Supervised Multilayer Sparse Coding Networks for Image Classification

Xiaoxia Sun
Johns Hopkins University
xxsun@jhu.edu

Nasser M. Nasrabadi
West Virginia University
nasser.nasrabadi@mail.wvu.edu

Trac D. Tran
Johns Hopkins University
trac@jhu.edu

Abstract

In this paper, we propose a novel multilayer sparse coding network capable of efficiently adapting its own regularization parameters to a given dataset. 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. We also incorporate a new weighted sparse coding scheme into our sparse recovery procedure, offering the system 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 13-layer sparse coding network are demonstrated on four benchmark datasets, namely Cifar-10, Cifar-100, SVHN and MNIST, 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 fuses the benefits of depth with sparse coding’s characteristic ability to operate on smaller datasets. In such data-constrained scenarios, we demonstrate our technique can overcome the limitations of deep neural networks by exceeding the state of the art in accuracy.

1. Introduction

Sparse coding has shown promising performance on a range of computer vision tasks including image classification and target detection [35, 37, 43, 44, 46, 48, 49]. 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 [43, 47]. 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 [10, 13, 20, 40, 45]. This feat has encouraged researchers to try and augment the learning capacity of traditionally shallow sparse coding methods by adding layers [11, 14, 26]. 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 [4] ameliorates information degeneracy issues within deep architectures [12, 15]. 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 [14, 26].

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 the first portion of Section 2, there has been inspiring breakthroughs in adapting supervised dictionary learning algorithms for use in shallow sparse coding frameworks [29, 47], but recent progress has slowed. We attempt to build on past achievements by training a multilayer sparse coding network using end-to-end supervised dictionary learning.
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 reconstruction feedback. As the dictionaries grow more obese, 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. Inspired by the Network in Network [25] and SqueezeNet [17] architectures, we propose a downsampling sparse coding layer that balances discriminative power with reconstructive potential. In contrast to the fat dictionary, the downsampling 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 downsampling sparse coding layers.
- Dynamic adaptation of $\ell_1$ regularization parameters with task-driven regularization.
- Supervised, end-to-end training of a multilayer sparse coding network with the aforementioned features.

In Section 2, we briefly review the works related to supervised dictionary learning and adaptive regularization. In Section 3, we elaborate on our network design, adaptive regularization technique, and end-to-end supervised training procedure. In order to clearly perceive the efficiency of supervised learning, we do not apply any unsupervised learning schemes to pretrain the dictionary. In Section 4, we evaluate our multilayer sparse coding network on four benchmark datasets, including CIFAR-10, CIFAR-100, SVHN, and MNIST. The first three datasets are considered to be highly challenging for sparse coding. Of particular interest is the CIFAR-100 which poses formidable challenges to sparse coding and deep networks alike. In our evaluation, we show our network to decisively outperform shallow sparse coding architectures as well as a similarly structured convolutional neural network baseline. Moreover, we demonstrate our network attains highly competitive results with state-of-the-art models such as residual representation [13] in terms of both classification accuracy and convergence rate.

2. Related Work

**Supervised dictionary learning:** Supervised dictionary learning strengthens the discriminative power of the sparse codes by exploiting the labeled samples. Due to the nonsmoothness of the $\ell_1$-regularizer, computing the gradient with respect to the dictionary is a tricky task. Overcomplete independent component analysis [21] 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 [11] 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 et al. [2] propose to directly compute the gradient of the dictionary by switching the $\ell_1$ regularizer with the smoothed Kullback-Leibler divergence. More thorough study on task-driven dictionary learning algorithms with various applications are carried out in [29]. Applying fixed point differentiation and error backpropagation, a supervised dictionary learning scheme for the shallow sparse coding model is proposed in [47].

**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 [52] and has been proved to satisfy the oracle property [9]. Do et al. [2] 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 [56] 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 [8]. 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.

3. Multilayer Sparse Coding networks

3.1. Multilayer Architecture

To begin, we formulate a generalized, multilayer sparse coding architecture as illustrated Fig. [4]. Let an image with $H \times W$ pixels and $C$ channels be represented by a
Figure 1: Architecture of our multilayer sparse coding network: (a) Composite sparse coding module consists of vectorization operation on a patch of pixels for generating a hyperpixel and followed by consecutively stacking upsampling and downsampling sparse coding layers. In our experiment, vectorization operations at all layers are conducted within 3×3 receptive fields. (b) Our multilayer sparse coding network is constructed by repeatedly stacking multiple composite sparse coding modules. The network does not contain any pooling operation, subsampling is conducted with a stride of 2.

3-dimensional tensor \( Y \in \mathbb{R}^{H \times W \times C} \). Denote a single \( C \)-channel pixel as \( y_i \in \mathbb{R}^C \), where \( i \in [HW] \) is the linear index of the pixel. We define the hyperpixel \( y_i \in \mathbb{R}^M \) as the concatenation of neighboring pixels of \( y_i \) within a \( K \times K \) receptive field (a patch of neighboring pixels) such that \( y_i = [y_{i-1}, \ldots, y_{i+K}]^\top \) and \( M = CK^2 \). We denote the sparse coding as a nonlinear function \( f : \mathbb{R}^M \rightarrow \mathbb{R}^N \) such that the sparse code at the location \( i \) can be recovered as

\[
x^*_i = f(y_i, \Theta),
\]

where \( \Theta \) represents the parameters for a given sparse coding layer.

The sparse code \( x^* \in \mathbb{R}^N \) 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 [25] and SqueezeNet [17], we introduce a downsampling 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. 1b. 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.

Our multilayer sparse coding network is constructed by the repeated stacking of our composite sparse coding modules, as depicted in Fig. 1b. There are three main operations within a module. First is \( i \) hyperpixel construction within 3×3 receptive fields of low dimensional inputs. Next, \( ii \) an upsampling sparse coding layer transforms the input hyperpixel into a feature map of high dimension sparse codes.

Finally, with \( iii \) downsampling sparse coding, our skinny dictionary compresses the high-dimensional sparse codes into a low-dimensional space. More specifically, we have

\[
x^* = f(f(y, \Theta_u), \Theta_d),
\]

where we have dropped the subscript indices for simplicity. \( \Theta_u, \Theta_d \) are the parameter sets of the upsampling and downsampling sparse coding layers, respectively. In this paper, all upsampling dictionaries have 3×3 receptive fields and all downsampling dictionaries have 1×1 receptive fields. The 1×1 receptive field is used to make the dimensionality reduction more efficient. Unlike multilayer neural networks, there is no need to implement nonlinear activation functions after the sparse coding layer because of the enforcement of the nonlinear sparsity regularization prior.

### 3.2. Weighted Nonnegative Sparse Coding

Sharing a formulation similar to adaptive Lasso [52], our loss function is designed to increase the efficiency of the proposed multilayer sparse coding network. Formally, given an input signal \( y \in \mathbb{R}^M \) and a dictionary \( W \in \mathbb{R}^{M \times N} \) with \( N \) atoms and \( M \) measurements, we would like to represent the local feature \( y \) with a sparse signal \( x \in \mathbb{R}^N \) by solving the following problem

\[
x^* = \arg \min_{x \succ 0} \frac{1}{2} \| y - Wx \|_2^2 + \sum_{i=1}^{N} [\lambda_{i1} |x_i| + \lambda_{i2} \frac{1}{2} \|x_i\|_2^2], \tag{3}
\]

where \( \{\lambda_{i1}\}_{i=1}^{N}, \lambda_{i2} \) are the regularization parameters, \( \lambda_{i1} \neq 0, \forall i \in [N] \) and \( \lambda_{i2} > 0 \). Eq. (3) can be simplified as

\[
x^* = \arg \min_{x \succ 0} \frac{1}{2} \| y - Wx \|_2^2 + \| \Gamma x \|_1 + \frac{\lambda_{i2}}{2} \|x\|_2^2, \tag{4}
\]
by denoting \(\Gamma = \text{diag}(\lambda_{11}, \ldots, \lambda_{1N}) \in \mathbb{R}^{N \times N}\).

There are two major differences between Eq. (4) and the adaptive Lasso models previously reported in \[13\ 52\]. First, in the case of the adaptive Lasso, the initial regularization parameter is set with a nonzero estimation. In our case, the adaptive \(\ell_1\) regularization parameters need not start from a nonzero point since we train these parameters with backpropagation. The nonnegative constraint on the sparse code prevents our network from getting trapped into a linear system. Second, the \(\ell_1\)-regularization parameters are only constrained to be nonzero, not nonnegative, so as to reduce the chance of getting stuck at zero, where the boundary of the projection set lies.

In this paper, nonnegativity is enforced upon sparse codes to promote stability and efficiency during sparse recovery \[16\ 26\ 30\ 51\]. At the outset of training, the near-zero initialized \(\ell_1\) regularization parameters have negligible effect on enforcing sparsity patterns. Therefore, the nonnegativity constraint has the supplementary upshot of preventing the network from collapsing into a linear system. We observe through experimentation the nonnegativity constraint hastens convergence.

We solve problem (4) using the algorithm of alternative direction method of multipliers (ADMM) \[1\]. For clarity, we describe the ADMM algorithm for our multilayer sparse coding network in Algorithm 1. In practice, we set the parameter of the augmented Lagrangian term to be a fixed value so that we only have to compute the matrix inversion once when given a fixed dictionary. Similar strategies have also been adopted in \[27\]. The parameter \(\rho\) in Algorithm 1 needs to be carefully chosen in order to achieve a fast convergence for the sparse codes. In the case of our multilayer sparse coding network, the norm of dictionary atoms in each layer have drastically different magnitudes, which will be discussed in fuller detail in the next section. To compensate for the fluctuation of the dictionary norm, we empirically choose the parameter \(\rho = \rho_0 \|A_i\|_2^2\) where \(i\) is the index of the dictionary atom with largest \(\ell_2\)-norm.

**Algorithm 1** ADMM for multilayer sparse coding network

Require: Dictionary \(\mathbf{W} \in \mathbb{R}^{H \times N}\), input feature \(\mathbf{y} \in \mathbb{R}^N\), regularization parameters \(\Gamma, \lambda_2, \rho = \rho_0 \|A_i\|_2^2\), \(\rho_0 = 0.1\), precomputed \(C^{-1} = (\mathbf{W}^\top \mathbf{W} + \rho \mathbf{I}_N + |\Gamma|)^{-1}\), \(\mathbf{B} = C^{-1} \mathbf{A}^\top \mathbf{y}\), \(\mathbf{u}, \mathbf{z} = 0 \in \mathbb{R}^N\).

1: while stopping criterion not satisfied do
2: \(x = \mathbf{B} + \rho C^{-1}(\mathbf{z} - \mathbf{u})\)
3: \(z = (x + u - \text{diag}(\Gamma)/\rho)_+\)
4: \(u = u + x - z\)
5: end while
6: return \(x\).

### 3.3. 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 \(\ell_1\)-regularization parameter \(\Gamma\) 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

\[
\lambda_{1i} \leftarrow \lambda_{1i} - \eta \frac{\partial L}{\partial \mathbf{x}^*} \frac{\partial \mathbf{x}^*}{\partial \lambda_{1i}}, \text{ s.t. } \lambda_{1i} \neq 0, \tag{5}
\]

where \(\eta > 0\) is the learning rate, \(L\) is the total task-driven loss function defined in Eq. (6). 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.

#### 3.4. Supervised Dictionary Learning

Most of the dictionary learning methods confine the dictionary atoms within a closed unit \(\ell_2\)-norm ball in order to keep the dictionary from exploding and producing trivially sparse solutions. During the experiment, we found that such restriction severely hampers the convergence rate of our sparse coding network when it becomes significantly deep. Furthermore, enforcing normalization on the dictionary atoms is dangerous when task-driven regularization is employed. As training progresses, some atoms will become permanently deactivated by regularization parameters which have exceeded a certain threshold. Therefore, we only loosely regularize the dictionary atoms with an \(\ell_2\)-norm, otherwise known as weight decay in neural networks. More specifically, we have

\[
L(\Theta) = V(\Theta) + \frac{\mu}{2} \sum_{i=1}^H (\|\mathbf{W}^h\|_F^2 + \|\mathbf{\Gamma}^h\|_F^2), \tag{6}
\]

where \(\mu > 0\) is the weight decay, \(h\) is the layer index, \(\Theta\) is the parameters of the whole networks, \(V(\cdot)\) is the discriminative logistic loss function and \(L(\cdot)\) is the overall task-driven loss function.

To optimize the network, we need to compute the gradient of the loss function with respect to the input \(\mathbf{y}\), output \(\mathbf{x}\) and the regularization parameter \(\mathbf{\Gamma}\) for each sparse coding layer. We apply fixed point differentiation to reach the
desired gradients. The updating rules are stated as follows and a detailed derivation is left to the Appendix:

\[
\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial x} (W^T W + \lambda \mathbf{I})_{\Lambda}^{-1} \frac{\partial W_{\Lambda}^T x}{\partial W_{ij}} - \frac{\partial W_{\Lambda}^T W_{\Lambda} x}{\partial W_{ij}} \\
\frac{\partial L}{\partial y_i} = \frac{\partial L}{\partial x} (W^T W + \lambda \mathbf{I})_{\Lambda}^{-1} \frac{\partial W_{\Lambda}^T y}{\partial y_i} \\
\frac{\partial L}{\partial \lambda_{ij}} = \frac{\partial L}{\partial x} (W^T W + \lambda \mathbf{I})_{\Lambda}^{-1} \text{sign}(x_{\Lambda})_j, \text{ s.t. } \lambda_{ij} \neq 0,
\]

(7)

where the subscript \( \Lambda \) denotes the active set of the sparse code \( x \), \( W_{\Lambda} \) is composed of the active columns of \( W \) and \( x_{\Lambda} \) is the active elements of the sparse code. During training, the computation of \( (W^T W + \lambda \mathbf{I})_{\Lambda}^{-1} \) could be a bottleneck since we have to compute it at every location of all the sparse coding layers. Fortunately, as training progresses, the computation demand decreases since the outputs of each layer become sparser and the average size of the active set shrinks.

In the case of shallow sparse coding models, active atoms are usually defined as \( \{ x_i : |x_i| > \epsilon, \forall i \in [N] \} \), where \( \epsilon \) is a small constant value to avoid numerical instability and \( x_i \) is the \( i \)th element of the sparse code \( x \). In multilayer sparse coding networks, a fixed threshold \( \epsilon \) does not function well since the magnitude of the sparse codes changes drastically from one layer to another due to the lack of normalization on dictionaries at atoms. To compensate for this effect, we set the threshold as

\[
\epsilon = \epsilon_0 \frac{\|y\|_2}{\|A_i\|_2},
\]

(8)

where \( i \) is the index of a dictionary atom with largest \( \ell_2 \) norm and \( \epsilon_0 \) is the threshold when both dictionary atoms and input signal are \( \ell_2 \)-normalized. We set \( \epsilon_0 = 10^{-3} \) throughout our paper.

### 4. Experimental Verification

We evaluate our multilayer sparse coding network on the benchmark dataset of CIFAR-10, CIFAR-100, SVHN and MNIST. All programs are written in MATLAB with C++ and CUDA based on the MatConvNet framework [41]. All experiments are conducted on a server with three Nvidia Titan X GPUs. The batch size is set to be 64 for all four datasets.

The architecture of our sparse coding network and an equivalent CNN baseline is shown in Table 1. For the CNN baseline, each convolutional layer is followed by a ReLU layer [32], which is omitted in the table. The configuration of the network architecture is inspired by the residual

| Sparse Coding Network                  | CNN baseline                      |
|----------------------------------------|-----------------------------------|
| 3 × 3 upsampling, 16                   | 3 × 3 conv, 16                    |
| 1 × 1 downsampling, 16                 | 1 × 1 conv, 16                    |
| 3 × 3 upsampling, 64                   | 3 × 3 conv, 64                    |
| 1 × 1 downsampling, 16                 | 1 × 1 conv, 16                    |
| 3 × 1 downsampling, 128, /2            | 3 × 3 conv, 128, /2               |
| 1 × 1 downsampling, 32                 | 1 × 1 conv, 32                    |
| 3 × 3 upsampling, 128                   | 3 × 3 conv, 128                   |
| 1 × 1 downsampling, 64                  | 1 × 1 conv, 64                    |
| 3 × 3 upsampling, 256, /2               | 3 × 3 conv, 256, /2               |
| global average pooling                  | 10 or 100 way fc, softmax         |

The number of dictionary atoms of each downsampling layer is set to be 4 times smaller than its preceding upsampling layer, yielding a 0.25 compression rate. Our network has a total of 0.27 million learnable parameters.

### 4.1. CIFAR-10

Our most extensive experiment is conducted on the CIFAR-10 dataset [19], which consists of 60,000 color images that are evenly split 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.

During the training, every training image undergoes data augmentation by applying random horizontal flipping as well as random translation with up to 4 pixels in each direction. Both training and testing images are preprocessed with per-pixel-mean subtraction, which is a common procedure for preprocessing CIFAR-10 [6, 13, 23, 25]. We only tune the initial learning rate by cross-validation. We use the first 45,000 samples for training and the remaining 5,000 samples for testing. The weight decay is set to 0.0001 and the initial learning rate is set to 0.01. The learning rate is decreased by a factor of 10 after 80 epochs. We run a total number of 120 epochs which takes 76.5 hours on our server.

Since we only tune the initial learning rate, we do

---

\(^2\) Codes used in this paper will become publicly available soon.
Figure 2: (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.

Figure 3: 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.

not guarantee that our multilayer sparse coding network or the baseline CNN can reach its best performance.

In Fig. 3 we display the feature maps of both the sparse coding network and the baseline CNN, which are produced by the output of the 5th layer of our sparse coding network and the corresponding ReLU layer of the CNN baseline, respectively. The output of the selected layer contains 64 channels and for each image we present the eight feature maps with the largest $\ell_2$-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.

4.1.1 Behavior of Sparse Coding Layers

We now study the behavior of the upsampling and downsampling layers of our multilayer sparse coding network as well as the evolution of the regularization parameters by referring to Fig. 2.

Optimization of regularization parameters: The evolution of the regularization parameter with respect to epochs is shown in Fig. 2a. The displayed regularization parameters are extracted from the last sparse coding layer, which contains a total of 256 learnable regularization parameters. The parameters grow to larger magnitude as training progresses and cease to grow when the learning rate is decreased by a factor of 10. Shown in Fig. 2b, more than 90% of the regularization parameters have a magnitude above 0.001, which is able to enforce the output to be highly sparse. Illustrated in Fig. 2c less than 10% of the output elements of the last sparse coding layer are nonzero.

Upsampling layer: Illustrated in Fig. 2c the outputs of the upsampling layers are much sparser than the downsampling layers. The shallower layers tend to have low reconstruction error with low sparsity level, whereas the deeper layers usually have high reconstruction error but low sparsity level. For instance, the first upsampling layer has approximately 45% nonzero sparse coefficients with less than 25% reconstruction errors, while the two deepest upsampling 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.

Downsampling layer: Unlike the upsampling layers, most of the downsampling layers are far less discriminative as shown in Fig. 2d. Except for the last downsampling layer that reaches a sparsity level of 20%, all others have 40% – 50% nonzero sparse coefficients.
4.1.2 Fast Convergence with Task-driven Regularization

We now demonstrate the advantage of using weighted Lasso over the heuristic \( \ell_0 \) pursuit, such as OMP. We train our 14-layer sparse coding network\(^3\) by using both weighted Lasso and OMP. For the OMP-based network, we set the desired number of nonzero elements to be 15 for all sparse coding layers. The convergence comparisons between the OMP-based network and the weighted Lasso-based network are shown in Fig. 4. At 120 epochs, the network with OMP has converged on a nearly 0.25 test error, while our multilayer sparse coding network descends below the same level in just 5 epochs. The result also shows our network to converge substantially faster and reach a higher classification accuracy than the CNN baseline.

4.1.3 Comparison with State-of-the-art Models

We compare the performance of our multilayer sparse coding network with other state-of-the-art models using CIFAR-10, most of which are based on a CNN architecture. Our 14-layer sparse coding network achieves 91.43% accuracy on CIFAR-10 using merely 0.27M parameters. Among all other methods, only the 20-layer residual network (ResNet) has a comparable number of parameters, yet we maintain fewer layers. Our model significantly overwhelms the previous sparse-coding-based models, including the OMP-based \(^2\) and nonnegative-OMP based models \(^6\), by a margin of 10%.

---

\(^3\)Including 13 sparse coding layers and 1 fully connected layer.

4.2. CIFAR-100

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. We directly use the same network configuration and hyperparameters used in the CIFAR-10 experiment. We do not guarantee that our network is able to reach its best performance. We preprocess the images in exactly the same way as CIFAR-10, i.e., subtract per-pixel mean and perform data augmentation. A summary of the state-of-the-art methods on CIFAR-100 is provided in Table 3. Comparison of the convergence rate with our CNN baseline is shown in Fig. 5. With 14 sparse coding layers, our network achieves a classification accuracy of 72.64\(\%\), which surpasses most of the state-of-the-art CNN-based methods. Due to the strong regularization of sparse coding, we gain greater improvements on CIFAR-100 compared with the gains achieved on CIFAR-10, thus validating the efficiency of our multilayer sparse coding network when dealing with a relatively small number of training samples per class.

| Method   | # params | # layers | Accuracy (%) |
|----------|----------|----------|--------------|
| Maxout \(^{26}\) | -        | -        | 60.08        |
| Maxout \(^{6}\)  | 1.09M    | 4        | 63.46        |
| DSN \(^{23}\)   | 1.34M    | 7        | 63.46        |
| All-CNN \(^{38}\) | 1.40M    | 10       | 66.29        |
| Highway \(^{39}\) | 2.3M     | 19       | 67.76        |
| ResNet \(^{13}\) | 0.40M    | 32       | 68.10        |
| CNN-baseline   | 0.27M    | 14       | 67.58        |
| Proposed       | 0.27M    | 14       | 72.64        |

4.3. SVHN

SVHN \(^{33}\) is a dataset consisting of color images of digits collected from Google Street View. The images are of size 32 \(\times\) 32 with 73,257 images for training and 26,032

---

Table 2: CIFAR-10 Classification Accuracy.

| Method   | # params | # layers | Accuracy (%) |
|----------|----------|----------|--------------|
| Maxout \(^{6}\) | -        | -        | 90.02        |
| SCKN \(^{23}\) | 10.50M   | 10       | 90.80        |
| NIN \(^{25}\)  | -        | -        | 91.19        |
| DSN \(^{23}\)  | 1.34M    | 7        | 92.03        |
| ResNet \(^{13}\) | 0.27M    | 20       | 91.75        |
| OMP \(^{6}\)   | 0.70M    | 2        | 81.50        |
| PCANet \(^{5}\) | 0.28B    | 3        | 78.67        |
| NOMP \(^{26}\) | 1.09B    | 4        | 81.40        |
| CNN-baseline  | 0.27M    | 14       | 88.56        |
| Proposed      | 0.27M    | 14       | 91.43        |

Table 3: CIFAR-100 Classification Accuracy.
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 4. 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 4. 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 4: SVHN Classification Error.

| Method   | # params | # layers | Error (%) |
|----------|----------|----------|-----------|
| RCNN [24] | 2.07M    | -        | 1.77      |
| ReNet [42] | 23.12M   | 7        | 2.38      |
| DSN [23]   | 1.34M    | 7        | 1.92      |
| Maxout [6]  | -        | -        | 2.37      |
| NIN [25]    | -        | -        | 2.35      |
| CNN-baseline | 0.27M   | 14       | 2.95      |
| Proposed   | 0.27M    | 14       | 2.16      |

4.4. MNIST

The MNIST [22] 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. We run a total of 20 epochs. The classification error on this dataset is reported in Table 5. With limited epochs, our sparse coding network achieves a classification error of 0.39%, which is comparable with state-of-the-art performance. Meanwhile, our approach also easily outperforms the CNN baseline with a margin of 0.8%.

Table 5: MNIST Classification Error.

| Method   | # params | # layers | Error (%) |
|----------|----------|----------|-----------|
| ScatNet [1] | -        | 3        | 0.43      |
| PCANet [5]  | -        | 3        | 0.62      |
| Maxout [6]  | -        | -        | 0.45      |
| NIN [25]    | -        | -        | 0.47      |
| CKN [31]    | 0.44M    | 3        | 0.39      |
| DSN [23]    | 0.35M    | 3        | 0.39      |
| Highway [39] | 0.15M   | 20       | 0.45      |
| ResNet [13, 14] | -       | 100      | 0.51      |
| CNN-baseline | 0.27M   | 14       | 0.47      |
| Proposed    | 0.27M    | 14       | 0.39      |

5. 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. Experimental results also confirm that our network converges substantially faster than the OMP-based sparse coding network. The high computational complexity of multilayer sparse coding networks has motivated us to explore more efficient strategies for accomplishing sparse recovery. We propose applying downsampling 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. We have demonstrated our sparse coding network easily outperforms a comparable baseline CNN. Moreover, 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.

References

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning, 3(1):1–122, Jan. 2011.
[2] D. M. Bradley and J. A. Bagnell. Differentiable sparse coding. in NIPS, Dec. 2008.
A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification.

L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using

M. Liang and X. Hu. Recurrent convolutional neural network for

C.-H. Z. Jian Huang, Shuangge Ma. Adaptive Lasso for sparse high-

F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, 
P. O. Hoyer. Non-negative sparse coding.
in IEEE Workshop on
Neural Networks for Signal Processing, Feb. 2002.

T. H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. PCANet: A simple deep learning baseline for image classification? IEEE Transactions on Image Processing, 24(12):5017–5032, Dec. 2015.

A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. in ICML, Jul. 2011.

T. T. Do, L. Gan, N. Nguyen, and T. D. Tran. Sparsity adaptive matching pursuit algorithm for practical compressed sensing. in Asilomar Conference on Signals, Systems and Computers, Oct. 2008.

W. Dong, L. Zhang, G. Shi, and X. Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. IEEE Transactions on Image Processing, 20(7):1838–1857, Jul. 2011.

J. Fan and R. Li. Variable selection via nonconcave penalized like-lihood and its oracle properties. Journal of the American Statistical Association, 96(456):1348–1360, Dec. 2001.

L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. in CVPR, Jun. 2016.

K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. in ICML, Jun. 2010.

K. He and J. Sun. Convolutional neural networks at constrained time cost. in CVPR, Jun. 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. in CVPR, Jun. 2016.

Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi. Unsupervised feature learning by deep sparse coding. in ICML, Jun. 2010.

K. He and J. Sun. Convolutional neural networks at constrained time cost. in CVPR, Jun. 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. in CVPR, Jun. 2016.

J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. in IEEE Transaction on Pattern Analysis and Machine Intelligence, 34(7):791–804, Apr. 2012.

J. Mairal, F. R. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research, 11:19–60, Mar. 2010.

J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. in NIPS, Dec. 2014.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. in ICML, Jun. 2010.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. in NIPS, Dec. 2011.

A. Patala. Residual networks in torch (MNIST layers). https://deepmlblog.wordpress.com/2016/01/05/residual-networks-in-torch-mnist/ Jun. 2016.

Y. Qian, Y. Xu, Y. Sun, Y. Huang, and H. Ji. Sparse coding for classification via discrimination ensemble. in CVPR, Jun. 2016.

V. C. Raykar and L. H. Zhao. Nonparametric prior for adaptive sparsity. Journal of Machine Learning Research, 9:629–636, May 2010.

X. Shen and Y. Wu. A unified approach to salient object detection via low rank matrix recovery. in CVPR, Jun. 2012.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. CoRR, abs/1412.6806, Dec. 2014.

R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. CoRR, abs/1505.00387, May 2015.

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. in CVPR, Jun. 2014.

A. Vedaldi and K. Lenc. Matconvnet–convolutional neural networks for matlab. in Proceeding of the ACM International Conference on Multimedia, Dec. 2015.

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, abs/1505.00393, May 2015.

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 Transaction on Pattern Analysis and Machine Intelligence, 34(2):372–386, Feb. 2012.

J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. IEEE Transaction on Pattern Analysis and Machine Intelligence, 31(2):210–227, Feb. 2009.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. in ICML, Jul. 2015.

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. in CVPR, Jun. 2009.

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

K. Yu, Y. Lin, and J. Lafferty. Learning image representations from the pixel level via hierarchical sparse coding. in CVPR, Jun. 2011.

K. Yu, Y. Lin, and J. Lafferty. Learning image representations from the pixel level via hierarchical sparse coding. in CVPR, Jun. 2011.

S. Zagoruyko and N. Komodakis. Wide residual networks. CoRR, abs/1605.07146, May 2016.

L. Zhuang, H. Gao, Z. Lin, Y. Ma, X. Zhang, and N. Yu. Non-negative low rank and sparse graph for semi-supervised learning. Jun. 2012.

H. Zou. The adaptive lasso and its oracle properties. Journal of the American Statistical Association, 101(476):1418–1429, Dec. 2006.