TO APPEAR IN 2016 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING

SPARSE-CODED NET MODEL AND APPLICATIONS†

Youngjune Gwon[1], *Miriam Cha*[2], *William Campbell*[1], *H. T. Kung*[2], *Cagri Dagli*[1]

[1]MIT Lincoln Laboratory, [2]Harvard University

ABSTRACT

As an unsupervised learning method, sparse coding can discover high-level representations for an input in a large variety of learning problems. Under semi-supervised settings, sparse coding is used to extract features for a supervised task such as classification. While sparse representations learned from unlabeled data independently of the supervised task perform well, we argue that sparse coding should also be built as a holistic learning unit optimizing on the supervised task objectives more explicitly. In this paper, we propose sparse-coded net, a feedforward model that integrates sparse coding and task-driven output layers, and describe training methods in detail. After pretraining a sparse-coded net via semi-supervised learning, we optimize its task specific performance in a novel backpropagation algorithm that can traverse nonlinear feature pooling operators to update the sparse coding dictionary. We evaluate the sparse-coded net model with multi-class classification problems in speech, image, and text data. Our preliminary results immediately confirm a significant improvement over semi-supervised learning as well as the superior classification performance against deep stacked autoencoder neural network and GMM-SVM pipelines in small to medium-scale settings.

1. INTRODUCTION

Originally used to explain neuronal activations [1], sparse coding emerges as an important method to learn underlying structures of unknown data. The representational power of sparse coding has been demonstrated by the excellent performance for recognition tasks in various data types. According to an empirical study by Coates *et al.* [2], a semi-supervised approach on sparse representations and a linear classifier performs on par with or sometimes even superior to more complex techniques such as RBM and deep neural network for visual recognition tasks on the CIFAR-10 and NORB benchmarks.

†This work is sponsored by the Defense Advanced Research Projects Agency under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

We have also been able to draw a similar conclusion from our experiments with sparse coding. Under its popular semi-supervised setting, sparse coding is used as a feature extractor for another algorithm and not directly involved in task-specific finetuning. With these in mind, it is sound to build an integrated learning unit based on sparse coding that can optimize the task objectives more explicitly. We propose sparse-coded net, a feedforward model consisting of sparse coding and task-specific output layers, and describe training methods. In particular, we discuss a novel backpropagation algorithm that significantly improves the performance of a pre-trained sparse-coded net via semi-supervised learning.

Similar to deep architectures, sparse-coded net can take in a large classifiable data unit (*e.g.*, high-resolution image, long speech sample, text corpus) to produce an efficient but label-consistent representation for the whole. Our preliminary results on speech, image, and text datasets indicate that sparse-coded net substantially improves over semi-supervised learning in the classification accuracy. We also find that sparse-coded net is superior or competitive to deep stacked autoencoder neural network and GMM-SVM in small to medium-scale evaluations.

Rest of this paper is organized as follows. In Section 2, we provide a background on sparse coding. Section 3 describes a semi-supervised learning framework based on sparse coding. Section 4 introduces the sparse-coded net model, explains architectural principles, and discusses training methods in detail. In Section 5, we present a comparative experimental evaluation using synthetic speech, CIFAR-10 image, and Wikipedia text datasets. Section 6 will conclude the paper.

2. SPARSE CODING BACKGROUND

Sparse coding is an unsupervised method to learn an efficient representation of data using a small number of basis vectors. Given an example $\mathbf{x} \in \mathbb{R}^N$, sparse coding searches for a representation $\mathbf{y} \in \mathbb{R}^K$ (*i.e.*, the feature vector for $\mathbf{x}$) while simultaneously updating the dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$ of K basis vectors by

$$\min_{\mathbf{D},\mathbf{y}} \|\mathbf{x} - \mathbf{D}\mathbf{y}\|_2^2 + \lambda \|\mathbf{y}\|_1 \quad \text{s.t. } \|\mathbf{d}_i\|_2 \leq 1, \forall i \qquad (1)$$

where $\mathbf{d}_i$ is ith dictionary atom in $\mathbf{D}$, and λ is a regularization parameter that penalizes over the ℓ_1-norm, which induces

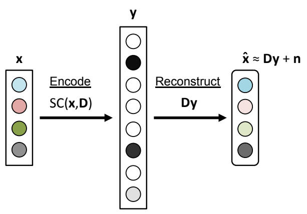


Fig. 1: Sparse coding $\texttt{SC}(\mathbf{x},\mathbf{D})$ encodes input data $\mathbf{x}$ as a sparse linear combination $\mathbf{y}$ of basis vectors from a dictionary $\mathbf{D}$ through the ℓ_0- or ℓ_1-regularized optimizations. The generative path reconstructs the original data by the matrix-vector multiplication $\hat{\mathbf{x}} \approx \mathbf{D}\mathbf{y}$.

a sparse solution. With $K > N$, sparse coding typically trains an overcomplete dictionary. This makes the sparse code $\mathbf{y}$ higher in dimension than $\mathbf{x}$, but only a few elements in $\mathbf{y}$ are nonzero.

A more direct way to control sparsity is to regularize on the ℓ_0 pseudo-norm $\|\mathbf{y}\|_0$, describing the number of nonzero elements in $\mathbf{y}$. However, it is known to be intractable to compute the sparsest ℓ_0 solution in general. The approach in Eq. (1) is called least absolute shrinkage and selection operator (LASSO) [3], a convex relaxation of the ℓ_0 sparse coding that induces sparse $\mathbf{y}$'s. We use least angle regression (LARS) [4] to solve the LASSO problem. We also consider orthogonal matching pursuit (OMP) [5], a greedy-ℓ_0 sparse coding algorithm that computes an at-most S-sparse ($S \ll N$) $\mathbf{y}$ extremely fast by

$$\min_{\mathbf{D},\mathbf{y}} \|\mathbf{x} - \mathbf{D}\mathbf{y}\|_2^2 \quad \text{s.t. } \|\mathbf{y}\|_0 \leq S, \|\mathbf{d}_i\|_2 \leq 1, \forall i. \qquad (2)$$

Fig. 1 summarizes the data encoding and reconstructive aspects of sparse coding.

3. SEMI-SUPERVISED LEARNING WITH SPARSE CODING

Semi-supervised learning [6] is a popular *data-driven* framework that combines the feature learning and task-specific optimization. It comprises both unsupervised and supervised stages. At the first stage, an unsupervised learning algorithm is applied to learn a representational mapping from unlabeled training examples. Using the learned mapping, we can compute the feature representation of raw examples. During the second stage, supervised training with a labeled dataset follows to optimize task-specific objectives.

In a discriminative machine learning problem such as classification, a small number of labeled data becomes a serious issue for classifier overfit although there will be sufficiently many unlabeled examples. Presumably, semi-supervised learning addresses the issue by introducing constraints on the classification model with unlabeled data. The model will be trained to discriminate different classes of data in terms of the presence and magnitude of specific features computable via the learned mapping.

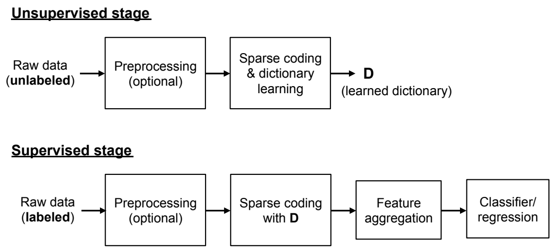


Fig. 2: Semi-supervised classification pipeline with sparse coding.

Given classification or regression of some target values being our primary task for text and multimedia applications, we describe a semi-supervised pipeline based on sparse coding in Fig. 2. Assuming m classes of data (*e.g.*, spoken languages in speech, image objects, text categories), we denote class labels $\mathcal{L} \in \{1, \ldots, m\}$. Ideally, an unbiased mix of unlabeled examples $\mathcal{D}_u = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots\}$ with each $\mathbf{x}^{(k)} \in \mathbb{R}^N$ from all classes will train a dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$. A learned $\mathbf{D}$ provides universal sparse modeling of the data regardless of class. Given an unknown input $\mathbf{x}$, sparse coding computes the representation $\mathbf{y}$ using $\mathbf{D}$

$$\mathbf{x} \approx y_1\mathbf{d}_1 + y_2\mathbf{d}_2 + \cdots + y_K\mathbf{d}_K,$$

where only several features y_j are nonzero. Optionally, raw data input can be preprocessed by normalization and whitening before sparse coding for better result.

Supervised training requires a labeled dataset such as $\{(\mathbf{x}^{(1)}, l^{(1)}), (\mathbf{x}^{(2)}, l^{(2)}), \ldots\}$. Here, an example $\mathbf{x}^{(j)}$ is associated with its class label $l^{(j)} \in \mathcal{L}$. For many applications in text and multimedia, a classifiable data unit can be too large for both unsupervised and supervised algorithms. For example, an entire image file is a classifiable unit, and if the image is too large, it should be broken down to smaller patches. Similarly, speech utterances and audio-video clips having a long duration are also cut into patches.

A labeled dataset $\mathcal{D}_l = \{(\mathbf{X}^{(1)}, l^{(1)}), (\mathbf{X}^{(2)}, l^{(2)}), \ldots\}$ has the jth example $\mathbf{X}^{(j)} = [\mathbf{x}^{(j,1)}, \mathbf{x}^{(j,2)}, \ldots, \mathbf{x}^{(j,M_j)}]$ consisting of M_j patches with a label $l^{(j)}$. We use notation $\mathbf{x}^{(j,k)}$ for the kth patch from $\mathbf{X}^{(j)}$. Since each patch is an input to sparse coding, there will be too many sparse codes for a large classifiable unit. While patching makes sparse coding (and other learning algorithms) to learn useful features, we must reduce the number of feature values to prevent a classifier overfit possible during the supervised training. We consider a nonlinear technique called "pooling" popularized in computer vision. In particular, average and max pooling methods described below can be used.

1. Average: $f(\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(M)}\}) = \frac{1}{M}\sum_{j=1}^{M} |\mathbf{y}^{(j)}|$
2. Max: $f(\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(M)}\})$
$= \max_{\forall i}(\{|y_i^{(1)}|, \ldots, |y_i^{(M)}|\})$

In Algorithm 1, we summarize our semi-supervised learning approach with sparse coding.

Algorithm 1 *Semi-supervised learning via sparse coding*

```
1: input: unlabeled data patches $\mathcal{D}_u = \{\mathbf{x}^{(k)}\}_{k=1}^{n_u}$, labeled dataset $\mathcal{D}_l = \{(\mathbf{X}^{(j)} = [\mathbf{x}^{(j,k)}]_{k=1}^{M_j}, l^{(j)})\}_{j=1}^{n_l}$
2: output: trained classifier $C$ or regression $R$
3: (optional) preprocess $\mathcal{D}_u$ and $\mathcal{D}_l$ by normalization and whitening
4: while unsupervised stage
5:     do sparse coding on $\mathcal{D}_u$ and learn dictionary $\mathbf{D}$
6: end
7: while supervised stage
8:     do sparse coding on $\mathcal{D}_l$ using $\mathbf{D}$: $\mathbf{x}^{(j,k)} \longrightarrow \mathbf{y}^{(j,k)}$
9:     form feature vector for each $\mathbf{X}^{(j)}$ via pooling: $\{\mathbf{y}^{(j,k)}\}_{k=1}^{M_j} \longrightarrow \mathbf{z}^{(j)}$
10:    train supervised task $C$ or $R$ using $\{(\mathbf{z}^{(j)}, l^{(j)})\}_{j=1}^{n_l}$
11: end
```

4. SPARSE-CODED NET

The semi-supervised framework for sparse coding is a powerful model for classification and regression with various types of data. In fact, it has been known that the features formed on sparse codes followed by a linear classifier performs as well as (or even better than) a complex nonlinear classifier on carefully handcrafted features. However, we argue that we can still improve Algorithm 1. Our approach for such improvement is to build a tightly integrated network of sparse coding, pooling nonlinearity, and a supervised task for joint optimization. In this section, we introduce sparse-coded net model.

4.1. Architectural overview

In a supervised task, we are ultimately interested in correctly predicted class labels or regressed values. Recall that sparse coding models input data by fitting the quadratic loss function in Eqs. (1) and (2). This reconstructive objective is essential to generative data modeling, but it is not necessarily optimal for the classification setting where sparse coding is used as a higher-level feature extractor. Under the semi-supervised framework, the sparse coding dictionary is only learned during the unsupervised stage and never adjusted to optimize the supervised task. This gives us a motivation to build a holistic learning unit based on sparse coding that can be trained first in the semi-supervised manner and fine-tuned by backpropagating the classification errors to adjust the dictionary.

Sparse-coded net (SCN) is a feedforward network built on sparse coding. Fig. 3 describes an architectural overview of SCN. At the input layer of SCN, we apply the entire M patches from a classifiable data unit $\mathbf{X} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}]$ (*e.g.*, $\mathbf{X}$ is a large image file, speech sample, text document, or video clip). Each $\mathbf{x}^{(k)}$ is sparse coded to $\mathbf{y}^{(k)}$, and the pooled sparse code $\mathbf{z}$ is applied to the softmax output layer for multi-class classification. We note that the SCN output layer is left as a configurable choice for application needs. While softmax regression is our default choice, other possibilities include linear regression, logistic regression, SVM, and perceptron.

4.2. Pretraining

Hinton *et al.* [7] suggested initialize a neural network by pretraining its hidden layers with an unsupervised method. For SCN, such pretraining can be fulfilled by semi-supervised learning. We use Algorithm 1 to pretrain an SCN. As a configurable template, we have purposefully left out specifics of the supervised task in Algorithm 1. Here, we explain supervised training of the softmax output layer for multi-class classification, which takes place after sparse coding and dictionary learning.

Softmax regression generalizes logistic regression for binary classification to an m-class classification problem. We use the matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ to denote the complete softmax model parameters for all m classes (*i.e.*, $\mathbf{w}_c$ is for class $c = 1, \ldots, m$). Softmax regression essentially estimates the likelihood of a given example $\mathbf{X}$ for being class c

$$p(l = c|\mathbf{X}) = h_{\mathbf{w}_c}(\mathbf{z}) = \frac{e^{\mathbf{w}_c^\top \mathbf{z}}}{\sum_{c'} e^{\mathbf{w}_{c'}^\top \mathbf{z}}},$$

where l is the class label, and $\mathbf{z}$ the pooled sparse code for $\mathbf{X}$ computed on the SCN feedforward path (see Fig. 3). Similar to logistic regression, softmax is based on log-linear model. It uses linear regression $\mathbf{w}_c^\top \mathbf{z}$ to calculate class-specific scores and maps to a probability space by normalizing the exponentiated scores. Class label prediction can be done in a soft decision by thresholding, which allows multiple predicted labels for a test example. For hard decision, we predict the label $\hat{l} = \arg\max_l p(l = c|\mathbf{X})\ \forall c$.

To train $\mathbf{W}$, we minimize the average cross-entropy loss function on the log-likelihood using the labeled dataset $\mathcal{D}_l$

$$J(\mathbf{W}) = -\frac{1}{n_l} \sum_{c=1}^{m} \sum_{j=1}^{n_l} \mathbb{1}\{l^{(j)} = c\} \log \frac{e^{\mathbf{w}_c^\top \mathbf{z}^{(j)}}}{\sum_{c'=1}^{m} e^{\mathbf{w}_{c'}^\top \mathbf{z}^{(j)}}}, \quad (3)$$

where $\mathbb{1}\{.\}$ is an indicator function. Although there is no known closed-form solution for the minimum $J(\mathbf{W})$, we can use an iterative optimization algorithm such as gradient descent. Taking derivatives with respect to $\mathbf{w}_c$ yields the gradient

$$\nabla_{\mathbf{w}_c} J(\mathbf{W}) = \frac{1}{n_l} \sum_{j=1}^{n_l} \mathbf{z}^{(j)} \left[h_{\mathbf{w}_c}(\mathbf{z}^{(j)}) - \mathbb{1}\{l^{(j)} = c\} \right].$$

Finally, we use the gradient descent update rule for each $c = 1, \ldots, m$ with a learning rate α

$$\mathbf{w}_c := \mathbf{w}_c - \alpha \nabla_{\mathbf{w}_c} J(\mathbf{W}) \quad (4)$$

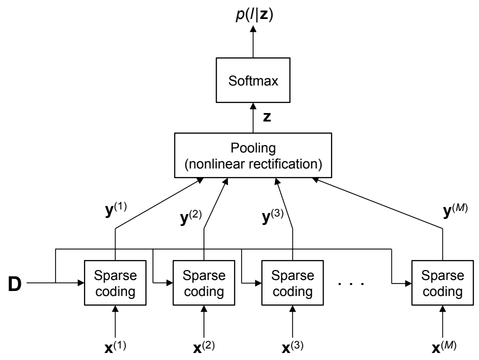


Fig. 3: Sparse-coded net modeling for a supervised task. The input layer of sparse-coded net takes in all patches $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}$ from an example $\mathbf{X}$. A classifiable data unit $\mathbf{X}$ can be the entire image, audio-video clip, and text document.

4.3. Optimization

By now, we have initialized the SCN with a softmax output layer for multi-class classification. While the pretrained SCN is already a good working pipeline—*i.e.*, its performance should be equivalent to that of the semi-supervised classification pipeline, we can further improve the performance. The biggest drawback for the pretrained SCN is that the classification error is never reflected to sparse coding. In fact, sparse coding and dictionary learning are done independently of the supervised training, which learns only the model parameters of the output layer. Therefore, if we can *backpropagate* the errors incurred at the output layer to the sparse coding layer and use them to update the dictionary $\mathbf{D}$, the SCN will be able to produce more label-consistent sparse representations for the output layer.

Unfortunately, the SCN backpropagation is not trivial. First of all, we cannot use backpropagation algorithms for neural networks due to obvious architectural differences. Also, the classification errors have to traverse the pooling layer that obscures flow of the error gradients with nonlinear rectification of multiple sparse codes into one feature vector. Let us examine the complete feedforward path of SCN:

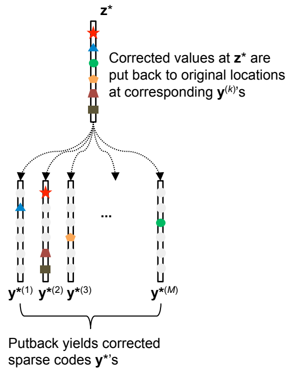
$$\{\mathbf{x}^{(j,k)}\}_{k=1}^{M_j} \xrightarrow{\mathbf{D}} \{\mathbf{y}^{(j,k)}\}_{k=1}^{M_j} \xrightarrow{\text{pool}} \mathbf{z}^{(j)} \xrightarrow{\text{softmax}} \hat{l}^{(j)}.$$

We rewrite the SCN softmax output loss as a function of $\mathbf{z}$

$$J(\mathbf{z}) = \frac{1}{2}\left\|\hat{l} - l\right\|^2 = \frac{1}{2}\left\|h_{\mathbf{w}_c}(\mathbf{z}^{(j)}) - \mathbb{1}\{l^{(j)} = c\}\right\|^2, \quad (5)$$

where $\hat{l}$ and l are the predicted and ground-truth labels, respectively. We evaluate $J(\mathbf{z})$ from examples $\mathbf{z}^{(j)}$, using the fact that $h_{\mathbf{w}_c}(\mathbf{z}^{(j)})$ should be 1 ideally if the ground-truth label for $\mathbf{z}^{(j)}$ is c, otherwise 0.

The objective for the SCN backpropagation is pass down the classification errors to adjust $\mathbf{z}$, traverse pooling layer, then adjust $\mathbf{D}$. To adjust $\mathbf{z}$, we hold the trained softmax parameters $\mathbf{W}$ and estimate the optimal $\mathbf{z}^*$ that minimizes $J(\mathbf{z})$.



$\mathbf{z}^*$
Corrected values at $\mathbf{z}^*$ are put back to original locations at corresponding $\mathbf{y}^{(k)}$'s
...
$\mathbf{y}^{*(1)}$
$\mathbf{y}^{*(2)}$
$\mathbf{y}^{*(3)}$
$\mathbf{y}^{*(M)}$
Putback yields corrected sparse codes $\mathbf{y}^*$'s

Fig. 4: Putback corrects sparse codes $\mathbf{y}^{(k)}$'s from $\mathbf{z}^*$

To do so, we perform the following gradient descent learning for each element $z_i^{(j)}$ of the vector $\mathbf{z}^{(j)}$ with a learning rate β

$$z_i^{(j)} := z_i^{(j)} - \beta\frac{\partial J(\mathbf{z})}{\partial z_i}. \quad (6)$$

The gradients are partial derivatives with respect to z_i

$$\begin{aligned}\frac{\partial J(\mathbf{z})}{\partial z_i} &= \left[h_{\mathbf{w}_c}(\mathbf{z}^{(j)}) - \mathbb{1}\{l^{(j)} = c\}\right]\frac{\partial h_{\mathbf{w}_c}(\mathbf{z})}{\partial z_i} \\ &= \left[h_{\mathbf{w}_c}(\mathbf{z}^{(j)}) - \mathbb{1}\{l^{(j)} = c\}\right] w_{i,c}.\end{aligned}$$

The value $w_{i,c}$ is the ith row and cth column element of the softmax parameter matrix $\mathbf{W}$. This update rule is intuitive because it adjust the elements of $\mathbf{z}$ according to their weighted error contribution $[h_{\mathbf{w}_c}(\mathbf{z}) - \mathbb{1}\{l^{(j)} = c\}]w_{i,c}$.

Next, we correct the unpooled original $\mathbf{y}^{(j,k)}$'s using the adjusted $\mathbf{z}^{*(j)}$'s. For max pooling, we introduce a procedure called *putback* illustrated in Fig. 4. For putback, we need to keep the original sparse codes $\mathbf{y}^{(j,k)}$ and the location of the maximum nonzero values that have resulted unadjusted $\mathbf{z}^{(j)}$. Each adjusted element in $\mathbf{z}^{*(j)}$ is put back to the corresponding location at the original sparse codes. For average pooling, we adjust all ith elements of $\mathbf{y}^{(j,k)}$'s by the average difference from $z_i^{*(j)}$

$$y_i^{(j,k)} := y_i^{(j,k)} + [z_i^{*(j)} - \frac{1}{M_j}\sum_{j=1}^{M} y_i^{(j,k)}]. \quad (7)$$

We denote $\mathbf{y}^{*(j,k)}$ the error-adjusted sparse codes by putback or Eq. (7). Using $\mathbf{y}^{*(j,k)}$, we can now update the sparse coding dictionary $\mathbf{D}$. (Note that it does not make sense to adjust given data input $\mathbf{x}^{(j,k)}$.) We describe three methods: 1) gradient descent; 2) rank-1 update; and 3) block-coordinate descent.

For gradient descent, we define the loss function with respect to $\mathbf{D}$

$$J(\mathbf{D}) = \frac{1}{2}\left\|\mathbf{D}\mathbf{y}^{*(j,k)} - \mathbf{x}^{(j,k)}\right\|_2^2. \quad (8)$$

To update each basis vector $\mathbf{d}_i$ in $\mathbf{D}$, we take the partial derivative with respect to $\mathbf{d}_i$

$$\frac{\partial J(\mathbf{D})}{\partial \mathbf{d}_i} = \left(\mathbf{D}\mathbf{y}^{*(j,k)} - \mathbf{x}^{(j,k)}\right) y_i^{*(j,k)},$$

which leads to the gradient descent update rule with a learning rate γ

$$\mathbf{d}_i := \mathbf{d}_i - \gamma\left(\mathbf{D}\mathbf{y}^{*(j,k)} - \mathbf{x}^{(j,k)}\right) y_i^{*(j,k)}. \qquad (9)$$

Rank-1 update first computes the residual matrix $\mathbf{E}_i = [\mathbf{X}^{(j)}]_{j=1}^{n_l} - \sum_{j\neq i} \mathbf{d}_i \cdot \mathrm{row}_i([\mathbf{Y}^{*(j)}]_{j=1}^{n_l})$ with respect to each $\mathbf{d}_i$. Here, all examples from $\mathcal{D}_l$ are concatenated to form the data matrix $[\mathbf{X}^{(j)}]_{j=1}^{n_l}$, and $[\mathbf{Y}^{*(j)}]_{j=1}^{n_l}$ are the corresponding error-adjusted sparse code matrix. The operator $\mathrm{row}_i(\mathbf{H})$ means the ith row vector of the matrix input $\mathbf{H}$. By taking the singular value decomposition $\mathbf{E}_i = \mathbf{U}\mathbf{\Delta}\mathbf{V}^\top$, rank-1 update replaces the new $\mathbf{d}_i$ with the first column of $\mathbf{U}$. This method is used in the inner-loop of the K-SVD dictionary learning algorithm [8].

Block-coordinate descent uses an update rule based on the following optimization

$$\mathbf{D} := \arg\min_{\mathbf{D}'} \frac{1}{2}\,\mathrm{Tr}(\mathbf{D}'^\top \mathbf{D}'\mathbf{A}) - \mathrm{Tr}(\mathbf{D}'^\top \mathbf{B}). \qquad (10)$$

This method is fast and efficient for online use [9]. The two auxiliary matrices $\mathbf{A}$ and $\mathbf{B}$ are initialized to zeros. Then, we update $\mathbf{A} := \mathbf{A} + \mathbf{y}^{*(j,k)}\mathbf{y}^{*(j,k)\top}$ and $\mathbf{B} := \mathbf{B} + \mathbf{x}^{(j,k)}\mathbf{y}^{*(j,k)\top}$ as we run more examples. We can run all examples in $\mathcal{D}_l$ or a selected subset for the block-coordinate descent updates.

Using the adjusted dictionary $\mathbf{D}^*$, we rerun the feedforward paths for all examples from $\mathcal{D}_l$ except the softmax regression

$$\{\mathbf{x}^{(j,k)}\}_{k=1}^{M_j} \xrightarrow{\mathbf{D}^*} \{\mathbf{y}^{\dagger(j,k)}\}_{k=1}^{M_j} \xrightarrow{\text{pool}} \mathbf{z}^{\dagger(j)}.$$

Using the newly computed pooled feature vectors $\mathbf{z}^{\dagger(j)}$'s, we retrain the softmax output layer. This completes a single iteration for the SCN backpropagation. Ideally, we run multiple iterations until convergence. Algorithm 2 presents the SCN backpropagation.

5. EXPERIMENTAL EVALUATION

In this section, we apply the sparse-coded net model for multiclass classification tasks over synthetic speech, CIFAR-10 image, and Wikipedia text datasets. For a metric, we average the softmax classification accuracy for each class c

$$\text{accuracy} = \frac{1}{m}\sum_{c=1}^{m} \frac{tp_c + tn_c}{\text{total \# of examples in } c}.$$

Algorithm 2 *SCN backpropagation*

```
 1: input: $\mathcal{D}_l = \{(\mathbf{X}^{(j)} = [\mathbf{x}^{(j,k)}]_{k=1}^{M_j}, l^{(j)})\}_{j=1}^{n_l}$
 2: output: fine-tuned classifier $C$ or regression $R$
 3: require: pretrain SCN by Algorithm 1
 4: repeat
 5:     adjust pooled feature vectors $\mathbf{z}^{(j)}$ via Eq. (6)
 6:     adjust sparse codes $\mathbf{y}^{(j,k)}$ via putback or Eq. (7)
 7:     adjust sparse coding dictionary $\mathbf{D}$ via Eq. (9)
        or rank-1 updates or via Eq. (10)
 8:     do sparse coding with adjusted dictionary $\mathbf{D}^*$
 9:     do pooling over recomputed sparse codes $\mathbf{y}^{\dagger(j,k)}$
10:     retrain softmax layer with new feature vectors $\mathbf{z}^{\dagger(j)}$
11: until convergence
```

Table 1: Speaker recognition performance on synthetic corpus.

Method	Accuracy
Semi-supervised via sparse coding (LARS)	37.9%
Semi-supervised via sparse coding (OMP)	32.2%
GMM-SVM	30.5%
SAE NN (4 layers)	44.8%
Sparse-coded net (LARS)	**51.3**%
Sparse-coded net (OMP)	**46.3**%

In reporting the sparse-coded net performance, we take the best result from the three update methods during the backpropagation. We also report the accuracy of semi-supervised learning with sparse coding, as well as Gaussian mixture model with support vector machine (GMM-SVM) and deep stacked autoencoder (SAE) neural networks [10] for comparison. We consider both forms of sparse coding, the ℓ_1-regularized LARS [4] and greedy-ℓ_0 OMP [5]. Throughout our experiments, we use a sparsity parameter $\lambda = 0.15$ for LARS and a sparsity bound $S = 0.2 \times \dim(\mathbf{x})$. Our implementation is done in MATLAB, using the INRIA SPAMS (SPArse Modeling Software) [11] for LARS and the Technion toolbox [12] for OMP. We use LIBSVM [13] for GMM-SVM.

5.1. Synthetic speech corpus

We use a synthetic speech corpus generated by the MSR Identity Toolbox for speaker recognition research [14]. The synthetic corpus is designed for a small-scale speaker recognition evaluation (SRE) that consists of 100 speakers. Each speaker has 50 utterances. All speech utterances are 1,000 frames long with a 10-msec frame duration. The tested methods take in 32-dimensional MFCC features per frame. We preprocess the MFCC features using PCA-whitening without dimensionality reduction. Table 1 reports the speaker recognition performance on the synthetic speech corpus. The accuracy improvements (about 14%) by sparse-coded net models on LARS and OMP are highlighted.

Table 2: Image classification performance on CIFAR-10.

Method	Accuracy
Semi-supervised via sparse coding (LARS)	80.1%
Semi-supervised via sparse coding (OMP)	76.8%
GMM-SVM	74.6%
SAE NN (4 layers)	78.9%
Sparse-coded net (LARS)	**83.1**%
Sparse-coded net (OMP)	**79.6**%

Table 3: Text classification performance on Wikipedia dataset.

Method	Accuracy
Semi-supervised via sparse coding (LARS)	69.4%
Semi-supervised via sparse coding (OMP)	61.1%
SAE NN (4 layers)	67.1%
Sparse-coded net (LARS)	**70.2**%
Sparse-coded net (OMP)	**62.1**%

5.2. CIFAR-10

CIFAR-10 [15] is a dataset of 60,000 32x32 color images belonging to 10 classes such as airplane, automobile, cat, and dog. As a preliminary evaluation, we prepare 2 subsets for training and test by uniform sampling. The train dataset has 2,000 images while the test dataset has 4,000 images. For each image, we draw densely overlapping patches using a receptive field with width $w = 6$ pixels and stride $s = 2$ (*i.e.*, resulting patches have a dimension $N = 3 \times 6 \times 6 = 108$). We preprocess the patches by ZCA-whitening before sparse coding. Table 2 presents the image classification performance on CIFAR-10. Again, we observe consistent accuracy improvements by the sparse-coded net models on LARS and OMP.

5.3. Wikipedia

The Wikipedia dataset consists of 2,866 documents. Each document has a variable number of paragraphs in English about a subject from one of the 10 categorical classes: art, biology, geography, history, literature, media, music, royalty, sport, and warfare. To focus on the text classification performance, we ignore all images in the dataset. Each text is represented by a histogram of the 10-topic latent Dirichlet allocation (LDA) model [16]. We apply these LDA features as input for sparse coding. Table 3 presents the text classification performance on the Wikipedia dataset. The SCN joint optimization of sparse coding, pooling, and classifier improves the categorical text classification performance by semi-supervised learning for both LARS and OMP.

6. CONCLUSION

Good feature learning performance of sparse coding has enabled high-performance classification pipelines for text and multimedia data. In this paper, we have presented Sparse-coded Net model that enhances semi-supervised learning with sparse coding. Our key idea is to tightly integrate sparse coding and dictionary learning with a supervised task at the output layer such as softmax regression. After pretraining a sparse-coded net via semi-supervised learning, we can optimize its task-specific objectives with a novel backpropagation algorithm that can effectively traverse nonlinear feature pooling to further update the dictionary trained only by unlabeled examples. We have tested sparse-coded nets for multi-class classification problems in speech, image, and text data. Our preliminary evaluation confirms the superior classification accuracy of sparse-coded net in small to medium-sized settings. In the future work, we will cover the SCN backpropagation hyperparameter optimization more comprehensively using broader datasets for sound, video, and wireless signals.

7. REFERENCES

[1] B. Olshausen and D. Field, "Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?," *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[2] A. Coates, A. Ng, and H. Lee, "An Analysis of Single-layer Networks in Unsupervised Feature Learning," in *AISTATS*, 2011.

[3] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of Royal Statistical Society*, vol. 58, pp. 267–288, 1994.

[4] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least Angle Regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.

[5] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition," in *Asilomar Conference on Signals, Systems and Computers*, 1993.

[6] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, The MIT Press, 2010.

[7] G. Hinton, S. Osindero, and Y. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[8] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *IEEE Trans. on Sig. Proc.*, vol. 54, no. 11, 2006.

[9] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online Dictionary Learning for Sparse Coding," in *ICML*, 2009.

[10] 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," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[11] INRIA, "Sparse modeling software," `http://spams-devel.gforge.inria.fr/`.

[12] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit," Tech. Rep., 2008.

[13] C. Chang and C. Lin, "LIBSVM: Library for Support Vector Machines," *ACM Trans. on Intelligent Systems and Technology*, 2011.

[14] S. Sadjadi, M. Slaney, and L. Heck, "MSR Identity Toolbox: A MATLAB Toolbox for Speaker-Recognition Research," *Speech and Language Processing Technical Committee Newsletter*, November 2013.

[15] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Tech. Rep., University of Toronto, 2009.

[16] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," in *JMLR*, 2003.