (4) with gradient descent method based on error backpropagation [9]. The derivation of the updating rules is based on the fixed point differentiation [35], [59], [61]. We state the first order optimality condition of the nonnegative elastic net, which is the core building block of the derivation:

Lemma 1 (Optimality Conditions of Nonnegative Elastic Net): The optimal sparse code $\boldsymbol{\alpha}^*$ *of (1) solves the following system:*

$$\mathbf{d}_j^\top(\mathbf{D}\boldsymbol{\alpha}^* - \mathbf{x}) + \lambda_2\alpha_j^* = -\lambda_1, \text{ if } \alpha_j^* > 0 \tag{6}$$
$$\mathbf{d}_j^\top(\mathbf{D}\boldsymbol{\alpha}^* - \mathbf{x}) + \lambda_2\alpha_j^* \geq -\lambda_1, \text{ otherwise.} \tag{7}$$

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 j^{th} element of $\mathbf{z}$. Followed by the classical result of Elastic Net [35] and the complementary slackness of KKT condition, when $\alpha_j = 0$, we have $z_j \leq 1$ and reach (6). Representation of $\boldsymbol{\alpha}_\Lambda^*$ can be achieved by applying algebraic simplification on (6) for all atoms j. When $\alpha_j > 0$, we have $z_j = 1$ and reach (7).

Eq. (6) demonstrates the relation between the sparse code and the dictionary. Based on Lemma 1, the desired gradients for optimizing SCN are summarized as follows:

$$\begin{cases} \partial L/\partial \mathbf{D} = -\mathbf{D}\boldsymbol{\gamma}\boldsymbol{\alpha}^\top + (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})\,\boldsymbol{\gamma}^\top, \\ \partial L/\partial \lambda_1 = -\boldsymbol{\gamma}, \\ \partial L/\partial \mathbf{x} = \mathbf{D}\boldsymbol{\gamma}, \end{cases} \tag{8}$$

where $\boldsymbol{\gamma}_\Lambda = (\mathbf{D}_\Lambda^\top\mathbf{D}_\Lambda + \lambda_2\mathbf{I}_{|\Lambda|})^{-1}\cdot\partial L/\partial\boldsymbol{\alpha}_\Lambda$. Optimizing dictionaries and regularization parameters with stochastic gradient descent is shown in Algorithm 1. We leave more a detailed derivation of the above equations in Appendix B.

Algorithm 1 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. $t = 1$.
 1: while stopping criterion not satisfied do
 2:     Randomly choose a sample pair $\{\mathbf{x}_i, y_i\}$ and let $\boldsymbol{\alpha}^{(0)} = \psi(\mathbf{x}_i)$.
 3:     for layer $h = 1$ to $H$ do
 4:         $\boldsymbol{\alpha}^{(h)^*} \leftarrow \arg\min_{\boldsymbol{\alpha}^{(h)}>\mathbf{0}} \frac{1}{2}\|\mathbf{x}^{(h)} - \mathbf{D}^{(h)}\boldsymbol{\alpha}^{(h)}\|_2^2 + \lambda_1^{(h)}\|\boldsymbol{\alpha}^{(h)}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}^{(h)}\|_2^2$.
            where $\mathbf{x}^{(h)} = \psi(\boldsymbol{\alpha}^{(h-1)})$.
 5:     end for
 6:     for layer $h = H$ down to 1 do
 7:         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 $\rho_t$ is the learning rate at time $t$.
 8:     end for
 9:     $t \leftarrow t + 1$.
10: end while
11: return $\{\mathbf{D}^{(h)}, \lambda^{(h)}\}_{h=1}^H$.
```

V. Experimental Verification

We conduct extensive experiments on CIFAR-10, CIFAR-100, STL-10, SVHN 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 15-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 [48].

Configuration of Network Architecture: Our sparse coding network consists of seven bottleneck modules with a total number of fourteen sparse coding layers. The architecture of the network is inspired by the Residual Network (ResNet) [18]. The network structure consists of two key features. First, there are no maxpooling layers. The spatial subsampling operation is fulfilled by specific sparse coding layers with a stride of 2. Second, the subsampling is carried out in deeper layers instead of the shallower ones. Both of these two strategies have been verified to improve the classification performance in multilayer architectures [18], [45]. Except for ImageNet dataset, the SCN architecture used in this paper is divided into three 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 ImageNet, the network structure is set to be $(32, 32K)\times 2-(64, 64K)\times 2-(128, 128K)\times 2-(256, 256K)\times 2$.

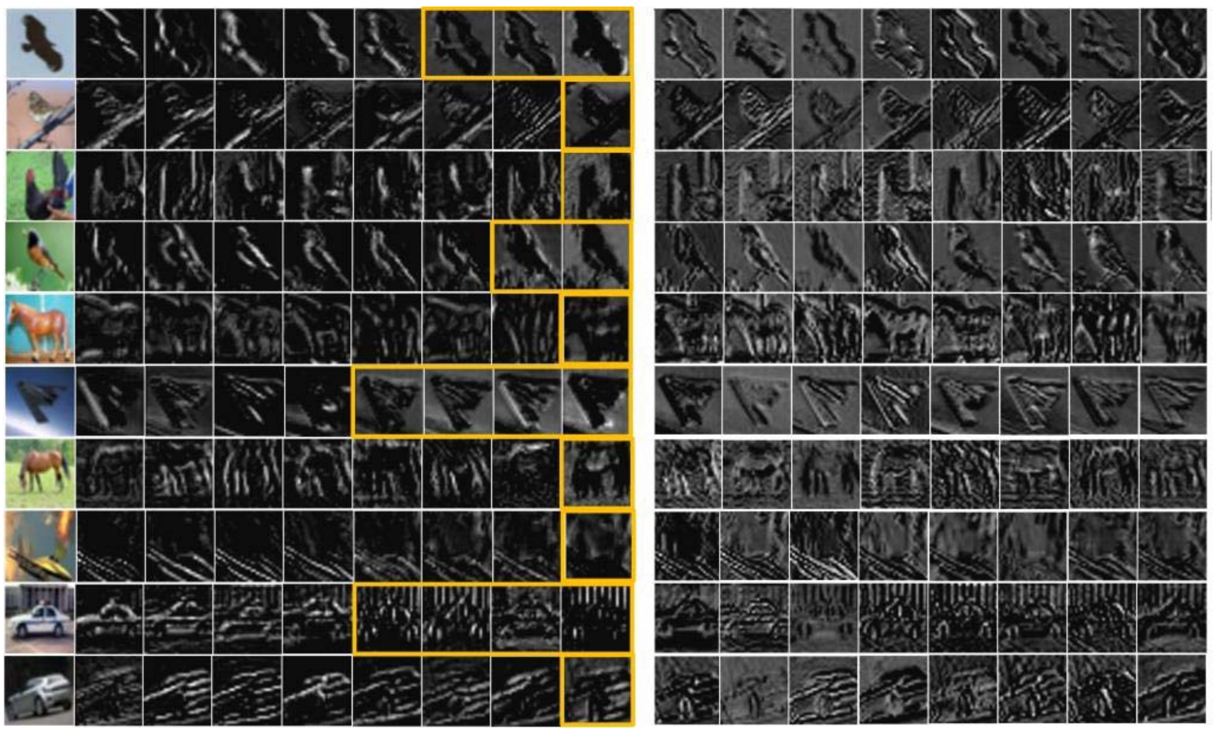


Fig. 2. Visualization of feature map: From left to right: Original image; feature maps of our sparse coding network - feature maps contain mostly background are labeled with yellow rectangles; and feature map of the baseline CNN.

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, SVHN and STL-10, we set $K = 4$. For ImageNet, 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. 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. We use the same network configurations for CIFAR-10, CIFAR-100, STL-10 and SVHN. For MNIST we set the number of filters at the first layer to be 8 due to the simplicity of the dataset.

Training: At the training stage, we apply data augmentation and preprocessing for all datasets except for MNIST and SVHN 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, which is a common procedure for preprocessing CIFAR-10 [8], [18], [30], [32]. 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 [8], [18], [30], [32]. We use a minibatch size of 128 for MNIST, CIFAR-10, CIFAR-100 and SVHN. 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 [19], 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. We evaluate our multilayer sparse coding network on the benchmark dataset of CIFAR-10, CIFAR-100, SVHN and MNIST.

Baseline Comparison Methods: We compare our proposed SCN with numerous multilayer sparse coding-based approaches, including multilayer sparsity regularized coding (OMP) [8] and nonnegative multilayer sparse coding (NOMP) [33]. We also compare with supervised convolutional kernel networks (SCKN) [34] and scattering network (ScatNet) [5]. For deep neural network baseline, we mainly compare with ResNet [18], [19], wide residual network (WRN) [63] and swapout networks (SwapOut) [44].

A. CIFAR-10 and CIFAR-100

Our most extensive experiment is conducted on the CIFAR-10 dataset [26], 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.

In Fig. 2, we display the feature maps of both the sparse coding network and the baseline CNN, which are produced

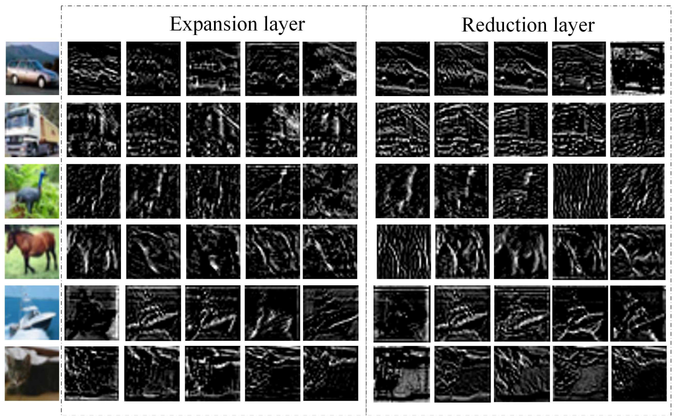


Fig. 3. Visualization of feature map in dimension expansion layer and reduction layer: From left to right: Original image; feature maps of expansion layer; feature maps of reduction layer.

by the output of the dimension expansion layer in the third bottleneck module of our sparse coding network and the corresponding ReLU layer of the CNN baseline, respectively. The baseline CNN is constructed by replacing every sparse coding layer of the SCN with a convolutional layer and a ReLU layer. The output of the selected layer contains 64 channels and for each image we present the eight feature maps with the largest ℓ_2-norms. These visualizations indicate the multilayer sparse coding network has a much better separation of the foreground and background. The background contains mostly low-frequency nondiscriminative information, which can be reconstructed easily with few dictionary atoms. Together with the nonnegativity constraint on the sparse codes, our network produces the unmixing effect as we see in the feature map. In addition, the feature map is also much sparser than that of the CNN. Moreover, the feature maps of the sparse coding network are similar to each other, verifying the fact that the atoms belonging to similar subspaces are activated.

We also study the relation between the features in the expansion layer and the reduction layer of a single bottleneck module. The corresponding learned features are extracted from the hidden outputs of the third bottleneck module. For each layer, we present the five feature maps with the largest ℓ_2-norms. Visualization of these features are illustrated in Fig. 3. The features in expansion layer usually contain smaller parts of the object with a higher sparsity level. In addition, features in expansion layer contain more edge information compared to those in the reduction layer. In contrast, the features in the reduction layer are composed of larger parts of objects with more texture information. Hence, the reduction layer and the expansion layer are specialized in learning distinctive patterns of the hidden feature maps.

We now study the behavior of the expansion and reduction layers of our multilayer sparse coding network as well as the evolution of the regularization parameters by referring to Fig. 4.

1) Optimization of Regularization Parameters: The evolution of the regularization parameter with respect to epochs is shown in Fig. 4a. The displayed regularization parameters are extracted from each of the sparse coding layers, which contains a total of 14 learnable regularization parameters. The parameters grow to larger magnitude as training progresses and start to decrease when the learning rate is decreased by a factor of 10. Shown in Fig. 4b, large portion of the regularization parameters have a magnitude above 0.05, which is able to enforce the output to be highly sparse. Illustrated in Fig. 4c, less than 10% of the output elements of the last sparse coding layer are nonzero.

2) Behavior of Expansion and Reduction Layer: Illustrated in Fig. 4c, the outputs of the expansion layers are much sparser than the reduction layers. The shallower layers tend to have low reconstruction error with low sparsity level, whereas the deeper layers usually have high reconstruction error but high sparsity level. For instance, the first expansion layer has approximately 45% nonzero sparse coefficients with less than 25% reconstruction errors, while the two deepest expansion layers have less than 10% nonzero coefficients with 50%−60% reconstruction errors. This observation verifies the fact that the shallower layers produce low-level reconstructive features, while the deeper layers produce discriminative features with weak reconstructive power. Table I and Fig. 5 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, which further verifies the importance of the expansion layer.

Unlike the expansion layers, most of the reduction layers are far less discriminative as shown in Fig. 4d. Except for the last reduction layer that reaches a sparsity level of 20%, all others have 40% − 50% nonzero sparse coefficients.

3) SCN With Bottleneck Module Uses Parameters Efficiently: From Table I, 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× 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.

4) 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 I. Consider the small size of our model, the performance of SCN is rather strong and competitive.

In addition, we also exploit the techniques in deep learning community in order to further improve the performance of the proposed SCN model. More specifically, we evaluate the effectiveness of the shortcut connection in the SCN model. We impose shortcut connection on every bottleneck module by adding its input and output together followed by a ReLU layer. The SCN architecture with shortcut connection is denoted as ResSCN. Classification error rates of ResSCN-1, ResSCN-2 and ResSCN-4 are demonstrated in Table I.

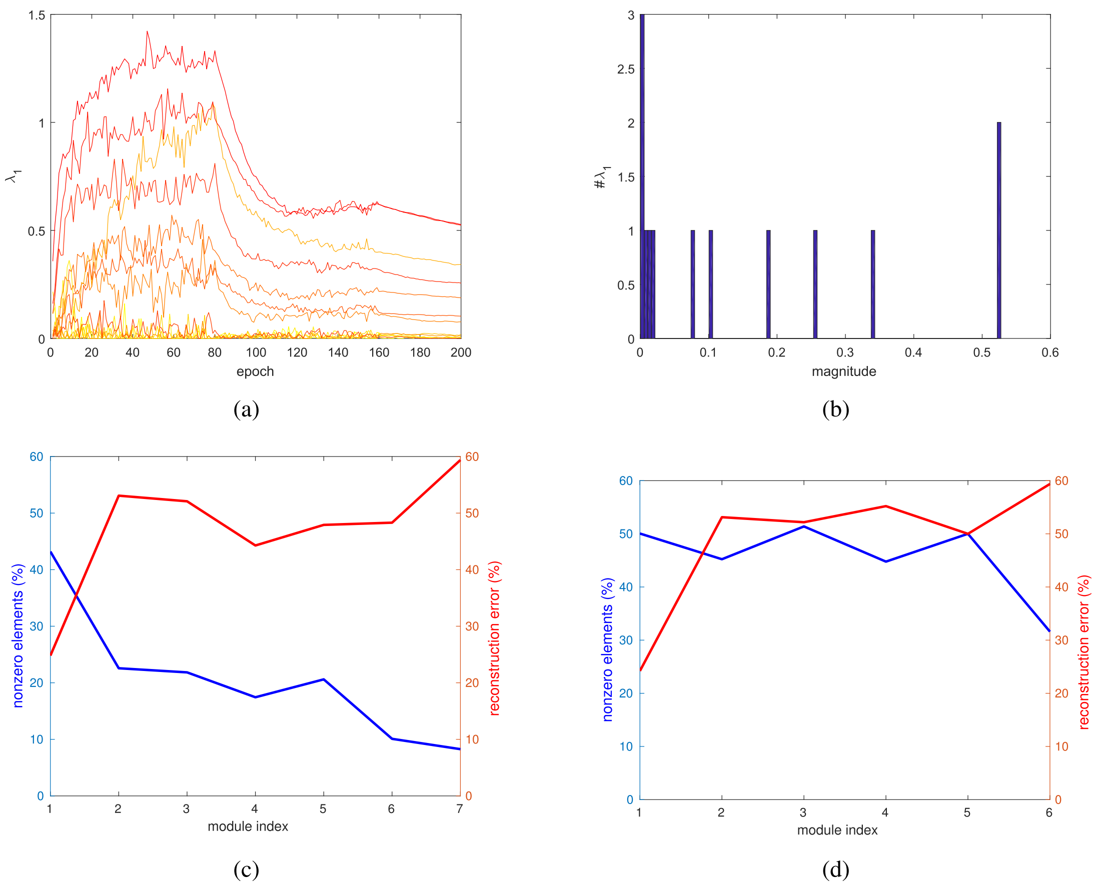


Fig. 4. (a)-(b): Evolution and distribution of the regularization parameters, respectively. The parameters are extracted from the last sparse coding layer. (c)-(d): Evaluation of the behavior of upsampling and downsampling layer, respectively. The blue and red lines indicate the nonzero elements and reconstruction error in percentage, respectively. Layer index specified by the module index.

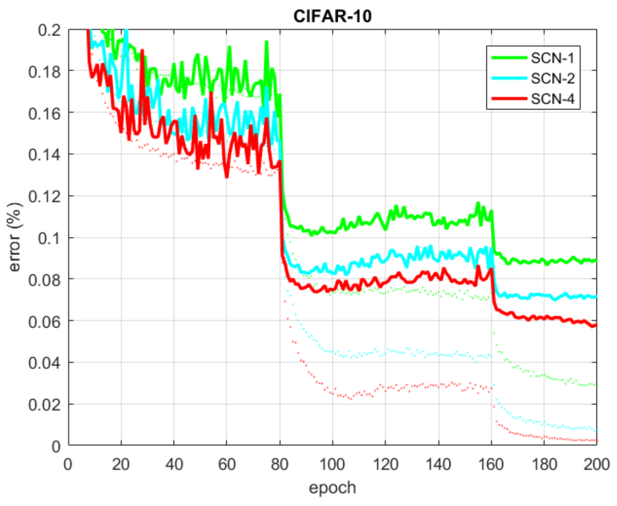
TABLE I
CLASSIFICATION ERROR (%) ON CIFAR-10 AND CIFAR-100

Method	# Params	# Layers	CIFAR-10	CIFAR-100
SCKN [34]	10.50M	10	10.20	-
OMP [8]	0.70M	2	18.50	-
PCANet [7]	0.28B	3	21.33	-
NOMP [33]	1.09B	4	18.60	39.92
NiN [32]	-	-	8.81	35.68
DSN [30]	1.34M	7	7.97	36.54
WRN [63]	36.5M	28	4.00	19.25
ResNet-110 [18]	0.85M	110	6.41	27.22
ResNet-1001 v2 [19]	10.2M	1001	4.92	27.21
ResNext-29 [56]	**68.10M**	**29**	**3.58**	**17.31**
SwapOut-20 [44]	1.10M	20	5.68	25.86
SwapOut-32 [44]	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**
ResSCN-1	0.17M	15	8.16	23.95
ResSCN-2	0.35M	15	6.91	21.42
ResSCN-4	**0.69M**	**15**	**5.52**	**18.78**

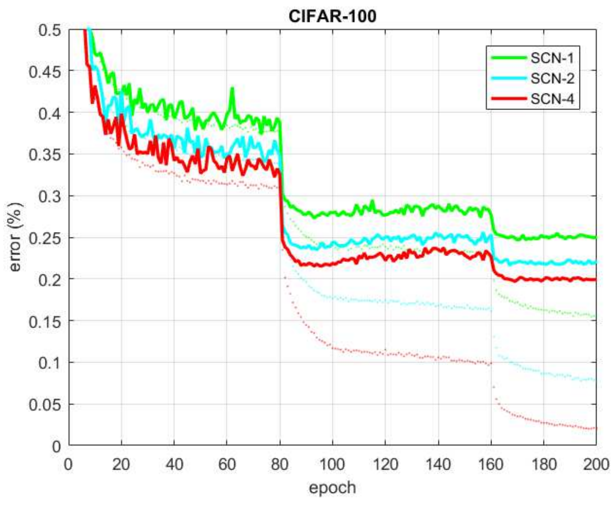
ResSCN-4 achieves classification error rates of 5.52% and 18.78% on CIFAR-10 and CIFAR-100. Hence, the performance of the SCN is further improved by employing shortcut connection.

B. 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 [64]. 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



(a)



(b)

Fig. 5. Learning curve of SCN on CIFAR-10 and CIFAR-100. Dotted and solid lines denote the learning curves of training and testing stage, respectively.

TABLE II
CLASSIFICATION ACCURACY (%) ON STL-10

Method	#Params	#Layers	Accuracy
SWWAE [67]	10.50M	10	74.33
Deep-TEN	25.60M	50	76.29
SCN-4	**0.69M**	**15**	**83.11**

TABLE III
SVHN CLASSIFICATION ERROR

Method	# params	# layers	Error (%)
RCNN [31]	2.67M	-	1.77
ReNet [50]	23.12M	7	2.38
DSN [30]	1.34M	7	1.92
Maxout [8]	-	-	2.37
NIN [32]	-	-	2.35
SCN-4	**0.69M**	**15**	**2.16**

state-of-the-art approach [67]. From table II, 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 [67] by almost 9%.

C. SVHN

SVHN [40] is a dataset consisting of color images of digits collected from Google Street View. The images are of size 32×32 with $73,257$ images for training and $26,032$ images for testing. The dataset also comes with $531,131$ additional labeled images. Again, we directly use the network configuration for CIFAR-100. This dataset is less difficult due to a large number of the labeled training samples. For a fair comparison, we delete 400 samples per training class and 200 samples per class from the extra set, which are used for cross-validation by the compared methods in Table III. The network is trained only on the training and the extra set. The image of the dataset is preprocessed by subtracting per-pixel-mean and we do not conduct any data augmentation. Due to the large size of the dataset, we only train our network with 20 epochs. We achieve a test error of 2.16% with a few learnable parameters. A summary of comparable methods is shown in Table III. Our sparse coding network outperforms the CNN-baseline with 0.8% and is comparable with other state-of-the-art performance while using substantially fewer parameters.

TABLE IV
CLASSIFICATION ERROR (%) ON MNIST

Method	#Params	#Layers	Accuracy
CKN [37]		2	0.39
ScatNet [5]	-	3	0.43
PCANet [7]	-	3	0.62
S-SC [61]	-	1	0.84
TDDL [35]	-	1	0.54
SCN-4	**0.69M**	**15**	**0.36**

D. MNIST

The MNIST [29] 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. We subtract the per-pixel-mean of each image and do not perform any data augmentation. The classification error on this dataset is reported in Table IV. With limited epochs, our sparse coding network achieves a classification error of 0.36%, which is comparable with state-of-the-art performance.

E. ImageNet

To further illustrate the efficiency and scalability of the proposed SCN on datasets with larger images, we train a 16-layer SCN which is composed of 8 bottleneck modules on the ImageNet 2012 dataset. The dataset consists of 1.28 million training images coming from $1,000$ classes, including

TABLE V
CLASSIFICATION ERROR (%) ON IMAGENET

Method	#Params	#Layers	Top-1 err.	Top-5 err.
ResNet-18 [18]	11.7M	18	29.66	10.50
ResNet-34 [18]	21.8M	34	26.17	8.56
SCN-4	**9.79M**	**16**	**29.58**	**10.75**

TABLE VI
EMPIRICAL INFERENCE TIME (MS) ON IMAGENET

Method	ResNet-18	ResNet-34	SCN-4
Inference time	9.06	10.87	18.25

50k validation images and 100k testing images. The models are trained on the 1.28 million training images, and evaluated on the 50k validation images. We evaluate both top-1 and top-5 error rates on this dataset.

We train the proposed SCN with a total number of 100 epochs and employ a batch size of 512. The model is trained with 8 Nvidia Volta V100 GPUs on Amazon Web Service (AWS) and takes a total number of 66 hours to finish the training. Initial learning rate is set to be 0.1 and the learning rate is reduce by a factor of 10 at epoch 30, 60 and 90. The batch size is set to 512. The purpose this experiment is to demonstrate the scalability of the proposed model, i.e., we do not aim at pursuing the highest accuracy but to achieve a competitive performance under fair comparison.

The experimental result on ImageNet is shown in Table V. We compare our work mainly with residual networks. For a fair comparison, we report the ResNet results produced by Tensorpack [55] which uses the same batch size and learning rate schedule for training. The proposed SCN-4 achieves an error rate of 29.58% and 10.75% for top-1 and top-5 error, respectively.

F. Empirical Computation Time Analysis

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.

On the dataset of ImageNet, we compare the empirical inference time of various networks including ResNet-18, ResNet-34 and SCN-4. For each of the networks, we repeatedly test $1,000$ images of the size 256×256 on single Nvidia Volta V100 GPU and report the averaged inference time in milliseconds (ms) in Table VI. On average, ResNet-18 and ResNet-34 inference each image at 9.06ms and 10.87ms, respectively. On the other hand, SCN-4 takes 18.25ms to inference single image.

VI. CONCLUSION AND DISCUSSION

In this paper, we have developed a novel multilayer sparse coding network by training the dictionaries and the regularization parameters simultaneously using an end-to-end supervised learning scheme. We have shown empirical evidence that the regularization parameters can adapt to the given training data. The high computational complexity of multilayer sparse coding networks has motivated us to explore more efficient strategies for accomplishing sparse recovery. We propose applying reduction layers within sparse coding modules to dramatically reduce the output dimensionality of the layers and mitigate computational costs. Moreover, we also show that our sparse coding network is compatible with other powerful deep learning techniques such as batch normalization. Our network produces results competitive with deep neural networks but uses significantly fewer parameters and layers. In particular, our network performs exceedingly well on CIFAR-100, indicating a lower training data requirement compared to multilayer neural networks.

Algorithm 2 FISTA for Nonnegative Elastic Net

```
Require: Dictionary $\mathbf{D} \in \mathbb{R}^{M\times N}$, $\kappa$ 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 $t = 0$, $\boldsymbol{\alpha}_t = \mathbf{0} \in \mathbb{R}^N$, $\boldsymbol{\gamma}_t =$
    $\mathbf{0} \in \mathbb{R}^N$, $s_0 = 1$.
1: while stopping criterion not satisfied do
2:     $\boldsymbol{\alpha}_{t+1} \leftarrow (\mathbf{A}\boldsymbol{\gamma}_t + \mathbf{b})_+$.
3:     $s_{t+1} \leftarrow (1 + (1 + 4s_t^2))/2$.
4:     $\boldsymbol{\gamma}_{t+1} \leftarrow \boldsymbol{\alpha}_{t+1} + (s_t - 1)(\boldsymbol{\alpha}_{t+1} - \mathbf{x}_t)/s_{t+1}$.
5:     $t \leftarrow t + 1$.
6: end while
7: return Nonnegative sparse code $\boldsymbol{\alpha}_t$.
```

APPENDIX

A. Solving Constrained Elastic Net Using FISTA

For the purpose of clarification, we describe the nonnegative FISTA in Algorithm 2, which is used for inference during training and testing. We denote $(\mathbf{A})_+$ as the element-wise nonnegative thresholding on $\mathbf{A}$.

B. Dictionary and Parameter Update

We now derive the backpropagation rule for solving problem 4. 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. We start by differentiating the empirical loss function with respect to every element of the dictionaries and regularization parameters:

$$\frac{\partial L}{\partial d_{jk}^{(h)}} = \frac{\partial L}{\partial \boldsymbol{\alpha}^{(H)}} \cdot \left(\prod_{i=H}^{h+1} \frac{\partial \boldsymbol{\alpha}^{(i)}}{\partial \boldsymbol{\alpha}^{(i-1)}}\right) \cdot \frac{\partial \boldsymbol{\alpha}^{(h)}}{\partial d_{jk}^{(h)}}, \tag{9}$$

$$\frac{\partial L}{\partial \lambda_1^{(h)}} = \frac{\partial L}{\partial \boldsymbol{\alpha}^{(H)}} \cdot \left(\prod_{i=H}^{h+1} \frac{\partial \boldsymbol{\alpha}^{(i)}}{\partial \boldsymbol{\alpha}^{(i-1)}}\right) \cdot \frac{\partial \boldsymbol{\alpha}^{(h)}}{\partial \lambda_1^{(h)}}, \quad \text{s.t.} \quad \lambda_1^{(h)} > 0, \tag{10}$$

where $d_{jk}^{(h)}$ is the (j,k)-element of the dictionary $\mathbf{D}^{(h)}$. To solve for (9) and (10), 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 [35], [61]. Let $\boldsymbol{\alpha} \in \mathbb{R}^N$ [2] be the optimal point of Lasso problem, it then satisfies the optimality condition based on (6) and for all $\boldsymbol{\alpha}_\Lambda > 0$:

$$(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})\boldsymbol{\alpha}_\Lambda - \mathbf{D}_\Lambda^\top \mathbf{x} + \lambda_1 \mathbf{1}_{|\Lambda|} = \mathbf{0}, \tag{11}$$

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.

1) 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 [35], [61] and $\partial \boldsymbol{\alpha}_{\Lambda^c}/\partial d_{jk} = \mathbf{0}$, where Λ^c is the complementary of Λ. Differentiate both sides of (11) with respect to d_{jk} for all $j \in \Lambda$:

$$\left(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|}\right)\frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial d_{jk}} + \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial d_{jk}}\boldsymbol{\alpha}_\Lambda - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial d_{jk}} = \mathbf{0}, \tag{12}$$

which is equivalent with

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

We reach the updating rule for a single dictionary element:

$$\frac{\partial L}{\partial d_{jk}} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1}\left(\frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial d_{jk}} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial d_{jk}}\boldsymbol{\alpha}_\Lambda\right). \tag{14}$$

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

$$\frac{\partial L}{\partial \mathbf{D}} = -\mathbf{D}\boldsymbol{\gamma}\boldsymbol{\alpha}^\top + (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})\,\boldsymbol{\gamma}^\top, \tag{15}$$

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 (21) on modern GPUs.

2) Differentiation of $\partial L/\partial \lambda$: Differentiating both sides of Eq. (11) with respect to λ_1:

$$\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda \frac{\partial \boldsymbol{\alpha}}{\partial \lambda_1} = -1, \tag{16}$$

and we reach at

$$\frac{\partial L}{\partial \lambda_1} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot -(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1} = -\boldsymbol{\gamma}. \tag{17}$$

3) 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 (11) with respect to x_i:

$$(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})\frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial x_i} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i} = \mathbf{0}, \tag{18}$$

where x_i is the ith element of $\mathbf{x}$. (18) is equivalent with

$$\frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial x_i} = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1}\frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i}. \tag{19}$$

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

Therefore we have

$$\frac{\partial L}{\partial x_i} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda + \lambda_2 \mathbf{I}_{|\Lambda|})^{-1}\frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i}, \tag{20}$$

which can be further simplified as

$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{D}\boldsymbol{\gamma}. \tag{21}$$

References

[1] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, Dec. 1997.

[2] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009.

[3] L. Bo, X. Ren, and D. Fox, "Multipath sparse coding using hierarchical matching pursuit," in *Proc. CVPR*, Jun. 2013, pp. 660–667.

[4] D. M. Bradley and J. A. Bagnell, "Differentiable sparse coding," in *Proc. NIPS*, Dec. 2008, pp. 113–120.

[5] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[6] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[7] T. H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A simple deep learning baseline for image classification," *IEEE Trans. Image Process*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.

[8] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proc. ICML*, Jul. 2011, pp. 921–928.

[9] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Ann. Oper. Res.*, vol. 153, no. 1, pp. 235–256, Sep. 2007.

[10] C. H. Q. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 45–55, Jan. 2010.

[11] T. T. Do, L. Gan, N. Nguyen, and T. D. Tran, "Sparsity adaptive matching pursuit algorithm for practical compressed sensing," in *Proc. ACSSC*, Oct. 2008, pp. 581–587.

[12] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *IEEE Trans. Image Process*, vol. 20, no. 7, pp. 1838–1857, Jul. 2011.

[13] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Stat.*, vol. 32, pp. 407–499, Apr. 2004.

[14] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *J. Amer. Stat. Assoc.*, vol. 96, no. 456, pp. 1348–1360, Dec. 2001.

[15] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. CVPR*, Jun. 2016, pp. 2414–2423.

[16] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. ICML*, Jun. 2010, pp. 399–406.

[17] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. CVPR*, Jun. 2015, pp. 5353–5360.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jul. 2016, pp. 770–778.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *CoRR*, vol. abs/1603.05027, pp. 1–15, Mar. 2016.

[20] Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi, "Unsupervised feature learning by deep sparse coding," in *Proc. ICDM*, Apr. 2014, pp. 902–910.

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, pp. 1–18, Jul. 2012.

[22] P. O. Hoyer, "Non-negative sparse coding," in *Proc. NNSP*, Feb. 2002, pp. 557–565.

[23] P. O. Hoyer and A. Hyvärinen, "A multi-layer sparse coding network learns contour coding from natural images," *Vis. Res.*, vol. 42, no. 12, pp. 1593–1605, Jun. 2002.

[24] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5MB model size," *CoRR*, Feb. 2016.
[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.
[26] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ., Toronto, Toronto, ON, Canada, Apr. 2009.
[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, Dec. 2012, pp. 1097–1105.
[28] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, "ICA with reconstruction cost for efficient overcomplete feature learning," in *Proc. NIPS*, Dec. 2011, pp. 1017–1025.
[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[30] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. JMLR*, Feb. 2015, pp. 562–570.
[31] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proc. CVPR*, Jun. 2015, pp. 3367–3375.
[32] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, pp. 1–10, Dec. 2013.
[33] T. Lin and H. T. Kung, "Stable and efficient representation learning with nonnegativity constraints," in *Proc. ICML*, Jun. 2014, pp. 1323–1331.
[34] J. Mairal, "End-to-end kernel learning with supervised convolutional kernel networks," *CoRR*, vol. abs/1605.06265, pp. 1–16, Dec. 2016.
[35] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 791–804, Apr. 2012.
[36] J. Mairal, F. R. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," in *Proc. JMLR*, Mar. 2010, pp. 19–60.
[37] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2014, pp. 2627–2635.
[38] M. Maire, S. X. Yu, and P. Perona, "Reconstructive sparse code transfer for contour detection and semantic labeling," in *Proc. ACCV*, Oct. 2014, pp. 273–287.
[39] T. Moreau and J. Bruna, "Understanding trainable sparse coding via matrix factorization," in *Proc. ICLR*, Apr. 2017, pp. 1–13.
[40] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with," in *Proc. NIPS*, Dec. 2011, pp. 371–280.
[41] V. C. Raykar and L. H. Zhao, "Nonparametric prior for adaptive sparsity," *CoRR*, vol. 9, pp. 629–636, May 2010.
[42] Y. Quan, Y. Xu, Y. Sun, Y. Huang, and H. Ji, "Sparse coding for classification via discrimination ensemble," in *Proc. CVPR*, Jul. 2016, pp. 5839–5847.
[43] X. Shen and Y. Wu, "A unified approach to salient object detection via low rank matrix recovery," in *Proc. CVPR*, Jul. 2012, pp. 853–860.
[44] S. Singh, D. Hoiem, and D. A. Forsyth, "Swapout: Learning an ensemble of deep architectures," in *Proc. NIPS*, Dec. 2016, pp. 28–36.
[45] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, pp. 1–14, Dec. 2014.
[46] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proc. CVPR*, Jun. 2014, pp. 1701–1708.
[47] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller, "A deep semi-NMF model for learning hidden representations," in *Proc. ICML*, Jun. 2014, pp. 1692–1700.
[48] A. Vedaldi and K. Lenc, "MatConvNet-convolutional neural networks for MATLAB," in *Proc. ICM*, Oct. 2015, pp. 689–692.
[49] 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 *Proc. JMLR*, Dec. 2010, pp. 3371–3408.
[50] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. C. Courville, and Y. Bengio, "Renet: A recurrent neural network based alternative to convolutional networks," *CoRR*, vol. abs/1505.00393, pp. 1–9, May 2015.
[51] 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 Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 372–386, Feb. 2012.
[52] Z. Wang, Q. Ling, and T. Huang, "Learning deep ℓ_o encoders," in *Proc. 13th AAAI Conf. Artif. Intell.*, Jan. 2016, pp. 2194–2200.
[53] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. ICCV*, Dec. 2015, pp. 370–378.
[54] J. Wright, A. A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.
[55] Y. Wu *et al.* (2016). *Tensorpack*. [Online]. Available: https://github.com/tensorpack/
[56] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," Nov. 2016, *arXiv:1611.05431*. [Online]. Available: https://arxiv.org/abs/1611.05431
[57] B. Xin, Y. Wang, W. Gao, B. Wang, and D. Wipf, "Maximal sparsity with deep networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2016, pp. 4340–4348.
[58] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. ICML*, Jul. 2015, pp. 2048–2057.
[59] J. Yang, Z. Wang, Z. Lin, X. Shu, and T. Huang, "Bilevel sparse coding for coupled feature spaces," in *Proc. CVPR*, Jun. 2012, pp. 2360–2367.
[60] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. CVPR*, Jun. 2009, pp. 1794–1801.
[61] J. Yang, K. Yu, and T. Huang, "Supervised translation-invariant sparse coding," in *Proc. CVPR*, Jun. 2010, pp. 3517–3524.
[62] K. Yu, Y. Lin, and J. Lafferty, "Learning image representations from the pixel level via hierarchical sparse coding," in *Proc. CVPR*, Jun. 2011, pp. 1713–1720.
[63] S. Zagoruyko and N. Komodakis, "Wide residual networks," *CoRR*, vol. abs/1605.07146, pp. 1–15, Jun. 2016.
[64] H. Zhang, J. Xue, and K. Dana, "Deep TEN: Texture encoding network," Dec. 2016, *arXiv:1612.02844*. [Online]. Available: https://arxiv.org/abs/1612.02844
[65] S. Zhang, J. Wang, X. Tao, Y. Gong, and N. Zheng, "Constructing deep sparse coding network for image classification," *Pattern Recognit.*, vol. 64, pp. 130–140, Apr. 2017.
[66] Y. Zhang, K. Lee, and H. Lee, "Augmenting supervised neural networks with unsupervised objectives for large-scale image classification," *CoRR*, vol. abs/1606.06582, pp. 1–17, Jun. 2016.
[67] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun, "Stacked what-where auto-encoders," *CoRR*, vol. abs/1506.02351, pp. 1–12, Nov. 2015.
[68] S. Zhou, S. Zhang, and J. Wang, "Deep sparse coding network for image classification," in *Proc. ICIMCS*, Aug. 2015, Art. no. 24.
[69] H. Zou, "The adaptive lasso and its oracle properties," *J. Amer. Stat. Assoc.*, vol. 101, no. 476, pp. 1418–1429, Jan. 2006.

Xiaoxia Sun received the B.Eng. degree in electronic engineering from Beihang University, Beijing, China, in 2011, and the Ph.D. degree in electrical and computer engineering from Johns Hopkins University, Baltimore, MD, USA, in 2017.

He was a Research Scientist with the Robert Bosch Research Center, Sunnyvale, CA, USA, where he focused on the field of deep network quantization and generative models. From 2012 to 2015, he was a part-time Associate Member of Technical Staff at the U.S. Army Research Laboratory, Adelphi, MD, USA. He is currently a Video Machine Learning Engineer at Apple Inc., Cupertino, CA, USA. His current research interests are object detection and real-time video analysis. His research fields include the theory and applications of compressed sensing and sparse representations, machine learning, deep learning, and large-scale optimization problems.

Nasser M. Nasrabadi (S'80–M'84–SM'92–F'01) received the B.S. and Ph.D. degrees in electrical engineering from the Imperial College of Science and Technology (University of London), London, U.K., in 1980 and 1984, respectively. He was a member of the Technical Staff at the Phillips Research Laboratory, NY, USA, as an Assistant Professor with the Department of Electrical Engineering, Worcester Polytechnic Institute, an Associate Professor with the Department of Electrical and Computer Engineering, University at Buffalo, and a Senior Research Scientist with the U.S. Army Research Laboratory. In 2016, he joined West Virginia University, where he is currently a Professor with the Lane Department of Computer Science and Electrical Engineering. He founded the Biometrics and Identity, Innovation Center, and he is also the Director of the Cognitive Computing Laboratory (CCL). His research and teaching interests are in the areas of image processing, computer vision, machine learning, deep neural networks, biometrics, and sparsity theory. His current research interests include image processing, computer vision, biometrics, deep learning, statistical machine learning theory, sparsity, robotics, and neural networks applications to image processing. He has served as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON CIRCUITS, SYSTEMS AND VIDEO TECHNOLOGY, and the IEEE TRANSACTIONS ON NEURAL NETWORKS.

Trac D. Tran (S'94–M'98–SM'08–F'14) received the B.S. and M.S. degrees from the Massachusetts Institute of Technology, Cambridge, in 1993 and 1994, respectively, and the Ph.D. degree from the University of Wisconsin–Madison, in 1998, all in electrical engineering.

In 1998, he joined the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD, USA, where he was recently promoted to the rank of Professor. In 2002, he was an ASEE/ONR Summer Faculty Research Fellow with the Naval Air Warfare Center Weapons Division (NAWCWD), China Lake, CA, USA. He is currently a regular Consultant for the U.S. Army Research Laboratory, Adelphi, MD, USA. His research interests are in the field of digital signal processing, particularly in sparse representation, sparse recovery, sampling, multi-rate systems, filter banks, transforms, wavelets, and their applications in signal analysis, compression, processing, and communications. His pioneering research on integer-coefficient transforms and pre-/post-filtering operators has been adopted as critical components of Microsoft Windows Media Video 9 and JPEG XR—the latest international still-image compression standard ISO/IEC 29199-2.

Dr. Tran was a former member of the IEEE Technical Committee on Signal Processing Theory and Methods (SPTM TC) and is a member of the IEEE Image Video and Multidimensional Signal Processing (IVMSP) Technical Committee. He was the Co-Director (with Prof. J. L. Prince) of the 33rd Annual Conference on Information Sciences and Systems (CISS'99), Baltimore, MD, USA, in 1999. He has served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the IEEE TRANSACTIONS ON IMAGE PROCESSING. He is currently serving his second term as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING.