Sparse Coding with Structured Sparsity Priors and Multilayer Architecture for Image Classification

by

Xiaoxia Sun

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

May, 2017

Abstract

Applying sparse coding on large dataset for image classification is a long standing problem in the field of computer vision. It has been found that the sparse coding models exhibit disappointing performance on these large datasets where variability is broad and anomalies are common. Conversely, deep neural networks thrive on bountiful data. Their success has encouraged researchers to try and augment the learning capacity of traditionally shallow sparse coding methods by adding layers. Multilayer sparse coding networks are expected to combine the best of both sparsity regularizations and deep architectures. To date, however, endeavors to marry the two techniques have not achieved significant improvements over their individual counterparts.

In this thesis, we first briefly review multiple structured sparsity priors as well as various supervised dictionary learning techniques with applications on hyperspectral image classification. Based on the structured sparsity priors and dictionary learning techniques, we then develop a novel multilayer sparse coding network that contains 13 sparse coding layers. The proposed sparse coding network learns both the dictionaries and the regularization parameters simultaneously using an end-to-end supervised learning scheme. We

ABSTRACT

show empirical evidence that the regularization parameters can adapt to the given training data. We also propose applying dimension reduction within sparse coding networks to dramatically reduce the output dimensionality of the sparse coding layers and mitigate computational costs. Moreover, our sparse coding network is compatible with other powerful deep learning techniques such as drop out, batch normalization and shortcut connections. Experimental results show that the proposed multilayer sparse coding network produces classification accuracy competitive with the deep neural networks while using significantly fewer parameters and layers.

Primary Reader: Professor Trac D. Tran

Secondary Reader: Professor Mark A. Foster

Acknowledgments

It is a significant stage of my life to spend four years to work on my PhD degree, and I am grateful and honored to work with thoughful and brilliant people during this period. First and foremost, I would like to thank my advisor Prof. Trac D. Tran, for introducing me to a wonderful land of sparse representation and providing me a free atmosphere for conducting the research I am interested in. I have received tremendous help from the first moment I came to the Johns Hopkins and I have never imagined a more enjoyable graduate school life without his guidance.

I am deeply grateful to Prof. Nasser M. Nasrabadi from the West Virginia University for his guidance and support over the past five years. His patience and kindness have guided my graduate school life with the right trail. He has encouraged me to explore new frontiers in both the field of sparse coding and deep learning, and has been great helpful over past few years whenever I encounter difficulties, both at campus and in my daily life. He has trained me in various aspects of my research and I always feel inspired to work with him. This thesis would not be completed without his help.

Throughout the graduate years, I am glad to work with many insightful colleages and

collaborators: Dr. Sang Peter Chin at Boston University, Dr. Heesung Kwon at Army Research Laboratory (ARL), Dr. Chiman Kwan at Signal Processing, Inc. Prof. Jerry L. Prince at JHU's Image Analysis and Communications Lab, Prof. Mark A. Foster at Ultrafast and Nonlinear Photonics Lab, Dr. Chengjie Tu at Uber and many others. I would also thank Prof. Quan Quan, my undergraduate advisor, who directed me to the correct path of proceeding research in the field of computer vision.

It has been an enjoyable time to work in the DSP lab, and I would like to thank the following labmates for their valuable and inspiring comments: Dr. Minh D. Dao, Dr. Yuanming Suo, Dr. Shuai Huang, Qing Qu, Joy Sonia, Dung N. Tran, Tao Xiong, Luoluo Liu, Akshay Rangamani, Xiang Xiang, and Arun Nair. I am also glad to work as a member of SOAR program in ARL, and I am thankful for the insightful discussion with the following colleagues: Dr. Zhaowen Wang, Dr. Tianpei Xie, Dr. Soheil Bahrampour, Dr. Jingwei Ye, Christopher Reale, Hao Wu, Boyu Lu, Ding Liu, Muchen Wu, and many others.

Finally but most importantly, many deepest gratitude to my parents for I have been deeply indebted to their endless love and selfless endurance. It is my great honor to dedicate this thesis for them.

Dedication

To my parents, Zhen Sun & Ying Zhang.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Motivation

A long standing and fundamental problem in computer vision is image classification [1, 2, 3, 4], where an image or a single pixel is labeled to a specified class according to its visual content. For example, given a natural image, one would like to know whether it contains an automobile or not [5]. Given a hyperspectral image, one would like to decide whether a specified hyperpixel belongs to the material of land or grass [4, 6, 7, 8]. It is well known that human visions are particularly good at dealing with such classification problems by demonstrating both quick response and high accuracy. Human is able to quickly learn from very few training samples and give the correct answer based on their life long experience, whereas most computer vision systems demand tremendous amount of labeled samples to achieve a comparable performance. In most cases, human vision is so reliable and superior that the researchers usually regard human vision as a guidance to devise the computer vision system. For a long period of time, consistent efforts have been endeavored

to investigate and design computer vision system that could be competitive with human vision performance in both classification accuracy and computational efficiency.

Years ago, manually engineered features such as SIFT [9] and HOG [10] once dominated the land of computer vision. These delicately engineered features have demonstrated a rather strong competence of distilling core essence from raw image pixels. However, with the advent of the big data era, these engineered features quickly become outdated for they cannot learn from the abundant amount of readily-available training samples. Therefore, recent years have seen an explosion of interests on developing algorithms that can extract learnable features from large dataset [3, 11, 12, 13]. Neural networks with multilayer architecture, usually referred to as deep learning, have resurged and occupied most of the research activities. Unlike the engineered features or any single layer model, deep architecture models have enough learning capacities to extract and absorb the large amount of representative information from big dataset. However, the deep neural network is prone to severe overfitting and therefore suffer from the data hungry issue even after trained with a huge amount of labeled samples. One notable trend in deep learning is to enforce various regularizations on the network, such as sparsity constraint [14], drop out [15] and batch normalization [16]. In addition, numbers of efforts have been attempted at enforcing the reconstruction constraint on the network, i.e. developing a generative model of the neural network. The intuition behind the generative approach follows a famous quote from Richard Feymann,

What I cannot create, I do not understand.

—Richard Feymann

In such cases, the network is trained with a loss function that is characterized by both discriminative and reconstructive constraints.

The formulation of such loss function reminds us with the sparse coding immediately: in the land of sparse coding, the loss function of sparse recovery is defined by a delicate balance of reconstruction error and sparsity level. In computer vision, sparse coding has already been successfully applied to numerous computer vision tasks, including face recognition [17, 18], scene categorization [19, 20, 21] and object detection [22, 23, 24]. Application of sparse representation-based classifier (SRC) on face recognition [18] has demonstrated startling robustness over noise and occlusions. Due to the powerful sparsity prior, the sparse coding is less likely to become overfitted and therefore requiring much fewer samples for training. More importantly, sparse coding can be easily trained in a unsupervised fashion, making it less demanded for the labeled training sample and can be trained on any unlabeled data. Therefore, extending the single-layer sparse coding model to a multilayer architecture can largely improve the performance of image classification on large datasets.

Due to the scale of topics involved in this thesis, the background and literature reviews are left to each individual chapter, which will be a self-containing part with discussion on how it fits in the overall theme of this thesis. Developing deep sparse coding network is nontrvial and require several key techniques in which this thesis will address:

- Evaluation and better understanding of sparse recovery algorithms with various structured sparsity priors (Chapter 2).

- Development of supervised dictionary for sparse coding with various structured sparsity priors (Chapter 3).
- Enforcing invariant property on sparse coding model using large displacement optical flow (Chapter 4).
- Greedy layer-wise unsupervised dictionary learning with invariant sparse coding (Chapter 5).
- Development of end-to-end supervised dictionary learning (Chapter 6).

Related papers of my previous work have been presented in smaller parts over a course of papers [25, 26, 27, 28, 29, 30, 31]. The ultimate goal for this thesis is to develop a sparse coding model with multilayer architecture that is able to show competitive performance with deep neural network (Chapter 6). As the dissertation goes through, we shall see that the structured sparsity priors, supervised dictionary learning and invariant sparse coding are the step stones that bring us to the approach of supervised deep sparse coding networks. We show how each individual component contributes to the overall contributions as follows:

Chapter 2 focuses on reviewing and developing various structured sparsity priors, which is critical for understanding how various sparsity patterns affect the classification performance. Instead of enforcing sparse regularizor element-wise, the structured sparsity prior regards the sparse codes as an entity in order to exploit more sophisticated structures of both the dictionary and the sparse code. More specifically, we review four structured sparsity priors including joint sparsity prior, Laplacian sparsity prior, group sparsity prior,

sparse group sparsity prior, and propose low rank group sparsity prior that is able to consistently improve the classification performance on hyperspectral image. We also show that the classification performance is not only determined by the structured priors, but also largely depends on the sparse recovery algorithms.

Chapter 3 further develops the supervised dictionary learning algorithm for various structured sparsity priors. The adoption of supervised dictionary learning is able to significantly reduce the required dictionary size for sparse recovery in order to substantially supress the computational cost. Optimizing the supervised dictionary learning problem is non-trivial due to the implicit relation between the sparse code and the dictionary. Mathematically, this chapter will show how to unravel the sparse code out of the structured sparsity regularizor by decoupling the sparse code and the dictionary using fixed point differentiation. As we shall see, the development of these dictionary learning algorithms paves the way for training dictionaries for invariant sparse coding.

Having discussed the supervised dictionary learning algorithms, **Chapter 4** moves on to a higher level to exploit the possible ways to generate sparse codes that are invariant to major transformations of the objects in the image. The poor generalization performance of sparse coding towards affine transformation motivates us to enforce reasonable manipulations on the dictionary atoms in order to produce invariant sparse codes. Specifically, I propose to employ the large displacement optical flow for the purpose of describing the misalignment between each dictionary atom and the testing image in order to align every single dictionary atom according to the optical flow field. The corresponding dictionary

is trained efficiently based on supervised task-driven dictionary learning and bilevel optimization.

Chapter 5 further extends the invariant sparse coding to a multilayer architecture, where the deep architecture is able to significantly increase the learning capacity of the sparse coding model. In this chapter, the local sparse coding is adopted to improve the computational efficiency and the invariant property is enforced through the employment of *bag of atoms*, which is inspired by the bag of words model. This chapter benefits from the invariant sparse coding and the task-driven dictionary learning presented in Chapter 4, and a layer-wise unsupervised dictionary learning algorithm is developed for the proposed invariant sparse coding framework, which is able to simultaneously reduce the reconstruction errors of both the sparse recovery and the local feature descriptor matching.

I present the supervised multilayer sparse coding network with 13 sparse coding layers in **Chapter 6**. To the best of my knowledge, this is the first time sparse coding is efficiently extended to a deep architecture with more than two layers while exhibiting a state-of-the-art performance. The proposed multilayer sparse coding network is capable of efficiently adapting its own regularization parameters to a given dataset, and is trained end-to-end with a supervised task-driven learning algorithm via error backpropagation. Furthermore, a sparse coding layer utilizing a 'skinny' dictionary is also devised. Integral to computational efficiency, these skinny dictionaries compress the high dimensional sparse codes into lower dimensional structures. This chapter will show that our multilayer architecture overwhelmingly outperforms traditional one-layer sparse coding architectures while using

much fewer parameters.

Last but not least, I draw the conclusion in **Chapter 7** summarizing the contribution of the dissertation and discuss my future works.

Chapter 2

Sparse Coding with Structured Sparsity Priors

In this chapter, we consider the problem of sparse representation with various structured sparsity priors with application on the hyperspectral image (HSI) classification. By representing a test pixel as a linear combination of a small subset of labeled pixels, a sparse representation classifier (SRC) gives rather plausible results compared with that of traditional classifiers such as the support vector machine (SVM). Recently, by incorporating additional structured sparsity priors, the second generation SRCs have appeared in the literature and are reported to further improve the classification performance of HSI. These priors are based on exploiting the spatial dependencies between the neighboring pixels, the inherent structure of the dictionary, or both. In this chapter, we review and compare several structured priors for sparse-representation-based image classification. We also propose a

new structured prior called the low rank group prior, which can be considered as a modification of the low rank prior. Furthermore, we will investigate how different structured priors improve the result for the image classification.

2.1 Introduction

In the image classification of HSI, each individual pixel is labeled to one of the classes based on its spectral characteristics. Due to the numerous demands in mineralogy, agriculture and surveillance, the HSI classification task is developing very rapidly and a large number of techniques have been proposed to tackle this problem [32]. Comparing with previous approaches, SVM is found highly effective on both computational efficiency and classification results. A wide variety of SVM's modifications have been proposed to improve its performance. Some of them incorporate the contextual information in the classifiers [33, 34]. Others design sparse SVM in order to pursue a sparse decision rule by using ℓ_1-norm as the regularizer [35].

Recently, SRC has been proposed to solve many computer vision tasks [18, 36], where the use of sparsity as a prior often leads to state-of-the-art performance. SRC has also been applied to HSI classification [37], relying on the observation that hyperspectral pixels belonging to the same class approximately lie in the same low-dimensional subspace. In order to alleviate the problem introduced by the lack of sufficient training data, Haq *et al.* [38] proposed the homotopy-based SRC. Another way to solve the problem of insufficient

training data is to employ the contextual information of neighboring pixels in the classifier, such as spectral-spatial constraint classification [39].

In SRC, a test sample $\mathbf{x} \in \mathbf{R}^M$, where M is the number of spectral bands, can be written as a sparse linear combination of all the training pixels (atoms in a dictionary) as

$$\hat{\boldsymbol{\alpha}} = \min_{\mathbf{z}} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda\|\mathbf{z}\|_1, \tag{2.1}$$

where $\mathbf{z} \in \mathbf{R}^N$, $\|\mathbf{z}\|_1 = \sum_{i=1}^{N} |z_i|$ is ℓ_1-norm. $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_N]$ is a structured dictionary formed from concatenation of several class-wise sub-dictionaries, $\{\mathbf{d}_i\}_{i=1,\ldots,N}$ are the columns of $\mathbf{D}$ and N is the total number of training samples from all the K classes, and λ is a scalar regularization parameter.

The class label for the test pixel $\mathbf{x}$ is determined by the minimum residual between $\mathbf{x}$ and its approximation from each class-wise sub-dictionary:

$$class(\mathbf{x}) = \arg\min_{g} \|\mathbf{x} - \mathbf{D}\delta_g(\boldsymbol{\alpha})\|_2^2, \tag{2.2}$$

where $g \subset \{1, 2, \cdots, K\}$ is the group or class index, and $\delta_g(\boldsymbol{\alpha})$ is the indicator operation zeroing out all elements of $\boldsymbol{\alpha}$ that do not belong to the class g.

In the case of HSI, SRC always suffers from the non-uniqueness or instability of the sparse coefficients due to the high mutual coherency of the dictionary [40]. Due to these undesired properties of the HSI dictionary, the sparse recovery can become unstable and unpredictable such that even pixels belonging to the same class can have totally different

sparse codes. Fortunately, a better reconstructed signal and a more robust representation can be obtained by either exploring the dependencies of neighboring pixels or exploiting the inherent dictionary structure. The problem induced by the high-coherency of the dictionary atoms can also be alleviated through decreasing the variation between the sparse codes of the hyperspectral pixels that belong to the same class. Recently, structured priors have been incorporated into HSI classification [37], which can be sorted into three categories. (*a*) Priors that only exploit the correlations and dependencies among the neighboring spectral pixels or their sparse coefficient vectors, which includes joint sparsity [41], graph regularized Lasso (referred as the Laplacian regularized Lasso) [42] and the low-rank Lasso [43]. (*b*) Priors that only exploit the inherent structure of the dictionary, such as group Lasso [44]. (*c*) Priors that enforce structural information on both sparse coefficients and dictionary, such as collaborative group Lasso [45] and collaborative hierarchical Lasso (CHiLasso) [46]. Besides SRC, structured sparsity prior can also be incorporated into other classifiers such as the logistic regression classifiers [47].

In HSI, pixels that are spatially close to each other usually have similar spectral features and belong to the same class. The sparse codes of neighboring pixels can become similar by enforcing a structured sparsity constraint (prior). The simultaneous sparse recovery is analytically guaranteed to achieve a sparser solution and a lower reconstruction error with a smaller dictionary [41]. A variety of structured sparsity priors are proposed in the literature [25] that are capable of generating different desired sparsity patterns for the sparse codes of neighboring pixels. The joint sparsity prior [37] assumes that the features of all

the neighboring pixels lie in the same low dimensional subspace and all the corresponding sparse codes share the same set of dictionary atoms. Therefore, the sparse codes have a row sparsity pattern, where only a few rows of the sparse codes are nonzero [48, 49]. The collaborative group sparsity prior [45] enforces the coefficients to have a group-wise sparsity pattern, where the coefficients within each active group are dense. The collaborative hierarchical sparsity prior [46] enforces the sparse codes to be not only group-wise sparse, but also sparse within each active group. The low rank prior [43] assumes that the neighboring pixels are linearly dependent. It does not necessary lead the coefficients to be sparse, which is detrimental for a good classification. However, the low rank group prior proposed in [25] is able to enforce both a group sparsity prior and a low rank prior on the sparse codes by forcing the same group of dictionary atoms to be active if and only if the corresponding neighboring pixels are linearly dependent. The Laplacian sparsity prior [42] uses a Laplacian matrix to describe the degree of similarity between the neighboring pixels. The neighboring pixels that have less spectral features in common are less encouraged to have a similar sparse codes. It has been shown that all the structured sparsity priors are capable of obtaining a smoother classification map and improving the classification performance [25].

The main contributions of this chapter are *(a)* to assess the SRC performance using various structured sparsity priors for HSI classification, and *(b)* to propose a conceptually similar prior to CHiLasso, which is called the low-rank group prior. This prior is based on the assumption that pure or mixed pixels from the same classes are highly correlated and can be represented by a combination of sparse low-rank groups (classes). The proposed

prior takes advantage of both the group sparsity prior, which enforces sparsity across the groups, and the low rank prior, which encourages sparsity within the groups, by only using one regularizer.

In the following sections, we investigate the roles of different structured priors imposed on the SRC optimization algorithm. Starting with the classical sparsity ℓ_1-norm prior, we then introduce several different priors with experimental results. The structured priors discussed are joint sparsity, Laplacian sparsity, group sparsity, sparse group sparsity, low-rank and low-rank group prior.

2.2 HSI Classification via Different Structured Sparse Priors

2.2.1 Joint Sparsity Prior

In HSI, pixels within a small neighborhood usually consist of similar materials. Thus, their spectral characteristics are highly correlated. The spatial correlation between neighboring pixels can be indirectly incorporated through a joint sparsity model (JSM) [48] by assuming that the underlying sparse vectors associated with these pixels share a common sparsity support. Consider pixels in a small neighborhood of T pixels. Let $\mathbf{X} \in \mathbf{R}^{M \times T}$ represent a matrix whose columns correspond to pixels in a spatial neighborhood in a hyperspectral image. Columns of $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]$ can be represented as a linear com-

bination of dictionary atoms $\mathbf{X} = \mathbf{DA}$, where $\mathbf{A} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_T] \in \mathbf{R}^{N \times T}$ represents a sparse matrix. In JSM, the sparse vectors of T neighboring pixels, which are represented by the T columns of $\mathbf{A}$, share the same support. Therefore, $\mathbf{A}$ is a sparse matrix with only few nonzero rows. The row-sparse matrix X can be recovered by solving the following Lasso problem

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{X} - \mathbf{DA}\|_F^2 + \lambda\|\mathbf{A}\|_{1,2}, \tag{2.3}$$

where $\|\mathbf{A}\|_{1,2} = \sum_{i=1}^{N} \|\boldsymbol{\alpha}^i\|_2$ is an $\ell_{1,2}$-norm and $\boldsymbol{\alpha}^i$ represents the ith row of $\mathbf{A}$.

The label for the center pixel $\mathbf{x}_c$ is then determined by the minimum total residual error

$$class(\mathbf{x}_c) = \arg\min_{g} \|\mathbf{X} - \mathbf{D}\delta_g(\mathbf{A})\|_F^2, \tag{2.4}$$

where $\delta_g(\mathbf{A})$ is the indicator operation zeroing out all the elements of $\mathbf{A}$ that do not belong to the class g.

2.2.2 Laplacian Sparsity Prior

In sparse representation, due to the high coherency of the dictionary atoms, the recovered sparse coefficient vectors for multiple neighboring pixels could be partially different even when the neighboring pixels are highly correlated, and this may led to misclassification. As mentioned in the previous section, joint sparsity is able to solve such a problem by enforcing multiple pixels to select exactly the same atoms. However, in many cases, when the neighboring pixels fall on the boundary between several homogeneous regions,

the neighboring pixels will belong to several distinct classes (groups) and should use different sets of sub-dictionary atoms. Laplacian sparsity enhances the differences between sparse coefficient vectors of the neighboring pixels that belong to different clusters. We introduce the weighting matrix $\mathbf{W}$, where $\mathbf{w}_{ij}$ characterizes the similarity between a pair of pixels x_i and x_j within a neighborhood. Optimization with an additional Laplacian sparsity prior can be expressed as

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda_1\|\mathbf{A}\|_1 + \lambda_2 \sum_{i,j} w_{ij}\|\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_j\|_2^2, \tag{2.5}$$

where λ_1 and λ_2 are the regularization parameters. The matrix $\mathbf{W}$ is used to characterize the similarity among neighboring pixels in the spectra space. Similar pixels will possess larger weights, and therefore, enforcing the differences between the corresponding sparse coefficient vectors to become smaller, and similarly allowing the difference between sparse coefficient vectors of dissimilar pixels to become larger. Therefore, the Laplacian sparsity prior is more flexible than the joint sparsity prior in that it does not always force all the neighboring pixels to have the same common support. In this chapter, the weighting matrix is computed using the sparse subspace clustering method in [50]. Note that this grouping constraint is enforced on the testing pixels instead of the dictionary atoms, which is different from group sparsity. Let $\mathbf{L} = \mathbf{I} - \mathbf{G}^{-1/2}\mathbf{W}\mathbf{G}^{-1/2}$ be the normalized symmetric Laplacian matrix [50], where $\mathbf{G}$ is the degree matrix computed from $\mathbf{W}$. We can rewrite

the equation (2.5) as

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{X} - \mathbf{DA}\|_F^2 + \lambda_1\|\mathbf{A}\|_1 + \lambda_2 tr(\mathbf{ALA}^T). \tag{2.6}$$

The above equation can be solved by a modified feature-sign search algorithm [42].

2.2.3 Group Sparsity Prior

The SRC dictionary has an inherent group-structured property since it is composed of several class sub-dictionaries, i.e., the atoms belonging to the same class are grouped together to form a sub-dictionary. In sparse representation, we classify pixels by measuring how well the pixels are represented by each sub-dictionary. Therefore, it would be reasonable to enforce the pixels to be represented by groups of atoms instead of individual ones. This could be accomplished by encouraging coefficients of only certain groups to be active and the remaining groups inactive. Group Lasso [44], for example, uses a sparsity prior that sums up the Euclidean norm of every group coefficients. It will dominate the classification performance especially when the input pixels are inherently mixed pixels. Group Lasso optimization can be represented as

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \sum_{g \in G} w_g \|\boldsymbol{\alpha}_g\|_2, \tag{2.7}$$

where $g \subset \{G_1, G_2, \cdots, G_K\}$, $\sum_{g \in G} \|\boldsymbol{\alpha}_g\|_2$ represents the group sparse prior defined in terms of K groups, w_g is the weight and is usually set to the square root of the cardinality of the corresponding group to compensate for the different group sizes. Here, $\boldsymbol{\alpha}_g$ refers to the coefficients of each group. The above group sparsity can be easily extended to the case of multiple neighboring pixels by extending problem (2.7) to collaborative group Lasso, which is formulated as

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda \sum_{g \in G} w_g \|\mathbf{A}_g\|_2, \tag{2.8}$$

where $\sum_{g \in G} \|\mathbf{A}_g\|_2$ represents a collaborative group Lasso regularizer defined in terms of group and $\mathbf{A}_g$ refers to each of the group coefficient matrix. When the group size is reduced to one, the group Lasso degenerates into a joint sparsity Lasso.

2.2.4 Sparse Group Sparsity Prior

In the formulations (2.7) and (2.8), the coefficients within each group are not sparse, and all the atoms in the selected groups could be active. If the sub-dictionary is overcomplete, then it is necessary to enforce sparsity within each group. To achieve sparsity within the groups, an ℓ_1-norm regularizer can be added to the group Lasso (2.7), which can be written as

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda_1 \sum_{g \in G} w_g \|\boldsymbol{\alpha}_g\|_2 + \lambda_2 \|\boldsymbol{\alpha}\|_1. \tag{2.9}$$

Similarly, Eq. (2.9) can be easily extended to the multiple feature case, which can be

written as

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda_1 \sum_{g \in G} w_g \|\mathbf{A}_g\|_2 + \lambda_2 \sum_{g \in G} w_g \|\mathbf{A}_g\|_1. \tag{2.10}$$

Optimization problem (2.9) is referred in the literature as the sparse group Lasso and problem (2.10) as the collaborative hierarchical Lasso (CHiLasso) [46].

2.2.5 Low Rank/Group Sparsity Prior

Based on the fact that spectra of neighboring pixels are highly correlated, it is reasonable to enforce the low rank sparsity prior on their coefficient matrix. The low rank prior is more flexible when compared with the joint sparsity prior which strictly enforces the row sparsity. Therefore, when neighboring pixels are composed of small non-homogeneous regions, the low rank sparsity prior outperforms the joint sparsity prior. Low rank sparse recovery problem has been well studied in [43] and is stated as the following Lasso problem

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda \|\mathbf{A}\|_*, \tag{2.11}$$

where $\|\mathbf{A}\|_*$ is the nuclear norm [43].

To incorporate the structure of the dictionary, we now extend the low rank prior to group low rank prior, where the regularizer is obtained by summing up the rank of every group

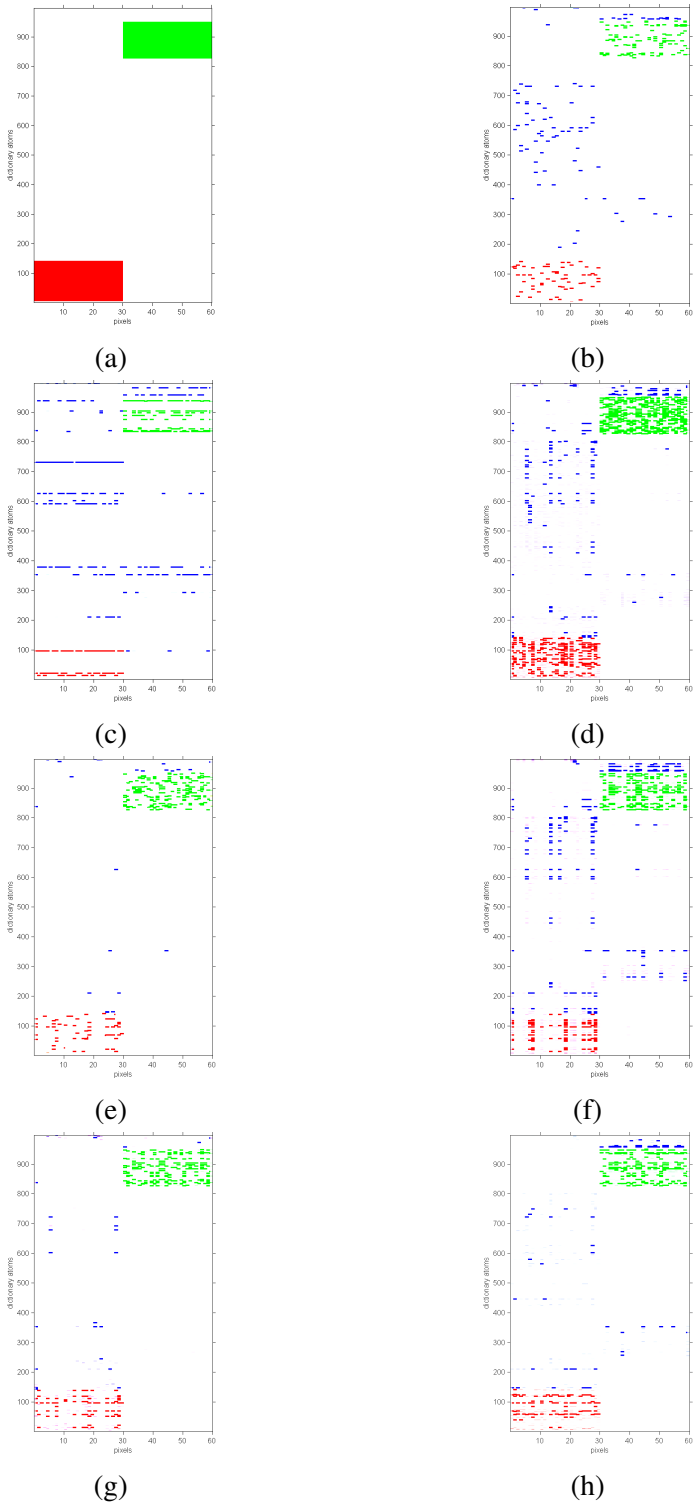


Figure 2.1: Sparsity patterns for the toy example: (a) desired sparsity regions, (b) ℓ_1 minimization using ADMM, (c) joint sparsity, (d) collaborative group sparsity, (e) collaborative sparse group sparsity, (f) low rank sparsity, (g) low rank group sparsity and (h) Laplacian sparsity via FFS.

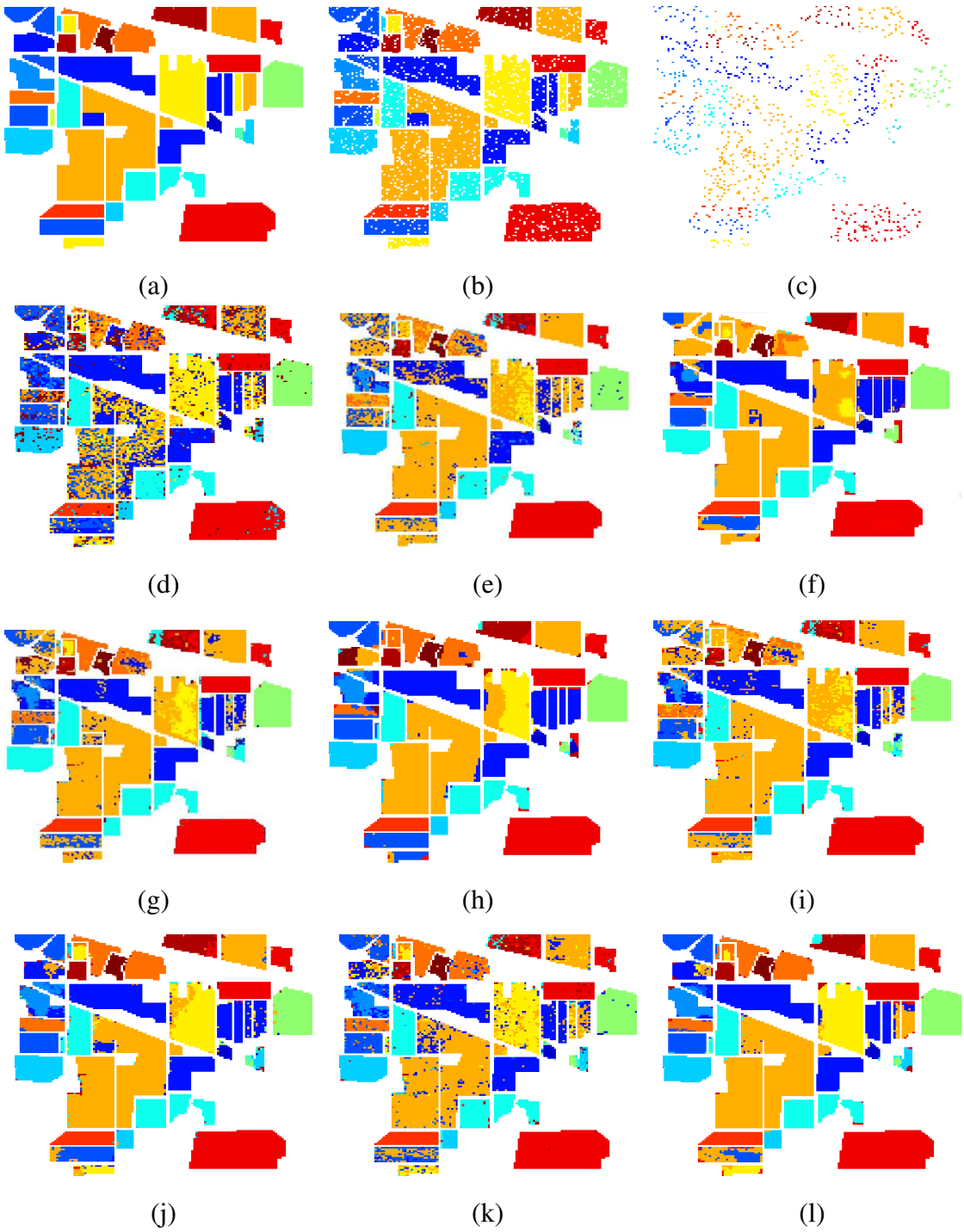


Figure 2.2: Results for the Indian Pine image: (a) ground truth, (b) training set and (c) test set. Classification map obtained by (d) SVM, (e) ℓ_1-minimization using ADMM, (f) joint sparsity, (g) collaborative group sparsity, (h) collaborative sparse group sparsity, (i) low rank sparsity, (j) low rank group sparsity, (k) ℓ_1 minimization via FSS and (l) Laplacian sparsity via FSS.

coefficient matrix,

$$\min_{\mathbf{A}} \frac{1}{2}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda \sum_{g \in G} w_g \|\mathbf{A}_g\|_*. \tag{2.12}$$

The low rank group prior is able to obtain the within-group sparsity by minimizing the nuclear norm of each group. Furthermore, the summation of nuclear norms empowers the proposed prior to obtain a group sparsity pattern. Hence, the low rank group prior is able to achieve sparsity both within and across groups by using only one regularization term.

Table 2.1: Number of training and test samples for the Indian Pine image

Class	Train	Test
1	6	48
2	137	1297
3	80	754
4	23	211
5	48	449
6	72	675
7	3	23
8	47	442
9	2	18
10	93	875
11	235	2233
12	59	555
13	21	191
14	124	1170
15	37	343
16	10	85
Total	997	9369

Table 2.2: Classification accuracy (%) for the Indian Pine image using 997 (10.64%) training samples

Optimization Techniques		ADMM/SpaRSA							Feature Sign Search	
Class	SVM	ℓ_1	JS	LS	GS	SGS	LR	LRG	ℓ_1	LS
1	77.08	68.75	79.17	85.42	79.17	87.50	75.00	**91.67**	66.67	83.33
2	84.96	58.84	81.94	81.34	80.62	79.92	78.60	81.71	74.42	**89.90**
3	62.67	24.40	56.67	47.35	62.13	76.13	29.87	**89.87**	69.87	78.38
4	8.57	49.52	27.62	49.76	37.14	54.29	15.24	67.62	64.76	**88.15**
5	77.18	81.88	85.46	83.96	84.79	82.55	82.10	83.45	91.72	**94.43**
6	91.82	96.88	98.36	97.48	**98.96**	98.36	98.21	98.36	97.02	98.52
7	13.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**69.57**	0.00
8	96.59	96.59	**100.00**	99.55	99.55	99.55	99.77	99.55	99.55	**100.00**
9	0.00	5.56	0.00	0.00	22.22	0.00	0.00	0.00	**61.11**	0.00
10	71.30	24.00	18.94	31.89	39.95	45.58	8.61	49.60	76.46	**87.43**
11	35.25	96.22	91.63	94.58	91.99	93.02	97.12	92.35	87.62	**98.84**
12	42.39	32.97	45.29	64.68	69.57	65.58	20.83	82.97	78.26	**91.71**
13	91.05	98.95	99.47	99.48	99.47	98.95	98.95	99.47	99.47	**100.00**
14	94.85	98.97	98.97	99.49	98.80	99.31	**99.83**	99.31	97.77	99.57
15	30.70	49.71	55.85	63.84	50.58	80.99	44.15	**89.47**	53.80	69.97
16	27.06	88.24	95.29	97.65	95.29	**98.82**	97.65	97.65	85.88	97.65
OA[%]	64.94	71.17	76.41	79.40	80.19	83.19	71.90	86.46	83.74	**92.58**
AA[%]	56.53	60.72	68.53	64.67	69.39	72.53	59.14	76.43	79.62	**79.87**
κ	0.647	0.695	0.737	0.712	0.781	0.807	0.695	0.843	0.833	**0.923**

2.3 Experimental Verification

2.3.1 Datasets and Dictionary Generation

We evaluate various structured sparsity priors on two different hyperspectral images and one toy example. The first hyperspectral image to be assessed is the Indian Pine, acquired by Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), which generates 220 bands, of which 20 noisy bands are removed before classification. The spatial dimension of this image is 145×145, which contains 16 ground-truth classes, as shown in Table I. We randomly choose 997 pixels (10.64% of all the labelled pixels) for constructing the dictionary and use the remaining pixels for testing. The second image is the University of Pavia, which is an urban image acquired by the Reflective Optics System Imaging Spectrometer (ROSIS), contains 610×340 pixels. It generates 115 spectral bands, of which 12 noisy

Table 2.3: Number of training and test samples for the University of Pavia image

Class	Train	Test
1	139	6713
2	137	1859
3	100	2107
4	133	3303
5	68	1310
6	135	4969
7	95	1261
8	131	3747
9	59	967
Total	997	42926

bands are removed. There are nine ground-truth classes of interests. For this image, we choose 997 pixels (2.32% of all the labelled pixels) for constructing the dictionary and the remaining pixels for testing, as shown in Table III. The toy example consists of two different classes (class 2 and 14 of the Indian Pine test set), and each class contains 30 pixels. The dictionary is the same as that for the Indian Pine. The toy example is used to evaluate the various sparsity patterns generated by the different structured priors.

2.3.2 Models and Methods

The tested structured sparse priors are: (*i*) joint sparsity (JS), (*ii*) Laplacian sparsity (LS), (*iii*) collaborative group sparsity (GS), (*iv*) sparse group sparsity (SGS), (*v*) low rank prior (LR) and (*vi*) low rank group prior (LRG), corresponding to Eqs. (7), (10), (12), (14), (16) and (17), respectively. For SRC, the parameters λ, λ_1 and λ_2 of different structured priors range from 10^{-3} to 0.1. Performance on the toy example will be visually examined by the difference between the desired sparsity regions and the recovered ones. For the two

Table 2.4: Classification accuracy (%) for the University of Pavia image using 997 (2.32%) training samples

Optimization Techniques		ADMM/SpaRSA							Feature Sign Search	
Class	SVM	ℓ_1	JS	LS	GS	SGS	LR	LRG	ℓ_1	LS
1	84.55	57.11	77.04	95.08	94.01	97.90	91.16	94.15	72.14	**95.85**
2	**82.45**	58.22	67.98	66.70	70.04	68.04	69.73	69.32	59.62	64.28
3	77.08	57.33	44.32	77.55	79.45	73.56	75.80	**79.73**	66.21	76.51
4	94.19	95.94	95.13	95.19	95.31	95.55	95.94	98.46	97.67	**98.97**
5	99.01	**100.00**	99.85	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	99.85	**100.00**
6	23.55	89.60	88.31	96.60	**100.00**	99.74	**100.00**	99.96	80.60	98.63
7	2.06	83.27	84.38	**96.59**	95.24	95.56	95.06	95.24	86.76	94.69
8	33.89	48,65	65.20	67.36	62.24	44.84	65.24	63.06	75.95	**95.76**
9	53.05	93.69	**99.59**	**99.59**	93.38	93.28	93.57	94.00	90.69	98.35
OA[%]	69.84	66.51	74.05	80.82	81.15	79.07	80.81	81.02	71.41	**81.84**
AA[%]	61.09	75.98	80.06	88.80	87.73	85.36	87.35	87.93	81.05	**91.45**
κ	0.569	0.628	0.681	0.758	0.675	0.624	0.611	0.66	0.672	**0.781**

hyperspectral images, classification performance is evaluated by the overall accuracy (OA), average accuracy (AA), and the κ coefficient measure on the test set. For each structured prior, we present the result with the highest overall accuracy using cross validation. A linear SVM is implemented for comparison, whose parameters are set in the same fashion as in [37].

In experiments, joint sparsity, group sparsity and low rank priors are solved by ADMM [51], while CHiLasso and Laplacian prior are solved by combining SpaRSA [52] and ADMM. In addition, in conformity with previous work [42], the Laplacian regularized Lasso is also solved by a modified feature sign search (FSS) method. In this chapter, we try to present a fair comparison among all priors. According to the optimization technique, we sort the structured priors into two categories: (*i*) priors solved by ADMM and SpaRSA and (*ii*) priors solved by FSS-based method. The first row of Table II and Table IV show the methods used to implement the sparse recovery for each structured prior.

Table 2.5: Computation time (in seconds) for the Indian Pine image

ADMM/SpaRSA							FFS	
ℓ_1	JS	LS	GS	SGS	LR	LRG	LS	ℓ_1
1124	1874	4015	2811	2649	4403	2904	1124	11628

2.3.3 Performance

Sparsity patterns of the toy example are shown in Fig. 2.1. The expected sparsity regions are shown in Fig. 2.1(a), where the y-axis labels the dictionary atom index and x-axis labels the test pixel index. The red and green regions correspond to the ideal locations of the active atoms for the class 2 and 14, respectively. Nonzero coefficients that belong to other classes are shown in blue dots. The joint sparsity, Fig. 2.1 (c), shows clear row sparsity pattern, but many rows are mistakenly activated. As expected, active atoms in Fig. 2.1 (d), (e) and (g) demonstrate group sparsity patterns. Comparing the GS (d) and SGS (e), it is observed that most of the atoms are deactivated within groups using SGS. The low rank group prior (g) demonstrates a similar sparsity pattern as that of SGS. For the Laplacian sparsity (h), similarity of sparse coefficients that belong to the same classes is clearly visible.

Table II and Fig. 2.2 show the performance of SRCs with different priors on the Indian Pine image. A spatial window of 9×9 ($T = 81$) is used since this image consists of mostly large homogeneous regions. Among SRCs with different priors, the worst result occurs when we use simple ℓ_1-ADMM. Joint sparsity prior gives better result than the low rank prior. This is due to the large areas of homogeneous regions in this image, which favors the joint sparsity model. The highest OA is given by the Laplacian sparsity prior via

FFS, such a high performance is partly contributed to the accurate sparse recovery of the feature sign search method. Both SGS and LRG outperform GS. We can see that among ADMM-based based methods, the low rank group prior yields the smoothest result. The computational time of various structured priors for Indian Pine image are shown in Table 2.5. Among ADMM/SpaRSA-based methods, LRG, GS and SGS take roughly similar time ($\sim$2500s) to process the image, while LR and JS require longer time ($\sim$4000s). LS via FFS significantly impedes the computational efficiency.

Results for the University of Pavia image are shown in Table IV. The window size for this image is 5×5 ($T = 25$) since many narrow regions are present in this image. The group sparsity prior gives the highest OA among the priors optimized by ADMM. The low rank sparsity prior gives a much better result than joint sparsity since this image contains many small homogeneous regions. The Laplacian sparsity prior via FFS gives the highest OA performance. However, the difference between performance of various structured priors is quite small.

2.4 Summary

This chapter reviews five different structured sparse priors and proposes a low rank group sparsity prior. Using these structured priors, classification results of SRCs on HSI are generally improved when compared with the classical ℓ_1 sparsity prior. The results have confirmed that the low rank prior is a more flexible constraint compared with the joint

sparsity prior, while the latter works better on large homogeneous regions. Imposing the group structured prior on the dictionary always gives higher overall accuracy compared with the ℓ_1 prior. We have also observed that the performance is not only determined by the structured priors, but also depend on the corresponding optimization techniques.

Chapter 3

Sparse Coding with Task-driven Dictionary Learning and Structured Sparsity Priors

In this chapter, we develop dictionary learning algorithms for sparse representation with various structured sparsity priors. As a generative model, sparse coding requires the dictionary to be highly redundant in order to ensure both a stable high sparsity level and a low reconstruction error for the signal. However, in practice, this requirement is usually impaired by the lack of labeled training samples. Fortunately, as we discussed in the previous chapter, the requirement for a redundant dictionary can be less rigorous if simultaneous sparse approximation is employed, which can be carried out by enforcing various structured sparsity constraints on the sparse codes of the neighboring pixels. In addition, numerous

works have shown that applying a variety of dictionary learning methods for the sparse representation model can also improve the classification performance. In this chapter, we highlight the task-driven dictionary learning algorithm, which is a general framework for the supervised dictionary learning method. We propose to enforce structured sparsity priors on the task-driven dictionary learning method in order to improve the performance of the hyperspectral classification. Our approach is able to benefit from both the advantages of the simultaneous sparse representation and those of the supervised dictionary learning. We enforce two different structured sparsity priors, the joint and Laplacian sparsity, on the task-driven dictionary learning method and provide the details of the corresponding optimization algorithms. Experiments on numerous popular hyperspectral images demonstrate that the classification performance of our approach is superior to sparse representation classifier with structured priors or the task-driven dictionary learning method.

3.1 Introduction

Numerous difficulties impede the improvement of image classification performance for HSI. For instance, the high dimensionality of HSI pixels introduce the problem of the 'curse of dimensionality' [53], and the classifier is always confronted with the overfitting problem due to the small number of labelled samples. Additionally, most hyperpixels are indiscriminative since they are undesirably highly coherent [54]. In the past few decades, numerous classification techniques, such as SVM [6], k-nearest-neighbor classifier [8],

multimodel logistic regression [55] and neural network [7], have been proposed to alleviate some of these problems to achieve an acceptable performance for HSI classification.

More recently, researchers have focused attention on describing the high dimensional data as a sparse linear combination of dictionary atoms. SRC has been applied to HSI classification by Chen *et. al.* [37], where a dictionary was constructed by stacking all the labelled samples. Success of SRC requires that the high dimensional data belonging to the same class to lie in a low dimensional subspace. The outstanding classification performance is due to the robustness of sparse recovery, which is largely provided by the high redundancy and low coherency of the dictionary atoms. A low reconstruction error and a high sparsity level can be achieved if the designed dictionary satisfies the above properties.

In the classical SRC, the dictionary is constructed by stacking all the training samples. The sparse recovery can be computationally burdensome when the training set is large. Besides, the dictionary constructed in this manner can neither be optimal for reconstruction purposes nor for classification of signals. Previous literature have shown that a dictionary can be trained to have a better representation of the dataset. Unsupervised dictionary learning methods, such as the method of optimal direction (MOD) [56], K-SVD [57] and online dictionary learning [58], are able to improve the signal restoration performance of numerous applications, such as compressive sensing, signal denoising and image inpainting.

However, the unsupervised dictionary learning method is not suitable for solving classification problems since a lower reconstruction error does not necessarily lead to a better

classification performance. In fact, it is observed that the dictionary can have an improved classification result by sacrificing some signal reconstruction performance [59]. Therefore, supervised dictionary learning methods [60] are proposed to improve the classification result. Unlike the unsupervised dictionary learning, which only trains the dictionary by pursuing a lower signal reconstruction error, the supervised learning is able to directly improve the classification performance by optimizing both the dictionary and classifier's parameter simultaneously. The discriminative dictionary learning in [23] minimizes the classification error of SRC by minimizing the reconstruction error contributed by the atoms from the correct class and maximizing the error from the remaining classes. The incoherent dictionary learning in [24] uses SRC as the classifier and tries to eliminate the atoms shared by pixels from different classes. It increases the discriminability of the sparse codes by decreasing the coherency of the atoms from different classes. The label consistent K-SVD (LC-KSVD) [17] optimizes the dictionary and classifier's parameter by minimizing the summation of reconstruction and classification errors. It combines the dictionary and classifier's parameter into a single parameter space, which makes it possible for the optimization procedure to be much simpler than those used in classical SRC. However, a desired and accurate solution is not guaranteed [61] because the cost function can be minimized by decreasing the reconstruction error while the classification error is increased. A bilevel optimization formulation would be more appropriate [62], where the update of the dictionary is driven by the minimization of the classification error. The task-driven dictionary learning (TDDL) [59] exploits this idea with theoretical proof and demonstrates a superior performance. The

supervised translation-invariant sparse coding, which uses the same scheme as TDDL, is developed independently by [63]. It is a more general framework that can be applied not only to classification, but also nonlinear image mapping, digital art authentiïňĄcation and compressive sensing. More recently, the group sparsity prior is enforced on a single measurement and the corresponding TDDL optimization algorithm is developed in [64] in order to improve the performance of region tagging.

In this chapter, we propose a novel method that enforces the joint or Laplacian sparsity prior on the sparse recovery stage of TDDL. The existing dictionary learning methods have only been developed for reconstructing or classifying a single measurement. Therefore, it is advantageous to incorporate structured sparsity priors into the supervised dictionary learning in order to achieve a better performance. This chapter makes the following contributions:

- We propose a new dictionary learning algorithm for TDDL with joint or Laplacian sparsity in order to exploit the spatial-spectral information of HSI neighboring pixels.
- We show experimentally that the proposed dictionary learning methods have a significantly better performance than SRC even when the dictionary is highly compact.
- We also describe an optimization algorithm for solving the Laplacian sparsity recovery problem. The proposed optimization method is much faster than the modified feature sign search used in [42].

The remainder of the chapter is organized as follows. In Section 3.2, a brief review

of TDDL is given. In Section 3.3, we propose a modified TDDL algorithm with the joint sparsity prior. TDDL with the Laplacian prior and a new algorithm for recovering the Laplacian sparse problem are stated in Section 3.4. In Section 3.5, we show that our method is superior to other HSI classification methods through experimental results on several HSI images. Finally, we provide our summary in Section 3.6.

3.2 Task-driven Dictionary Learning

In TDDL [59], signals are represented by their sparse codes, which are then fed into a linear regression or logistic regression. Consider a pair of training samples $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} \in \mathbb{R}^M$ is the HSI pixel, M is the number of spectral bands, and $\mathbf{y} \in \mathbb{R}^K$ is a binary vector representation of the label of the sample $\mathbf{x}$. K is the maximum class index. Pixel $\mathbf{x}$ can be represented by a sparse coefficient vector $\boldsymbol{\alpha}(\mathbf{D}, \mathbf{x}) \in \mathbb{R}^N$ with respect to some dictionary $\mathbf{D} \in \mathbb{R}^{M \times N}$ consisting of N atoms by solving the optimization

$$\boldsymbol{\alpha}(\mathbf{D}, \mathbf{x}) = \arg\min_{\mathbf{z}} \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda\|\mathbf{z}\|_1 + \frac{\epsilon}{2}\|\mathbf{z}\|_2^2, \tag{3.1}$$

,where λ and ϵ are the regularization parameters. λ controls the sparsity level of the coefficients $\boldsymbol{\alpha}$. In our experiments, we set ϵ to 0 since it does not affect the convergence of the algorithm and always gives satisfactory results.

To optimize the dictionary, TDDL first defines a convex function $\mathcal{L}(\mathbf{D}, \mathbf{W}, \{\mathbf{x}_i\}_{i=1}^S)$ to describe the classification risk in terms of the dictionary atoms, sparse coefficients and the

classifier's parameter $\mathbf{W}$. The function is then minimized as follows

$$\min_{\mathbf{D},\mathbf{W}} \mathcal{L}(\mathbf{D},\mathbf{W},\{\mathbf{x}_i\}_{i=1}^S) = \min_{\mathbf{D},\mathbf{W}} f(\mathbf{D},\mathbf{W},\{\mathbf{x}_i\}_{i=1}^S) + \frac{\mu}{2}\|\mathbf{W}\|_F^2, \tag{3.2}$$

where $\mu > 0$ is a classifier regularization parameter to avoid overfitting of the classifier. The convex function f is defined as

$$f(\mathbf{D},\mathbf{W},\{\mathbf{x}_i\}_{i=1}^S) \triangleq \frac{1}{S}\sum_{i=1}^{S} \mathcal{J}(\mathbf{y}_i,\mathbf{W},\boldsymbol{\alpha}_i(\mathbf{D},\mathbf{x}_i)), \tag{3.3}$$

where S is the total number of training samples and $\mathcal{L}(\mathbf{y}_i,\mathbf{W},\boldsymbol{\alpha}_i(\mathbf{D},\mathbf{x}_i))$ is the classification error for a training pair $(\mathbf{x}_i,\mathbf{y}_i)$ which is measured by a linear regression, i.e. $\mathcal{J}(\mathbf{y}_i,\mathbf{W},\boldsymbol{\alpha}_i(\mathbf{D},\mathbf{x}_i)) = \frac{1}{2}\|\mathbf{y}_i - \mathbf{W}\boldsymbol{\alpha}_i\|_2^2$.

In the following part of the section, we omit the subscript i of $\boldsymbol{\alpha}$ for notational simplicity. The dictionary $\mathbf{D}$ and the classifier parameter $\mathbf{W}$ are updated using a stochastic gradient descent algorithm, which has been independently investigated by [59, 63]. The update rules for $\mathbf{D}$ and $\mathbf{W}$ are

$$\begin{cases} \mathbf{D}^{(t+1)} = \mathbf{D}^{(t)} - \rho^{(t)} \cdot \partial\mathcal{L}^{(t)}/\partial\mathbf{D}, \\ \mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \rho^{(t)} \cdot \partial\mathcal{L}^{(t)}/\partial\mathbf{W}, \end{cases} \tag{3.4}$$

where t is the iteration index and ρ is the step size. The equations for updating the classifier parameter $\mathbf{W}$ is straightforward since $\mathcal{L}(\mathbf{y},\mathbf{W},\boldsymbol{\alpha}(\mathbf{D},\mathbf{x}))$ is both smooth and convex with

respect to $\mathbf{W}$. We have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = (\mathbf{W}\boldsymbol{\alpha} - \mathbf{y})\,\boldsymbol{\alpha}^\top + \mu\mathbf{W}. \tag{3.5}$$

The updating equation for the dictionary can be obtained by applying error backpropagation, where the chain rule is applied

$$\frac{\partial \mathcal{L}}{\partial \mathbf{D}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}} \frac{\partial \boldsymbol{\alpha}}{\partial \mathbf{D}}. \tag{3.6}$$

The difficulty of acquiring a specific form of the above equation comes from $\partial\boldsymbol{\alpha}/\partial\mathbf{D}$. Since the sparse coefficient $\boldsymbol{\alpha}(\mathbf{D}, \mathbf{x})$ is an implicit function of $\mathbf{D}$, an analytic form of $\boldsymbol{\alpha}$ with respect to $\mathbf{D}$ is not available. Fortunately, the derivative $\partial\boldsymbol{\alpha}/\partial\mathbf{D}$ can still be computed by either applying optimality condition of elastic net [59, 65] or using fixed point differentiation [63, 66].

We now focus on computing the derivative using the fixed point differentiation. As suggested in [66], the gradient of Eq. (3.1) reaches $\mathbf{0}$ at the optimal point $\hat{\boldsymbol{\alpha}}$

$$\frac{\partial \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2}{\partial \boldsymbol{\alpha}}\Big|_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}} = -\lambda \frac{\partial \|\boldsymbol{\alpha}\|_1}{\partial \boldsymbol{\alpha}}\Big|_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}}. \tag{3.7}$$

Expanding Eq. (3.7), we have

$$2\mathbf{D}^\top(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})\Big|_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}} = \lambda \cdot sign(\boldsymbol{\alpha})\Big|_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}}. \tag{3.8}$$

In order to evaluate $\partial\boldsymbol{\alpha}/\partial\mathbf{D}$, the derivative of Eq. (3.8) with respect to each element

D_{mn} of the dictionary is required. Since the differentiation of the $sign$ function is not well defined at zero points, we can only compute the derivative of Eq. (3.8) at fixed points when $\boldsymbol{\alpha}_{[n]} \neq 0$ [63]

$$\frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial D_{mn}} = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1} \left(\frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial D_{mn}} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial D_{mn}} \boldsymbol{\alpha}_\Lambda \right) \text{ and } \frac{\partial \boldsymbol{\alpha}_{\Lambda^C}}{\partial D_{mn}} = \mathbf{0}, \tag{3.9}$$

where Λ and Λ^c are the indices of the active and inactive set of $\boldsymbol{\alpha}$ respectively. $D_{mn} \in \mathbb{R}$ is the (m, n) element of $\mathbf{D}$. $(\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1}$ is always invertible since the number of active atoms $|\Lambda|$ is always much smaller than the feature dimension M.

3.3 Task-driven Dictionary Learning with Joint Sparsity Prior

We now extend TDDL by using a joint sparsity (JS) prior (TDDL-JS). The joint sparsity prior [48, 49] enforces the sparse coefficients of the test pixel and its neighboring pixels within the neighborhood window to have row sparsity pattern, where all pixels are represented by the same atoms in the dictionary so that only few rows of the sparse coefficients matrix are nonzero. The joint sparse recovery can be solved by the following Lasso

problem

$$\mathbf{A} = \arg\min_{\mathbf{Z}} \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 + \lambda\|\mathbf{Z}\|_{1,2}, \tag{3.10}$$

where $\mathbf{A}, \mathbf{Z} \in \mathbb{R}^{N\times P}$ are sparse coefficient matrices and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_P] \in \mathbb{R}^{M\times P}$ represents all the pixels within a neighborhood window centered on a test (center) pixel $\mathbf{x}_c$. Define the label of the center pixel as $\mathbf{y}_c$. P is the total number of pixels within the neighborhood window. $\|\mathbf{Z}\|_{1,2} = \sum_{i=1}^{P} \|\mathbf{Z}_i\|_2$ is the $\ell_{1,2}$-norm of $\mathbf{Z}$. $\mathbf{Z}_i \in \mathbb{R}^{1\times P}$ is the i^{th} row of $\mathbf{Z}$. Many sparse recovery techniques are able to solve Eq. (3.10), such as the Alternating Direction Method of Multipliers [67], Sparse Reconstruction by Separable Approximation (SpaRSA) [52] and Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [68].

Once the sparse code $\mathbf{A}$ is obtained, the sparse codes $\boldsymbol{\alpha}_c$ of the center pixel $\mathbf{x}_c$ is projected on each of the K decision planes of the classifier. The plane with the largest projection indicates the class that the center pixel $\mathbf{x}_c$ belongs to,

$$\text{identity}(\mathbf{x}_c) = \arg\max_k \hat{\mathbf{y}}_k = \arg\max_k (\mathbf{W}\boldsymbol{\alpha}_c)_k, \tag{3.11}$$

where $\boldsymbol{\alpha}_c \in \mathbb{R}^N$ is the sparse coefficients of the center pixel. In the training stage, it is expected that the projection of the decision plane corresponding to the class of the center pixel should be increased while other planes should be orthogonal to $\boldsymbol{\alpha}_c$. Therefore, given

the training data $(\mathbf{X}, \mathbf{y}_c)$, the classification error for the center pixel $\mathbf{x}_c$ is defined as

$$\mathcal{L}(\mathbf{y}_c, \mathbf{W}, \boldsymbol{\alpha}_c(\mathbf{D}, \mathbf{X})) = \|\mathbf{y}_c - \mathbf{W}\boldsymbol{\alpha}_c\|_2^2 + \frac{\mu}{2}\|\mathbf{W}\|_F^2, \tag{3.12}$$

In order to update the dictionary $\mathbf{D}$, we need to apply a chain rule similar to the one in Eq. (3.6):

$$\frac{\partial \mathcal{L}}{\partial \mathbf{D}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}} \frac{\partial \mathbf{A}}{\partial \mathbf{D}}. \tag{3.13}$$

Now we focus on the difficult part $\frac{\partial \mathbf{A}}{\partial \mathbf{D}}$ of Eq. (3.13). Employing the fixed point differentiation on Eq. (3.10), we have

$$\frac{\partial \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2}{\partial \mathbf{A}}\Big|_{\mathbf{A}=\hat{\mathbf{A}}} = -\lambda \frac{\partial \|\mathbf{A}\|_{1,2}}{\partial \mathbf{A}}\Big|_{\mathbf{A}=\hat{\mathbf{A}}}. \tag{3.14}$$

In the following part of this section, we omit the fixed point notation. Eq. (3.14) is only differentiable when $\|\mathbf{A}_i\|_2 \neq \mathbf{0}$, where $\mathbf{A}_i$ denotes the i^{th} row of $\mathbf{A}$. At points where $\|\mathbf{A}_i\|_2 = \mathbf{0}$, the derivative is not well defined, so we set $\frac{\partial \|\mathbf{A}_i\|_2}{\partial \mathbf{A}_i} = \mathbf{0}$. Denote $\tilde{\mathbf{A}} = \mathbf{A}_\Lambda \in \mathbb{R}^{N_\Lambda \times P}$, where Λ is the active set such that $\Lambda = \{i : \|\mathbf{A}_i\|_2 \neq 0, i \in \{1, \ldots, N\}\}$, $N_\Lambda = |\Lambda|$, $\mathbf{A}_\Lambda$ is composed of active rows of $\mathbf{A}$, and $\tilde{\mathbf{D}}$ is the active atoms of $\mathbf{D}$. Expanding the derivative of Eq. (3.14) on both sides on the feasible points,

$$\tilde{\mathbf{D}}^\top \left(\mathbf{X} - \tilde{\mathbf{D}}\tilde{\mathbf{A}}\right) = \lambda \left[\frac{\tilde{\mathbf{A}}_1^\top}{\|\tilde{\mathbf{A}}_1\|_2}, \ldots, \frac{\tilde{\mathbf{A}}_{N_\Lambda}^\top}{\|\tilde{\mathbf{A}}_{N_\Lambda}\|_2}\right]^\top. \tag{3.15}$$

Algorithm 1 Stochastic gradient descent algorithm for task-driven dictionary learning with joint sparsity prior

Require: Initial dictionary $\mathbf{D}$ and classifier $\mathbf{W}$. Parameter λ, ρ and t_0.

1: **for** $t = 1$ to T **do**

2: Draw one sample $(\mathbf{X}, \mathbf{y}_c)$ from training set.

3: Find sparse sparse code $\mathbf{A}$ according to Eq. (3.10).

4: Find the active set Λ and define $N_\Lambda = |\Lambda|$

$$\Lambda \leftarrow \{i : \|\mathbf{A}_i\|_2 \neq 0, i \in \{1, \ldots, N\}\},$$

where $\mathbf{A}_i$ is the i^{th} row of $\mathbf{A}$.

5: Compute $\boldsymbol{\Gamma} \in \mathbb{R}^{N_\Lambda P \times N_\Lambda P}$

$$\boldsymbol{\Gamma} = \boldsymbol{\Gamma}_1 \oplus \cdots \oplus \boldsymbol{\Gamma}_{N_\Lambda},$$
$$\boldsymbol{\Gamma}_i = \frac{\mathbf{I}_P}{\|\tilde{\mathbf{A}}_i\|_2} - \frac{\tilde{\mathbf{A}}_i \tilde{\mathbf{A}}_i^\top}{\|\tilde{\mathbf{A}}_i\|_2^3}, i = 1, \ldots, N_\Lambda,$$

where $\oplus$ is the direct sum of matrices.

6: Compute $\boldsymbol{\gamma} \in \mathbb{R}^{N_\Lambda P}$

$$\boldsymbol{\gamma} = (\tilde{\mathbf{D}}^\top \tilde{\mathbf{D}} \otimes \mathbf{I}_P + \lambda \boldsymbol{\Gamma})^{-\top} vec((\mathbf{W}\hat{\mathbf{A}} - \hat{\mathbf{Y}})^\top \tilde{\mathbf{W}}).$$

where $vec(\cdot)$ and $\tilde{\mathbf{W}}$ denote the vectorization operator and Λ columns of $\mathbf{W}$ respectively.

7: Let $\boldsymbol{\beta} \in \mathbb{R}^{N \times P}$. Set $\boldsymbol{\beta}_{\Lambda^C} = \mathbf{0}$ and construct $\boldsymbol{\beta}_\Lambda \in \mathbb{R}^{N_\Lambda \times P}$ that satisfies

$$vec\left(\boldsymbol{\beta}_\Lambda^\top\right) = \boldsymbol{\gamma}.$$

8: Choose the learning rate $\rho_t \leftarrow \min(\rho, \rho \frac{t_0}{t})$.

9: Update the parameters by gradient projection step

$$\mathbf{W} \leftarrow \mathbf{W} - \rho_t\big((\mathbf{W}\boldsymbol{\alpha}_c - \mathbf{y})\boldsymbol{\alpha}_c^\top + \mu\mathbf{W}\big),$$
$$\mathbf{D} \leftarrow \mathbf{D} - \rho_t(-\mathbf{D}\boldsymbol{\beta}\mathbf{A}^\top + (\mathbf{X} - \mathbf{D}\mathbf{A})\boldsymbol{\beta}^\top),$$

and normalize every column of $\mathbf{D}^{(t+1)}$ with respect to ℓ_2-norm.

10: **end for**

11: **return** $\mathbf{D}$ and $\mathbf{W}$.

Computing the derivative of Eq. (3.15) with respect to D_{mn} and transposing both sides

$$\frac{\partial \left\{ (\mathbf{X} - \mathbf{D}\mathbf{A})^\top \tilde{\mathbf{D}} \right\}}{\partial D_{mn}} = \lambda \left[\boldsymbol{\Gamma}_1 \frac{\partial \tilde{\mathbf{A}}_1^\top}{\partial D_{mn}}, \ldots, \boldsymbol{\Gamma}_{N_\Lambda} \frac{\partial \tilde{\mathbf{A}}_{N_\Lambda}^\top}{\partial D_{mn}} \right], \tag{3.16}$$

where $\mathbf{\Gamma}_i = \frac{\mathbf{I}_P}{\|\tilde{\mathbf{A}}_i\|_2} - \frac{\tilde{\mathbf{A}}_i^\top \tilde{\mathbf{A}}_i}{\|\tilde{\mathbf{A}}_i\|_2^3}$, $i = 1, \ldots, N_\Lambda$. By vectorizing Eq. (3.16), we have

$$vec\left(\frac{\partial \mathbf{X}^\top \tilde{\mathbf{D}}}{\partial D_{mn}} - \tilde{\mathbf{A}}^\top \frac{\partial \tilde{\mathbf{D}}^\top \tilde{\mathbf{D}}}{\partial D_{mn}} - \frac{\partial \tilde{\mathbf{A}}^\top}{\partial D_{mn}} \tilde{\mathbf{D}}^\top \tilde{\mathbf{D}}\right) = \lambda \cdot \mathbf{\Gamma} vec\left(\frac{\partial \tilde{\mathbf{A}}^\top}{\partial D_{mn}}\right), \tag{3.17}$$

where $\mathbf{\Gamma} = \mathbf{\Gamma}_1 \oplus \cdots \oplus \mathbf{\Gamma}_{N_\Lambda}$. From Eq. (3.17), we reach the vectorization form of the derivative of $\tilde{\mathbf{A}}$ with respect to D_{mn}, given as

$$vec\left(\frac{\partial \tilde{\mathbf{A}}^\top}{\partial D_{mn}}\right) = \left(\tilde{\mathbf{D}}^\top \tilde{\mathbf{D}} \otimes \mathbf{I}_P + \lambda \mathbf{\Gamma}\right)^{-1} vec\left(\tilde{\mathbf{A}}^\top \frac{\partial \tilde{\mathbf{D}}^\top \tilde{\mathbf{D}}}{\partial D_{mn}} + \frac{\partial \mathbf{X}^\top \tilde{\mathbf{D}}}{\partial D_{mn}}\right). \tag{3.18}$$

Now we can update the dictionary element-wise using Eq. (3.18). In order to reach a more concise form for updating the dictionary, we perform algebraic transformations on Eq. (3.13) and Eq. (3.18), which are illustrated in Appendix 8.1. We illustrate the overall optimization for TDDL-JS in Algorithm 1. It should be noted that in the Algorithm 1, we define $\hat{\mathbf{A}} = [\mathbf{0}, \ldots, \boldsymbol{\alpha}_c, \ldots, \mathbf{0}] \in \mathbb{R}^{N \times P}$ and $\hat{\mathbf{Y}} = [\mathbf{0}, \ldots, \mathbf{y}, \ldots, \mathbf{0}] \in \mathbb{R}^{K \times P}$.

3.4 Task-driven Dictionary Learning with Laplacian Sparsity Prior

The joint sparsity prior is a relatively stringent constraint on the sparse codes since it assumes that all the neighboring pixels have the same support as the center pixel. The

assumption of the joint sparsity prior can easily be violated on non-homogeneous regions, such as a region that contains pixels from different classes. This makes choosing a proper neighborhood window size a difficult problem. When the window size is too large, the sparse codes of the non-homogeneous regions within the window are indiscriminative. On the other hand, the sparse codes are not stable if the window size is chosen to be too small. Ideally, we hope that the performance is insensitive to both the choice of the window size and the topology of the image. To achieve this requirement, we propose to enforce the Laplacian sparsity (LP) prior (TDDL-LP) on the TDDL, where the degree of similarity between neighboring pixels can be utilized to push the sparse codes of the neighboring pixels that belong to the same class to be similar, instead of enforcing all the neighboring pixels to have a similar sparse codes blindly. The corresponding Lasso problem can be stated as follows

$$\mathbf{A} = \arg\min_{\mathbf{Z}} \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_2^2 + \lambda\|\mathbf{Z}\|_1 + \gamma \sum_{i,j}^{P} c_{ij}\|\mathbf{Z}^i - \mathbf{Z}^j\|_2^2, \tag{3.19}$$

where $\mathbf{Z}^i$ and $\mathbf{Z}^j$ denote the i^{th} and j^{th} columns of $\mathbf{Z}$. c_{ij} is a weight whose value is proportional to the spectral similarity of $\mathbf{X}^i$ and $\mathbf{X}^j$, which are the i^{th} and j^{th} columns of $\mathbf{X}$. γ is a regularization parameter.

The Laplacian sparse recovery described by Eq. (3.19) in [42] is able to discriminate pixels from different classes by defining an appropriate weighting matrix $\mathbf{C} = [c_{ij}] \in \mathbb{R}^{P\times P}$. Additionally, it enforces both the support and the magnitude of sparse coefficients of similar spectral pixels to be similar, whereas the joint sparsity prior enforces sparse

coefficients of all the pixels within the neighborhood window to have the same support. Eq. (3.19) can be reformulated as

$$\mathbf{A} = \arg\min_{\mathbf{Z}} \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_2^2 + \lambda\|\mathbf{Z}\|_1 + \gamma tr(\mathbf{Z}\mathbf{L}\mathbf{Z}^\top), \tag{3.20}$$

where $\mathbf{L} = \mathbf{B} - \mathbf{C} \in \mathbb{R}^{P\times P}$ is the Laplacian matrix [69]. $\mathbf{B} = [b_{ij}] \in \mathbb{R}^{P\times P}$ is a diagonal matrix such that $b_{ii} = \sum_j c_{ij}$.

In this chapter, we adopt the method of Sparse Reconstruction by Separable Approximation (SpaRSA) [25, 52] to solve the Laplacian sparse coding problem.

3.4.1 Sparse Recovery Algorithm

A modified feature sign search [42] is capable of solving the optimization problem (3.20). It uses coordinate descent to update each column of $\mathbf{A}$ iteratively. Although it gives plausible performance for the SRC-based HSI classification [25], it demands a high computational cost. The SpaRSA-based method can achieve a similar optimal solution of Eq. (3.20) while being less computational burdensome. Despite the fact that our previous work [25] has shown that the performance of the SRC-based approach for HSI classification can be largely influenced by the choice of specific optimization technique, we found that such influence is reasonably small when employing the dictionary-learning-based approach. Therefore, we use a SpaRSA-based method to solve the sparse recovery for the Laplacian sparsity prior. Although, SpaRSA is originally designed to solve the optimiza-

tion of single-signal case, it can be easily extended to tackle the problem with multiple signals, such as the collaborative hierarchical Lasso (C-Hilasso) [46].

SpaRSA is able to solve optimization problems that have the following form

$$\min_{\mathbf{A}\in\mathbb{R}^{N\times P}} f\left(\mathbf{A}\right)+\lambda\psi\left(\mathbf{A}\right), \tag{3.21}$$

where $f:\mathbb{R}^{N\times P}\rightarrow\mathbb{R}$ is a convex and smooth function, $\psi:\mathbb{R}^{N\times P}\rightarrow\mathbb{R}$ is a separable regularizer and λ is the regularization parameter. In the particular case of the Laplacian sparse recovery, the regularizer ψ is chosen to be the ℓ_1-norm, i.e. $\psi(\mathbf{A})=\|\mathbf{A}\|_1$, and the convex function f is set as

$$f\left(\mathbf{A}\right)=\|\mathbf{X}-\mathbf{D}\mathbf{A}\|_F^2+\gamma tr\left(\mathbf{A}\mathbf{L}\mathbf{A}^\top\right). \tag{3.22}$$

In order to search the optimal solution of Eq. (3.21), SpaRSA generates a sequence of iterations $\mathbf{A}^{(t)}$, $t=1,2,\ldots,$ by solving the following subproblem

$$\mathbf{A}^{(t+1)}\in\arg\min_{\mathbf{Z}\in\mathbb{R}^{N\times P}}\left(\mathbf{Z}-\mathbf{A}^{(t)}\right)^\top\nabla f(\mathbf{A}^{(t)})+\frac{\eta^{(t)}}{2}\|\mathbf{Z}-\mathbf{A}^{(t)}\|_F^2+\gamma\psi\left(\mathbf{Z}\right), \tag{3.23}$$

where $\eta^{(t)}>0$ is a nonnegative scalar such that $\eta^{(t)}=\mu\eta^{(t-1)}$ and $\mu>1$. The Eq. (3.23) can be simplified into the following form by eliminating the terms independent of $\mathbf{Z}$

$$\min_{\mathbf{Z}\in\mathbb{R}^{N\times P}}\frac{1}{2}\|\mathbf{Z}-\mathbf{U}^{(t)}\|_F^2+\frac{\gamma}{\eta^{(t)}}\psi(\mathbf{Z}), \tag{3.24}$$

where $\mathbf{U}^{(t)}=\mathbf{A}^{(t)}-\frac{1}{\eta^{(t)}}\nabla f(\mathbf{A}^{(t)})$. The optimization problem in Eq. (3.24) is separable

Algorithm 2 Sparse recovery for Laplacian sparsity prior using SpaRSA

Require: Dictionary $\mathbf{D}$, constants $\eta_0 > 0$, $0 < \eta_{\min} < \eta_{\max}$, $\mu > 1$
1: Set $t = 0$ and $\mathbf{A}^{(0)} = \mathbf{0}$
2: **repeat**
3: choose $\eta^{(t)} \in [\eta_{\min}, \eta_{\max}]$
4: compute $\mathbf{U}^{(t)} \leftarrow \mathbf{A}^{(t)} - \frac{1}{\eta^{(t)}} \nabla f(\mathbf{A}^{(t)})$.
5: **repeat**
6: $\mathbf{A}^{(t)} \leftarrow S_{\frac{\gamma}{\eta^{(t)}}} \left(\mathbf{U}^{(t)} \right)$,
7: $\eta^{(t)} \leftarrow \mu \eta^{(t)}$.
8: **until** stopping criterion is satisfied
9: $t \leftarrow t + 1$.
10: **until** stopping criterion is satisfied
11: **return** The optimal sparse coefficients $\mathbf{A}^*$.

element-wise, which can be reformulated into

$$\min_{A_{ij}} \frac{1}{2}(z_{ij} - u_{ij}^{(t)})^2 + \frac{\lambda}{\eta^{(t)}} \psi_{ij}(\mathbf{Z}), \forall i = 1, \ldots, N \text{ and } j = 1, \ldots, P. \tag{3.25}$$

The problem in Eq. (3.25) has a unique solution and can be solved by the well-known soft thresholding operator $S(\cdot)$

$$z_{ij}^* = S_{\frac{\gamma}{\eta^{(t)}}} \left(u_{ij}^{(t)} \right) = sign(u_{ij}^{(t)}) \max\{0, |u_{ij}| - \frac{\lambda}{\eta^{(t)}}\}. \tag{3.26}$$

Comparing with the algorithm proposed in [42], which is based on the coordinate descent, Laplacian sparse recovery using SpaRSA is more computationally efficient since it is able to cheaply search for a better descent direction $\nabla f(\mathbf{A})$. The corresponding optimization is stated in Algorithm 2.

3.4.2 Dictionary Updating Rule

In order to adjust the dictionary, we now follow Eq. (3.13) to derive $\frac{\partial \mathbf{A}}{\partial \mathbf{D}}$ using the fixed point differentiation. Applying differentiation on Eq. (3.19) on the fixed point $\hat{\mathbf{A}}$

$$\frac{\partial \|\mathbf{X}-\mathbf{D}\mathbf{A}\|_F^2+\gamma tr\left(\mathbf{A}\mathbf{L}\mathbf{A}^\top\right)}{\partial \mathbf{A}}\Big|_{\mathbf{A}=\hat{\mathbf{A}}}=-\lambda\frac{\partial \|\mathbf{A}\|_1}{\partial \mathbf{A}}\Big|_{\mathbf{A}=\hat{\mathbf{A}}}. \tag{3.27}$$

In the following part, we omit the fixed point notation. By computing the derivation and then applying the vectorization on Eq. (3.27), we have

$$vec\left(\mathbf{D}^\top(\mathbf{X}-\mathbf{D}\mathbf{A})-\gamma\mathbf{A}\mathbf{L}\right)=\lambda\cdot vec\left(sign\left(\mathbf{A}\right)\right). \tag{3.28}$$

The differentiation $\frac{\partial vec(sign(\mathbf{A}))}{\partial D_{mn}}$ is not well defined on zero points of $vec\left(sign\left(\mathbf{A}\right)\right)$. Similar as in TDDL-JS, we set the i^{th} element $\frac{\partial vec(sign(\mathbf{A}))_i}{\partial D_{mn}}=0$ when $vec\left(sign\left(\mathbf{A}\right)\right)_i=0$. Denote the Λ as the index set of nonzero elements of $vec\left(sign\left(\mathbf{A}\right)\right)$. Compute the derivative of Eq. (3.28) with respect to D_{mn}

$$\frac{\partial\left\{vec\left(\mathbf{D}^\top(\mathbf{X}-\mathbf{D}\mathbf{A})-\gamma\mathbf{A}\mathbf{L}\right)_\Lambda\right\}}{\partial D_{mn}}=\mathbf{0}, \tag{3.29}$$

which leads to

$$vec\left(\frac{\partial \mathbf{D}^\top\mathbf{D}}{\partial D_{mn}}\mathbf{A}-\frac{\partial \mathbf{D}^\top\mathbf{X}}{\partial D_{mn}}+\mathbf{D}^\top\mathbf{D}\frac{\partial \mathbf{A}}{\partial D_{mn}}+\gamma\frac{\partial \mathbf{A}}{\partial D_{mn}}\mathbf{L}\right)_\Lambda=\mathbf{0}. \tag{3.30}$$

Algorithm 3 Stochastic gradient descent algorithm for task-driven dictionary learning with Laplacian sparsity prior

Require: Initial dictionary $\mathbf{D}$ and classifier $\mathbf{W}$. Parameter λ, ρ and t_0.

1: **for** $t = 1$ to T **do**

2: Draw one sample $(\mathbf{X}, \mathbf{y}_c)$ from training set.

3: Find sparse code $\mathbf{A}$ according to Eq. (3.10).

4: Find the active set Λ

$$\Lambda \leftarrow \{i : vec(\mathbf{A})_i \neq 0, i \in \{1, \ldots, NP\}\},$$

where $vec(\mathbf{A})_i$ is the i^{th} element of $vec(\mathbf{A})$.

5: Let $\boldsymbol{\beta} \in \mathbb{R}^{N \times P}$. Set $vec(\boldsymbol{\beta})_{\Lambda^c} = \mathbf{0}$ and compute $vec(\boldsymbol{\beta})_\Lambda$

$$vec(\boldsymbol{\beta})_\Lambda = (\mathbf{I}_P \otimes \mathbf{D}^\top\mathbf{D} + \gamma\mathbf{L} \otimes \mathbf{I}_N)^{-1}_{\Lambda,\Lambda} vec(\mathbf{W}^\top(\mathbf{W}\hat{\mathbf{A}} - \hat{\mathbf{Y}}))_\Lambda,$$

and $\otimes$ denotes the Kronecker product.

6: Choose the learning rate $\rho_t \leftarrow \min(\rho, \rho\frac{t_0}{t})$.

7: Update the parameters by gradient projection step

$$\mathbf{W} \leftarrow \mathbf{W} - \rho_t\big((\mathbf{W}\boldsymbol{\alpha}_c - \mathbf{y})\boldsymbol{\alpha}_c^\top + \mu\mathbf{W}\big),$$
$$\mathbf{D} \leftarrow \mathbf{D} - \rho_t(-\mathbf{D}\boldsymbol{\beta}\mathbf{A}^\top + (\mathbf{X} - \mathbf{D}\mathbf{A})\boldsymbol{\beta}^\top),$$

and normalize every column of $\mathbf{D}^{(t+1)}$ with respect to ℓ_2-norm.

8: **end for**

9: **return** $\mathbf{D}$ and $\mathbf{W}$.

Now we reach the desired gradient

$$vec\left(\frac{\partial \mathbf{A}}{\partial D_{mn}}\right)_\Lambda = \left(\mathbf{I}_P \otimes \mathbf{D}^\top\mathbf{D} + \gamma\mathbf{L} \otimes \mathbf{I}_N\right)^{-1}_{\Lambda,\Lambda} vec\left(\frac{\partial \tilde{\mathbf{D}}^\top}{\partial D_{mn}}\tilde{\mathbf{D}}\tilde{\mathbf{A}} + \frac{\partial \tilde{\mathbf{D}}^\top}{\partial D_{mn}}\mathbf{X}\right)_\Lambda. \tag{3.31}$$

By applying algebraic simplification to Eq. (3.31), which is shown in Appendix 8.1, we reach the optimzation for TDDL-LP as stated in the Algorithm 3. It should be noted that $\hat{\mathbf{A}}$ and $\hat{\mathbf{Y}}$ have the same definitions as those in Algorithm 1.

3.5 Experimental Verification

3.5.1 Datasets and Dictionary Generation

Cross-validation to obtain the optimal values for all parameters, including $\lambda, \epsilon, \gamma$ (sparse coding regularization parameters), μ (regularization parameter for the classifier), ρ_0 (initial step size), N (dictionary size) and P (number of neighboring pixels), would introduce significant computational cost. Instead, we search for the optimal values for the above parameters according to the following procedure.

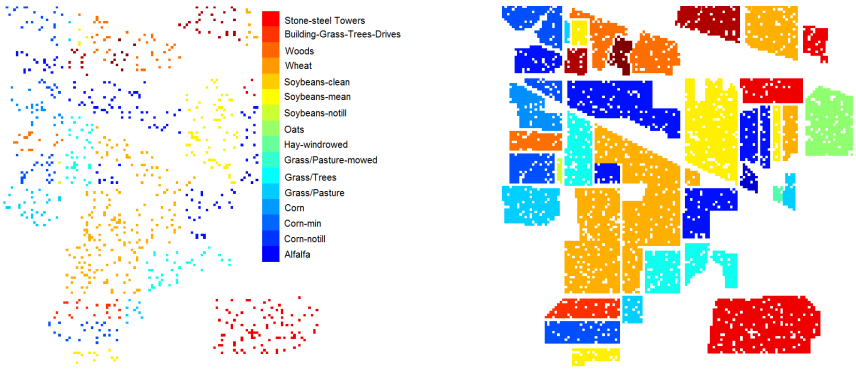
- The candidate dictionary sizes are from 5 to 10 atoms per class. The choice of dictionary size depends on the classification performance and computational cost. In our experiment, we set the dictionary size to be 5 atoms per class.

- Searching for the optimal window size and the regularization parameters would be cumbersome. Empirically, we found that the optimal regularization parameters are less likely to be affected by the choice of the window size. Therefore, for each image, we fix the window size to be 3×3 in order to save computational resource during the search of the optimal regularization parameters. Candidate regularization parameters are $\{10^{-3}, 10^{-2}, 10^{-1}\}$.

- The possible candidate window sizes are 3×3, 5×5, 7×7 and 9×9. We search for the optimal window size for each image after finding the optimal regularization parameters.

Table 3.1: Parameters used for dictionary learning on the Indian Pine image

Structured Priors	λ	γ	ρ
ℓ_1	10^{-2}	-	10^{-2}
JS	10^{-2}	-	10^{-3}
LP	10^{-2}	10^{-3}	10^{-1}



(a) (b)

Figure 3.1: (a) Training sets and (b) test sets of the Indian Pine image.

Computing the gradient for a single training sample at each iteration of Algorithm 1 or 3 will make the algorithm converge very slowly. Therefore, following the previous work [58, 59], we implement the two proposed algorithms with the mini-batch method, where the gradients of multiple training samples are computed in each iteration. For the unsupervised learning methods, the batch size is set to 200. For the supervised learning methods, the batch size is set to 100 and $t_0 = T/10$. We search the optimal regularization parameters for each image and found that their optimal values are coincidentally the same. The reason could be due to our choice of a large interval for the search grid. The regularization parameters used in our chapter are shown in Table 3.1. We set $\mu = 10^{-4}$. As a standard procedure, we evaluate the classification performance on HSI image using the overall accuracy (OA), average accuracy (AA) and kappa coefficient (κ). The classification methods

that are tested and compared are SVM, SRC, SRC with joint sparsity prior (SRC-JS), SRC with Laplacian sparsity prior (SRC-LP), unsupervised dictionary learning (ODL), unsupervised dictionary learning with joint or Laplacian sparsity prior (ODL-JS, ODL-LP), TDDL, TDDL-JS and TDDL-LP. During the testing stage, all training pixels are excluded from the HSI image, which means there may be some 'holes' (training pixels deleted) inside a neighborhood window. This is reasonable since we do not want the classification results to be affected by the spatial distribution of the labelled samples. We use SPAMS toolbox [70] to perform the joint sparse recovery via the Fast Iterative Shrinkage-Thresholding Algorithm [68]. The sparse recovery for SRC-based methods are performed via the Alternating Direction Method of Multipliers [67]. The modified SpaRSA shown in Algorithm 2 is used to solve the Laplacian sparse recovery problem.

For the unsupervised dictionary learning methods, the dictionary is initialized by randomly choosing a subset of the training pixels from each class and updated using the online dictionary learning (ODL) procedure in [58]. The classifier's parameter are then obtained by using a multi-class linear regression. For the supervised dictionary learning methods, the dictionary and classifier's parameter are initialized by the training results of ODL for the unsupervised method.

Table 3.2: Number of training and test samples for the Indian Pine image

Class #	Name	Train	Test
1	Alfalfa	6	48
2	Corn-notill	137	1297
3	Corn-min	80	754
4	Corn	23	211
5	Grass/Pasture	48	449
6	Grass/Trees	72	675
7	Grass/Pasture-mowed	3	23
8	Hay-windrowed	47	442
9	Oats	2	18
10	Soybeans-notill	93	875
11	Soybeans-min	235	2233
12	Soybean-clean	59	555
13	Wheat	21	191
14	Woods	124	1170
15	Building-Grass-Trees-Drives	37	343
16	Stone-steel Towers	10	85
Total		997	9369

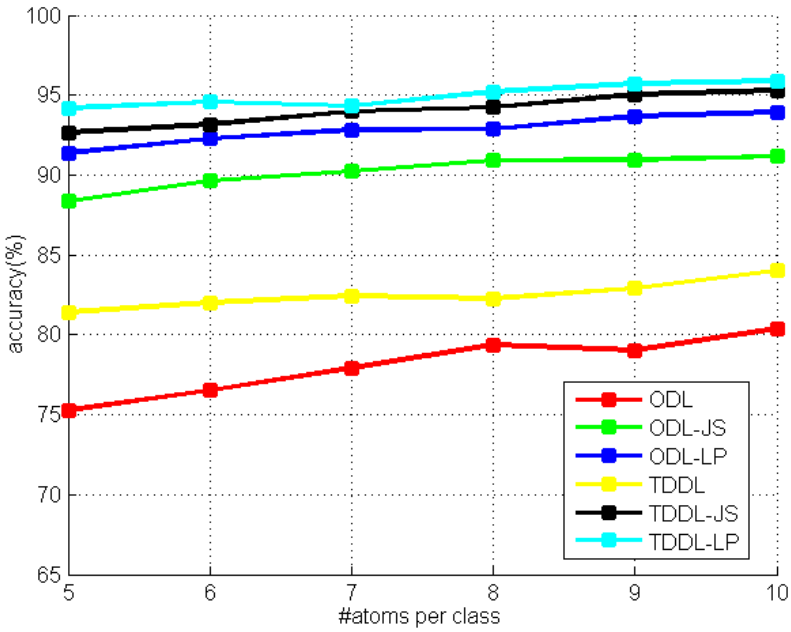


Figure 3.2: The result with different dictionary sizes for the Indian Pine image.

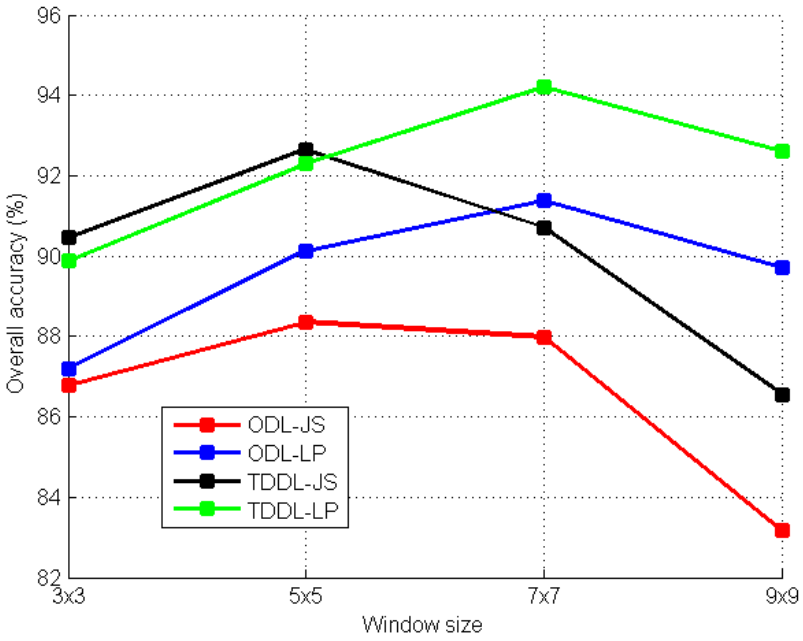


Figure 3.3: The effect of different window sizes for the Indian Pine image. The dictionary size is fixed at five atoms per class.

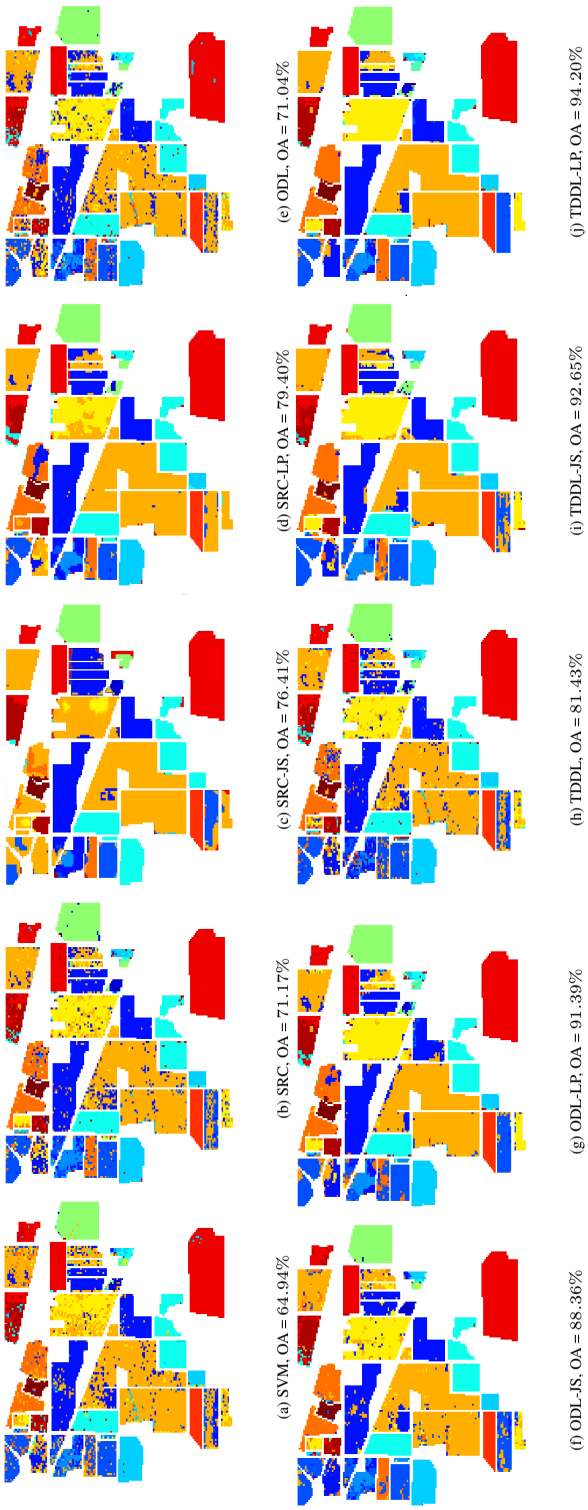


Figure 3.4: Classification map of the Indian Pine image obtained by (a) SVM, (b) SRC, (c) SRC-JS, (d) SRC-LP, (e) ODL, (f) ODL-JS, (g) ODL-LP, (h) TDDL, (i) TDDL-JS and (j) TDDL-LP.

Table 3.3: Classification accuracy (%) for the Indian Pine image

Dictionary Size		$N = 997$			$N = 80$					
Class	SVM	SRC	SRC-JS	SRC-LP	ODL	ODL-JS	ODL-LP	TDDL	TDDL-JS	TDDL-LP
1	77.08	68.75	79.17	82.42	75.00	**97.92**	70.83	50.00	35.42	56.25
2	84.96	58.84	81.94	81.34	59.69	91.24	94.26	84.03	**94.57**	93.95
3	62.67	24.40	56.67	47.35	62.93	81.20	84.40	69.73	84.13	**92.13**
4	8.57	49.52	27.62	49.76	23.81	47.62	61.90	14.76	**79.05**	46.19
5	77.18	81.88	85.46	83.96	82.55	**93.29**	92.62	89.04	90.16	90.83
6	91.82	96.88	98.36	97.48	88.24	99.55	98.96	98.66	**99.55**	98.96
7	13.04	0.00	0.00	0.00	4.35	**17.39**	0.00	0.00	0.00	**95.65**
8	96.59	96.59	**100.00**	99.55	96.36	99.32	99.32	99.09	100.00	100.00
9	0.00	**5.56**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	71.30	24.00	18.94	31.89	67.51	77.73	91.04	72.90	90.13	**94.03**
11	35.25	96.22	91.63	94.58	67.94	88.25	94.10	85.46	96.22	**97.37**
12	42.39	32.97	45.29	64.68	80.62	88.59	83.15	59.06	86.78	**95.47**
13	91.05	98.95	99.47	99.48	95.79	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
14	94.85	98.97	98.97	99.49	87.20	97.77	99.14	98.11	**99.40**	**99.40**
15	30.70	49.71	55.85	63.84	32.16	70.76	67.84	47.66	77.78	**82.75**
16	27.06	88.24	95.29	97.65	69.41	96.47	85.88	92.94	91.76	**98.82**
OA[%]	64.94	71.17	76.41	79.40	71.04	88.36	91.39	81.43	92.65	**94.20**
AA[%]	56.53	60.72	64.67	64.67	62.10	77.94	82.18	66.43	76.56	**83.86**
κ	0.647	0.695	0.737	0.712	0.691	0.851	0.907	0.8087	0.924	**0.940**

3.5.2 Performance on AVIRIS Indian Pine Dataset

We first perform HSI classification on the Indian Pine image, which is generated by Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). Every pixel of the Indian Pine consists of 220 bands ranging from 0.2 to 2.4μm, of which 20 water absorption bands are removed before classification. The spatial dimension of this image is 145×145. The image contains 16 ground-truth classes, most of which are crops, as shown in Table 3.2. We randomly choose 997 pixels (10.64% of all the interested pixels) as the training set and the rest of the interested pixels for testing.

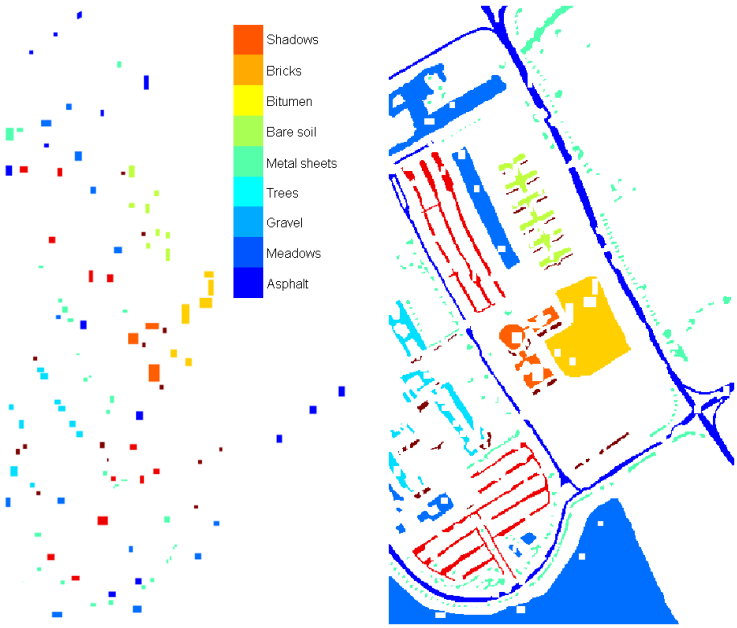
The total iterations of unsupervised and supervised dictionary learning methods are set to 15 and 200 respectively for this image. The classification results with varying dictionary size N are shown in Fig. 3.2. In most cases, the classification performance increases with the increment in the dictionary size. All methods attain their highest OA when the

Table 3.4: Number of training and test samples for the University of Pavia image

Class #	Name	Train	Test
1	Asphalt	548	6304
2	Meadows	540	18146
3	Gravel	392	1815
4	Trees	524	2912
5	Metal sheets	265	1113
6	Bare soil	532	4572
7	Bitumen	375	981
8	Bricks	514	3364
9	Shadows	231	795
Total		3921	40002

dictionary size is 10 atoms per class. The OA of ODL-JS, ODL-LP, TDDL-JS and TDDL-LP do not change much when the dictionary size increase from 5 to 10 atoms per class. Therefore, it is reasonable to set the dictionary size to be 5 atoms per class by taking computational cost into account. Fig. 3.2 also suggests that a plausible performance can be obtained even when the dictionary is very small and not over-complete. The classification performance with respect to the window size is demonstrated in Fig. 3.3. Using a window size of 5×5, ODL-JS and TDDL-JS achieves the highest OA of 88.36% and 92.65%, respectively. When the window size is set to 7×7, the ODL-LP and TDDL-LP reach their highest OA $= 91.39\%$ and OA $= 94.20\%$, respectively. ODL-JS and TDDL-JS reach better performance when the window size is not larger than 5×5. The TDDL-LP outperforms all other methods when the window size is 7×7 or larger. Since a larger window size has more chances to include non-homogeneous regions, it verifies our argument that the Laplacian sparsity prior works better for classifying pixels lying in the non-homogeneous regions.

Detailed classification results of various methods are shown in Table 3.3 and visually



(a) (b)

Figure 3.5: (a) Training sets and (b) test sets of the University of Pavia image.

displayed in Fig. 3.4. The OA of ODL-LP reaches 91.39%, which is more than 20% higher than that of ODL and 3% higher than that of ODL-JS. The TDDL-LP has the highest classification accuracy for most classes. Most methods have 0% accuracy for class 9 since there are too few training samples in this class. The overall performance of TDDL-JS and TDDL-LP have at least 13% improvement over the other conventional dictionary learning techniques. TDDL-LP significantly outperforms other methods on the classes that occupy small regions in the image. The class 7 (Grass/Pasture-mowed), lying in a non-homogeneous region, has only 3 training samples and 23 test samples. The TDDL-LP is capable of correctly classify 95.65% test samples while the second highest accuracy is only 17.39%. We notice that the AA of both ODL-LP (82.18%) and TDDL-LP (83.86%) are at least 4% higher than that of the other methods. This also suggests that the Laplacian-

sparsity-enforced dictionary learning methods work better on non-homogeneous regions, since the AA can only attain high value when both the most regions reach high accuracy.

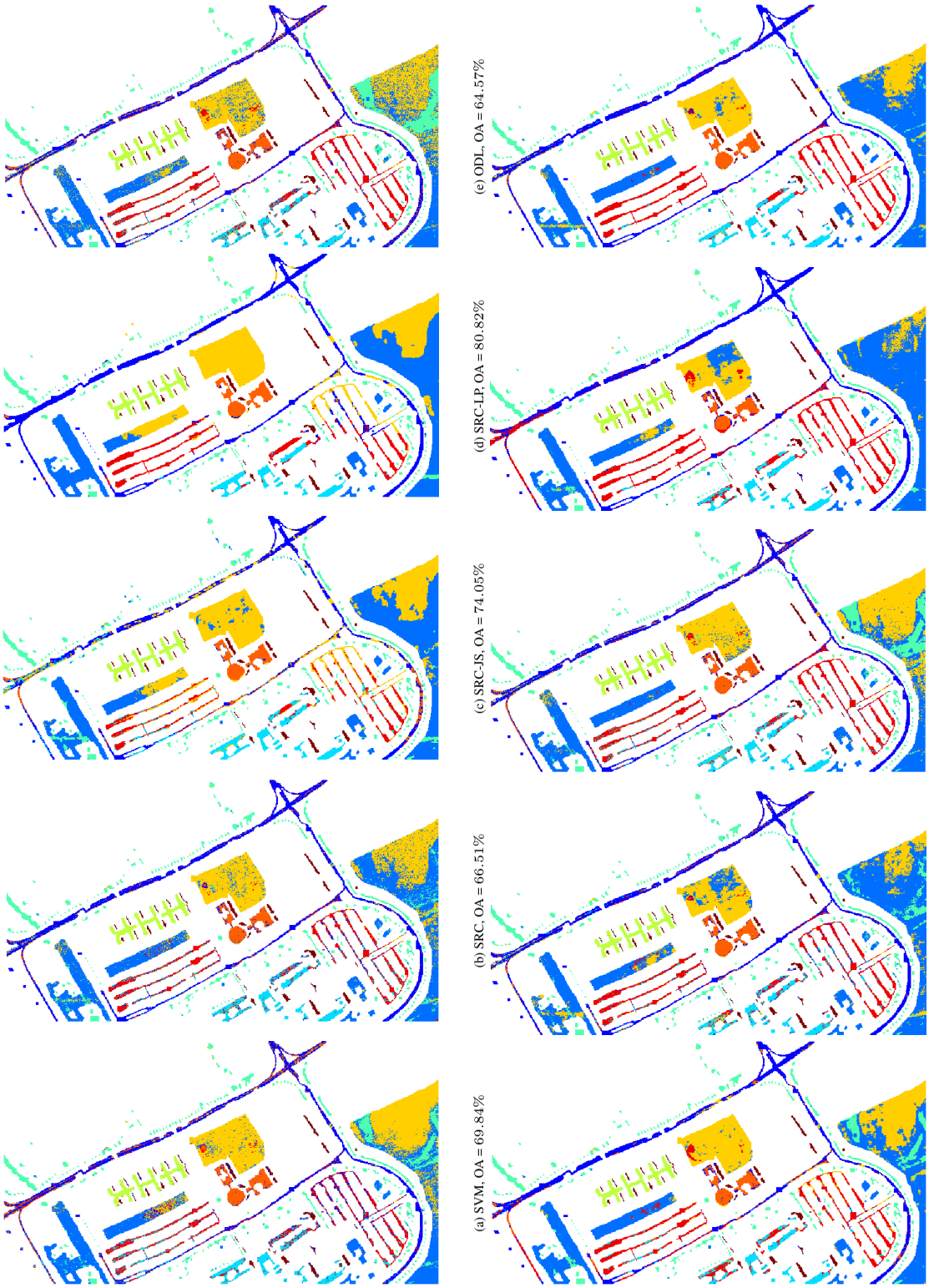


Figure 3.6: Classification map of the University of Pavia image obtained by (a) SVM, (b) SRC, (c) SRC-JS, (d) SRC-LP, (e) ODL, (f) ODL-JS, (g) ODL-LP, (h) TDDL, (i) TDDL-JS and (j) TDDL-LP.

Table 3.5: Classification accuracy (%) for the University of Pavia image

Dictionary Size		$N = 3921$			$N = 45$					
Class	SVM	SRC	SRC-JS	SRC-LP	ODL	ODL-JS	ODL-LP	TDDL	TDDL-JS	TDDL-LP
1	84.55	57.11	77.04	**95.08**	39.16	86.64	79.38	74.60	79.27	87.77
2	82.45	58.22	67.98	66.70	66.37	56.48	75.89	51.27	**86.85**	78.89
3	77.08	57.33	44.32	77.55	65.40	**80.72**	62.42	77.19	71.13	78.79
4	94.19	95.94	95.13	95.19	78.67	**99.04**	96.91	98.08	98.87	98.21
5	99.01	**100.00**	99.85	**100.00**	99.91	100.00	99.82	99.91	99.91	99.91
6	23.55	89.60	88.31	96.60	64.94	**96.89**	72.13	90.07	68.74	91.64
7	2.06	83.27	96.59	**96.59**	91.64	91.23	84.10	86.14	68.09	93.17
8	33.89	48.65	65.20	67.36	67.36	90.81	75.98	78.00	95.54	**94.20**
9	53.05	93.69	99.59	**99.59**	71.07	98.37	93.46	95.72	91.82	95.09
OA[%]	69.84	66.51	74.05	80.82	64.57	75.83	78.15	69.30	84.48	**85.70**
AA[%]	61.09	75.98	80.06	88.80	71.66	88.91	82.23	83.44	84.47	**90.85**
κ	0.569	0.628	0.681	0.758	0.549	0.731	0.747	0.662	0.817	**0.835**

3.5.3 Performance on ROSIS Pavia Urban Dataset

The last two images to be tested are the University of Pavia and the Center of Pavia, which are urban images acquired by the Reflective Optics System Imaging Spectrometer (ROSIS). It generates 115 spectral bands ranging from 0.43 to 0.86μm.

The University of Pavia image contains 610×340 pixels. 12 noisiest bands out of all 115 bands are removed. There are nine ground-truth classes of interests as shown in Table 3.4. For this image, the training samples were manually labelled by an analyst. The total number of training and testing samples is $3,921$ (10.64% of all the interested pixels) and $40,002$ respectively. The training and testing map are visually displayed in Fig. 3.5.

For the University of Pavia, we set the total iterations of unsupervised and supervised dictionary learning methods to be 30 and 200 respectively. The window size is set to 5×5 for all joint or Laplacian sparse regularized methods to obtain the highest OA. The ODL-LP is able to reach a performance of 78.15% for OA, which is more than 14% higher than that of ODL. The ODL-JS also significantly improves the OA, which is more than 11% higher than that of ODL. TDDL-LP has the highest OA = 85.70%, which indicates that it

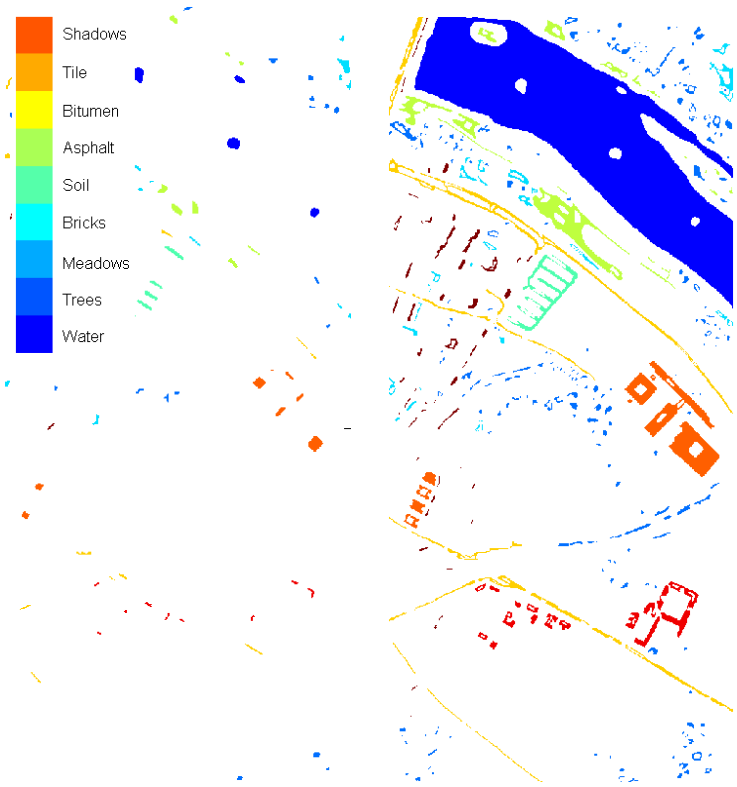
Table 3.6: Number of training and test samples for the Center of Pavia image

Class #	Name	Train	Test
1	Water	745	64533
2	Trees	785	5722
3	Meadows	797	2094
4	Bricks	485	1667
5	Soil	820	5729
6	Asphalt	678	6847
7	Bitumen	808	6479
8	Tile	223	2899
9	Shadows	195	1970
Total		5536	97940

outperforms other methods when classify large regions of the image. It also has the highest $\kappa = 0.935$. The best classification accuracy for class 1 (Asphalt), which consists of narrow strips, is obtained by using TDDL-LP (87.77%). Class 2 (Meadows) is composed of large smooth regions, as expected, TDDL-JS gives the highest accuracy (86.85%) for this class. TDDL has large amount of misclassification pixels for class 2. The highest AA (90.85%) is given by TDDL-LP, which confirms that the TDDL-LP is superior to other methods when classify the pixels in non-homogeneous regions.

The third image where we evaluate various approaches is the Center of Pavia, which consists of 1094×492 pixels. Each pixel has 102 bands after removing 13 noisy bands. This image consists of nine ground-truth classes of interest as shown in Table 3.6 and Fig. 3.7. $5,536$ manually labelled pixels are designated as the training samples and the remaining $97,940$ interested pixels are used for testing.

Since this image has more labeled samples than the other two images, we set the total iterations of unsupervised and supervised dictionary learning methods to be 75 and 1000



(a) (b)

Figure 3.7: (a) Training sets and (b) test sets of the Center of Pavia image.

respectively. The window size is set to 5×5 for the joint sparse and Laplacian sparse regularized methods. Although the OA of most methods are close, the OA of ODL-JS and ODL-LP are still around 3% higher than that of ODL. The TDDL-LP reach the highest OA = 98.67% over all the other methods. The OA of TDDL-JS (98.01%) is slightly lower than that of the TDDL-LP. We notice that SRC-JS (OA = 98.01%) and SRC-LP (OA=98.36%) also render competitive performance when compared to TDDL-JS and TDDL-LP due to the fact that the raw spectral features of this image is already highly discriminative. TDDL-LP outperforms other methods on almost all classes and works especially well for Class 4 (Bricks), achieving highest accuracy of 97.41%. Except for SRC-LP where the accuracy

Table 3.7: Classification accuracy (%) for the Center of Pavia image

Dictionary Size		$N = 5536$			$N = 45$					
Class	SVM	SRC	SRC-JS	SRC-LP	ODL	ODL-JS	ODL-LP	TDDL	TDDL-JS	TDDL-LP
1	96.97	99.58	99.52	99.28	96.26	99.13	**99.69**	98.54	98.76	99.13
2	91.09	90.07	96.89	92.11	84.25	94.63	90.63	89.55	**97.59**	93.01
3	96.08	95.42	99.47	98.62	93.36	96.23	97.61	95.18	96.85	**98.71**
4	86.32	79.96	78.28	94.72	61.61	64.73	97.30	85.78	85.18	**97.41**
5	88.57	93.70	97.05	97.14	89.40	84.62	90.00	88.08	98.25	**99.59**
6	95.27	95.62	98.19	97.18	94.35	95.03	94.49	94.39	**99.36**	99.18
7	94.03	93.86	97.01	96.84	86.31	86.90	97.33	91.65	94.46	**98.66**
8	99.83	99.17	99.66	99.66	96.76	99.79	99.00	98.17	99.38	**99.73**
9	85.74	98.58	99.19	**99.95**	93.25	90.56	94.42	95.53	91.27	95.61
OA[%]	95.68	97.57	98.01	98.36	93.67	96.13	97.86	96.30	98.01	**98.67**
AA[%]	93.77	94.00	95.03	97.28	88.39	90.18	95.61	92.99	95.68	**97.89**
κ	0.923	0.961	0.965	0.971	0.899	0.938	0.965	0.940	0.968	**0.979**

is 94.72%, none of others reaches accuracy over 90% for Class 4. Additionally, the AA of TDDL-LP (97.21%) is almost 2% better than that of TDDL-JS (95.68%). These results support our assertion that the Laplacian sparsity prior provides stronger discriminability on nonhomogeneous regions. Performance comparison between the SRC-based and TDDL-based methods have shown that the dictionary size can be drastically decreased by applying supervised dictionary learning while achieving even better performance.

3.6 Summary

In this chapter, we proposed novel a task driven dictionary learning method with joint or Laplacian sparsity prior for HSI classification. The corresponding optimization algorithms are developed using fixed point differentiation, and are further simplified for ease of implementation. We also derived the optimization algorithm for solving the Laplacian sparse recovery problem using SpaRSA, which improves the computational efficiency due to the availability of a more accurate descent direction. The performance and the behavior

of the proposed methods, i.e. TDDL-JS and TDDL-LP, have been extensively studied on the popular hyperspectral images. The results confirm that both TDDL-JS and TDDL-LP give plausible results on smooth homogeneous regions, while TDDL-LP one works better for classifying small narrow regions. Compared to TDDL-JS, TDDL-LP is able to obtain a more stable performance by describing the similarities of neighboring pixels' sparse codes more delicately. The results also confirm that a significantly better performance can still be achieved when joint or Laplacian prior is imposed by using a very small dictionary. The overall accuracy of our algorithm can be improved by applying kernelization to the proposed approach. This can be achieved by kernelizing the sparse representation [33] and using a composite kernel classifier [71].

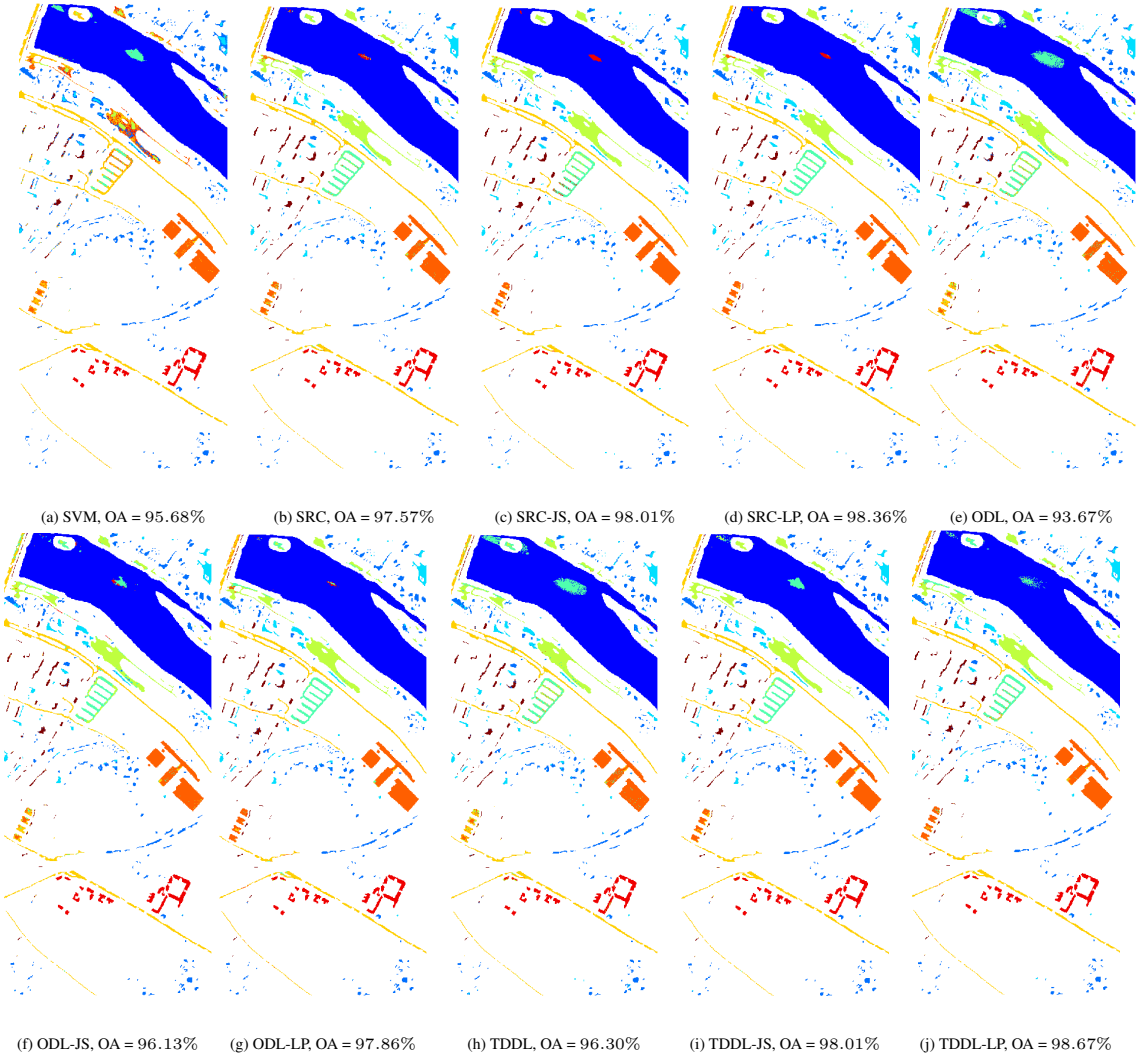


Figure 3.8: Classification map of the Center of Pavia image obtained by (a) SVM, (b) SRC, (c) SRC-JS, (d) SRC-LP, (e) ODL, (f) ODL-JS, (g) ODL-LP, (h) TDDL, (i) TDDL-JS and (j) TDDL-LP.

Chapter 4

Invariant Single Layer Sparse Coding

In this chapter, we show how to enforce invariant property for sparse coding using large displacement optical flow. Although sparse representation-based classifiers have shown outstanding accuracy and robustness in image classification tasks even with the presence of intense noise and occlusion, it has been discovered that the performance degrades significantly either when test image is not aligned with the dictionary atoms or the dictionary atoms themselves are not aligned with each other, in which cases the sparse linear representation assumption fails. In this chapter, having both training and test images misaligned, we introduce a novel sparse coding framework that is able to efficiently adapt the dictionary atoms to the test image via large displacement optical flow. In the proposed algorithm, every dictionary atom is automatically aligned with the input image and the sparse code is then recovered using the adapted dictionary atoms. A corresponding supervised dictionary learning algorithm is also developed for the proposed framework. Experimental results on

digit datasets recognition verify the efficacy and robustness of the proposed algorithm.

4.1 Alignment Issue with Sparse Representation Classifier

SRC has been found to be highly sensitive to the misalignment of the image dataset: a small amount of image distortion due to translation, rotation, scaling and 3-dimensional pose variations can lead to a significant degradation on the classification performance [72]. One straightforward way to solve the misalignment problem is to register the test image with dictionary atoms before sparse recovery. By assuming the dictionary atoms are registered, Wagner *et al.* [72] parameterize the misalignment of the test image with an affine transformation. These parameters are optimized using generalized Gauss-Newton methods after linearizing the affine transformation constraints. By minimizing the sparse registration error iteratively and sequentially for each class, their framework is able to deal with a large range of variations in translation, scaling, rotation and even 3D pose variations. Due to the adoption of holistic features, sparse coding is more robust and less likely to overfit.

In the case of local feature-based sparse coding, max pooling strategy [73] is often employed over the neighboring coefficients to produce local translation-invariant property. Based on spatial pyramid matching framework, Yang *et. al.* [74] proposed a local sparse coding model with local SIFT features followed by multi-scale max pooling. The results on several large variance datasets achieved plausible performance that can hardly be pur-

sued by simply applying holistic sparse coding. To improve the discriminability of the sparse codes, their dictionary was trained with supervised learning via backpropagation [63]. Classification performance of local feature-based sparse coding has also been evaluated on several large datasets in [75], demonstrating a state-of-art performance that is competitive with deep learning [76, 77]. Another interesting approach is the convolutional sparse coding [78], where the local features are reconstructed by convoluting the local sparse codes using local dictionary. Visualization of its dictionary shows that the dictionary atoms contain more complex features, therefore having more discriminative power.

In this chapter, we present a novel sparse coding framework that is robust to image transformation. In the proposed model, each dictionary atom is constructed in the form of a tensor and is aligned with the test image using the large displacement optical flow concept [79]. We show experimentally that the proposed sparse coding framework outperforms most other sparsity-based methods. Specifically, our chapter has the following novelties and contributions: *(i)* The proposed algorithm does not require the training dataset to be pre-aligned. *(ii)* Adapting the dictionary to the input test image is highly efficient: requiring only $O(PT)$ operations for adapting each dictionary atom, where T is the number of pixels in a searching window and P is the total number of *subatoms* to be aligned. *(iii)* Supervised dictionary learning algorithm is developed for the proposed sparse coding framework.

The remainder of the chapter is organized as follows: We first introduce the proposed sparse coding framework for dealing with dataset misalignment in Section 4.2.1. Next, in Section 5.3, we show how to train the dictionary in a supervised manner by solving a

bilevel optimization problem. Finally, in Section 6.4, experimental results demonstrate that the proposed framework has a state-of-art performance, which is more promising over most existing sparsity-based methods.

Figure 4.1: Proposed sparse coding framework: Dictionary tensor atoms $\{\mathbf{D}_n\}_{n=1}^N$ and the test tensor image $\mathbf{X}$ are shown in the lower part of the figure. Searching window of size $T = 3 \times 3$ within each tensor atom is colored with purple. Each group of neighboring T subatoms $\mathbf{B}_{np}$ is matched with the corresponding vector pixel $\mathbf{x}_p$ of the test tensor image, resulting in an aligned subatom. After the matching process, the sparse code for $\mathbf{x}_p$ is recovered using all the aligned subatoms. For illustration purposes, only five dictionary tensor atoms are shown in the figure and the magnitude of the sparse codes are displayed with various intensities in red.

4.2 Large Displacement Optical Flow

4.2.1 Invariant Sparse Coding via Large Displacement Optical Flow

In this section, we first introduce how to construct the dictionary atoms and input images in the form of tensors. We then illustrate how to eliminate the misalignment by dynamically adapt the tensor dictionary atoms to the input tensor image.

In the proposed sparse coding model, as shown in Fig. 5.2, both dictionary atom and input image are represented by image tensors. Each pixel in the tensor image is a vectorized version of a local patch in the original image, referred to as a vector pixel. Denote the n^{th} tensor atom as $\mathbf{D}_n = [\mathbf{d}_{n1}, \ldots, \mathbf{d}_{nP}] \in \mathbb{R}^{M \times P}$ and a given test tensor image as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_P] \in \mathbb{R}^{M \times P}$, where $\mathbf{d}_{np} \in \mathbb{R}^M$ is the p^{th} subatom of the n^{th} tensor atom and $\mathbf{x}_p \in \mathbb{R}^M$ is the p^{th} vector pixel of the input image. M is the dimension of vector pixel, n is the dictionary atom index and P is the total number of subatoms in the tensor atom, which is the same number of vector pixels in the test tensor image. The dictionary is denoted as $\mathbf{D} = [\mathbf{D}_1, \ldots, \mathbf{D}_N] \in \mathbb{R}^{M \times NP}$. Given a dictionary with N tensor atoms, a typical sparse recovery problem [18] is formulated as:

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{p=1}^{P} \| \sum_{n=1}^{N} \alpha_n \mathbf{d}_{np} - \mathbf{x}_p \|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1, \tag{4.1}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^\top \in \mathbb{R}^N$ is the sparse coefficient and $\lambda > 0$ is the regularization

parameter. Problem (4.1) is a standard form of ℓ_1-sparse recovery problem that can be efficiently solved using alternating direction method of multipliers (ADMM) [67].

When images in both the training and test datasets are misaligned, sparse coefficients recovered by solving the problem (4.1) become unreliable, thus resulting in poor classification performance. To alleviate the misalignment problem, we propose to register each tensor atom with the input test image via large displacement optical flow [79]. The notion of optical flow field is used here to describe the displacements of vector pixels within each tensor atom, and the sparse recovery is then performed by using only the best matching subatoms selected from the tensor atoms. The proposed framework is illustrated in Fig. 5.2. Denote $\mathbf{B}_{np} \in \mathbb{R}^{M\times T}$ as the T subatoms within the searching window centered at the location p of the n^{th} tensor atom. The recovery of the optical flow and sparse codes can be formally described as follows:

$$\begin{aligned} &(\hat{\boldsymbol{\alpha}}, \{\hat{\mathbf{c}}_{np}\}) = \arg\min_{\boldsymbol{\alpha},\{\mathbf{c}_{np}\}} \frac{1}{2}\sum_{p=1}^{P}\|\sum_{n=1}^{N}\alpha_n \mathbf{B}_{np}\mathbf{c}_{np} - \mathbf{x}_p\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1, \\ &\text{s.t.} \quad \|\mathbf{c}_{np}\|_0 = 1, \|\mathbf{c}_{np}\|_1 = 1, \mathbf{c}_{np} \geq \mathbf{0}, \\ &\qquad \forall n \in [N], p \in [P], \end{aligned} \tag{4.2}$$

where $\|\mathbf{c}_{np}\|_0 = 1$ is the cardinality constraint and $\mathbf{c}_{np} \in \mathbb{R}^T$ is the sparse index vector that is used to characterize the optical flow field. The constraint in (4.2) suggests that $\mathbf{c}_{np}$ is a binary index vector and only one element is nonzero, which means that it can only select one subatom within the searching window.

The optimization problem in (4.2) is a mixed-integer problem and NP-hard [80]. Therefore, we propose a heuristic algorithm to find an informative $\boldsymbol{\alpha}$ and the sparse index vectors $\{\mathbf{c}_{np}\}_{n,p=1}^{N,P}$ for all vector pixels. As shown in Fig. (5.2), the optical flow field for each vector pixel is found by searching for the best match between neighboring subatoms and the corresponding input vector pixel. In practice, we found that searching for the best match without involving the sparse code is the key to render plausible performance in both classification accuracy and computational efficiency. Formally, we propose to find a local optimum of problem (4.2) by solving the following optimization problem:

$$\begin{aligned}
\hat{\boldsymbol{\alpha}} &= \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{p=1}^{P}\|\sum_{n=1}^{N}\alpha_n \mathbf{B}_{np}\hat{\mathbf{c}}_{np} - \mathbf{x}_p\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1 \\
\text{s.t.} \quad \hat{\mathbf{c}}_{np} &= \arg\min_{\mathbf{c}_{np}} \frac{1}{2}\|\mathbf{B}_{np}\mathbf{c}_{np} - \mathbf{x}_p\|_2^2, \\
&\|\mathbf{c}_{np}\|_0 = 1, \|\mathbf{c}_{np}\|_1 = 1, \mathbf{c}_{np} \geq \mathbf{0}, \\
&\forall n \in [N], p \in [P].
\end{aligned} \tag{4.3}$$

In our approach, the sparse coding part of (4.3) is solved by using the alternating direction method of multipliers (ADMM) [67]. One important advantage of the above model is that it is highly computational efficient because it only takes $\mathcal{O}(T)$ operations to search for the best match for each vector pixel.

4.2.2 Supervised Dictionary Learning for Invariant Sparse Coding

In order to improve the efficiency of sparse coding and discriminablity of the dictionary, we employ the supervised dictionary learning framework [59, 63, 81] to optimize the dictionary and the classifier parameters simultaneously. Formulated as a bilevel optimization problem, the dictionary is updated using back propagation to minimize the classification error. Formally, the supervised dictionary learning problem can be formulated as follows:

$$\min_{\mathbf{W},\mathbf{D}} \mathbb{E}_{\mathbf{y},\mathbf{X}} \left[\ell(\mathbf{y}, \mathbf{W}\hat{\boldsymbol{\alpha}}(\mathbf{X}, \{\hat{\mathbf{c}}_{np}(\mathbf{D})\}, \mathbf{D}))\right] + \frac{\mu}{2}\|\mathbf{W}\|_F^2, \tag{4.4}$$

where $\ell(\cdot)$ is some smooth and convex function that is used to define the classification error and $\mu > 0$ is the regularization parameter used to alleviate the overfitting of the classifier. Due to the triviality of updating classifier parameters, here we only state the update for the dictionary:

$$\mathbf{D} \leftarrow \Pi(\mathbf{D} - \rho^t \cdot \partial\ell/\partial\mathbf{D}), \tag{4.5}$$

where $\rho > 0$ is the learning rate, t is the iteration counter and Π is the projection that regulate the Frobenius norm of every tensor atom to be one. Similar to [59, 63, 81], (4.4) suggests that the update of both the dictionary and the classifier are driven by reducing classification error. The local optima can be solved by using descent method [62] based on

error backpropagation. The sparse code $\boldsymbol{\alpha}$ is an implicit function of $\mathbf{X}, \{\mathbf{c}_{np}\}$ and $\mathbf{D}$. In addition, each optical flow field $\mathbf{c}_{np}$ is an implicit function of $\mathbf{D}$ and $\mathbf{x}_{np}$. Therefore, given an input image $\mathbf{X}$ and an optimal sparse code $\hat{\boldsymbol{\alpha}}$, apply the chain rule of differentiation, the direction along which the upper-level cost decreases can be formulated as:

$$\frac{\partial \ell(\mathbf{y}, \mathbf{W}\boldsymbol{\alpha})}{\partial \mathbf{D}} = \frac{\partial \ell}{\partial \boldsymbol{\alpha}} \frac{\partial \boldsymbol{\alpha}}{\partial \mathbf{D}} + \sum_{p=1}^{P} \frac{\partial \ell}{\partial \mathbf{C}_p} \frac{\partial \mathbf{C}_p}{\partial \mathbf{D}}, \tag{4.6}$$

where $\mathbf{C}_p = \bigoplus_{n=1}^{N} \bar{\mathbf{c}}_{np} \in \mathbb{R}^{NP \times N}$ and $\bigoplus$ denotes the direct sum. Also, $\bar{\mathbf{c}}_{np} \in \mathbb{R}^{NP}$ is obtained by zero-padding with $\mathbf{c}_{np}$, where $(N-1)P+1$ to NP elements of $\bar{\mathbf{c}}_{np}$ are from those of $\mathbf{c}_{np}$. Due to the binary constraints on $\{\mathbf{c}_{np}\}$, every element of the gradient $\partial \mathbf{C}_p / \partial \mathbf{D}$ equals to zero. On the other hand, the first part of the derivative can be solved by applying fixed point differentiation [66]. Due to the page limitation of the chapter and the triviality for deriving the term $\partial \ell / \partial \boldsymbol{\alpha}$, we only show the final derivation of $\partial \boldsymbol{\alpha} / \partial \mathbf{D}$ as follows:

$$\frac{\partial \boldsymbol{\alpha}_\Lambda}{\partial d_{mnp}} = \boldsymbol{\Theta}_{\Lambda,\Lambda}^{-1} \left(\frac{\partial (\mathbf{D}\mathbf{C}_p)_\Lambda^\top}{\partial d_{mnp}} \mathbf{x}_p - \frac{\partial \boldsymbol{\Theta}_{\Lambda,\Lambda}}{\partial d_{mnp}} \boldsymbol{\alpha}_\Lambda \right), \tag{4.7}$$

where Λ is the index set of active atoms of the sparse code $\boldsymbol{\alpha}$. $(\mathbf{D}\mathbf{C}_p)_\Lambda$ is the matrix obtained by collecting the active columns of $\mathbf{D}\mathbf{C}_p$, $\boldsymbol{\Theta} = \sum_{\rho=1}^{P} \mathbf{C}_p^\top \mathbf{D}^\top \mathbf{D} \mathbf{C}_p$ and $\boldsymbol{\Theta}_{\Lambda,\Lambda}$ is the submatrix obtained by selecting the active columns and rows of $\boldsymbol{\Theta}$. The matrix $\boldsymbol{\Theta}_{\Lambda,\Lambda}$ is always nonsingular since the total number of measurement MP is always significantly larger than the number of active atoms. Combining (5.11) with (5.15) for each dictionary element, the gradient for updating the dictionary can be achieved. For a large dataset, the

dictionary and the classifier parameters are updated in an online manner.

4.3 Experimental Verification

In this section, we evaluate the proposed algorithm on hand-written digits datasets including the MNIST and USPS. The sparse coding is performed with a single dictionary and linear SVM is used for classification. For a fair comparison, we only compare with the results that are produced with the same SRC strategy. The dictionary size in our chapter is set to be no larger than those used in other methods. Similar to [63], parameters in our experiments are chosen heuristically. The batch size for updating the dictionary is 512. Initial learning rate ρ is set to 0.001 and $\lambda = 0.01$.

4.3.1 Evaluation on the MNIST Database

MNIST [82] consists of a total number of $70,000$ images of digits, of which $60,000$ are training set and the rest $10,000$ are test set. Each digit is centered and normalized in a 28×28 field. The dictionary size N is set to be 150 for this database.

We first evaluate the performance of the proposed algorithm under various number of training samples. We follow the same experimental setting as in [83], examining the classification accuracy given the training size $\{300, 1K, 2K, 5K, 10K, 20K, 40K, 60K\}$. The performance is shown in Fig. 4.2 (a). The proposed method significantly outperforms the ℓ_1 sparse coding-based algorithm (L1SC) [59].

We then demonstrate the robustness of the proposed method towards various image deformations. Following a similar setting as in [72], we perform the translation along x direction, rotation and scaling separately only on the test samples. We report the classification accuracy with respect to various levels of deformation and compare the performance with L1SC. The experimental results are shown in Fig. 4.2(b)-(d). Performance of our method and L1SC are illustrated in red and blue lines, respectively. The shadow area at the bottom of each figure is the accuracy difference between the two methods. We can see for all three deformations, the proposed method consistently outperforms L1SC. In addition, the hump shape of the shadow area indicates that the proposed method is robust to numerous image deformations.

Finally, the error rate for the MNIST is shown in Table 4.1. Our method reaches the lowest error rate of 1.12%. On MNIST, differences of more than 0.1% are statistically significant [84]. Comparing with the second best algorithm, the proposed method reduces the error rate by 0.12%, exhibiting better generality and dictionary compactness.

4.3.2 Evaluation on the USPS Database

The USPS dataset has $7,291$ training and $2,007$ test images, where each of them is of size 16×16. Being compared to MNIST, the USPS dataset has a much larger variance and a smaller training set, which challenges the dictionary generality. For a fair comparison, the dictionary size N is set to be 80. Local patch size is 5×5 ($M = 25$). Searching window size is 5×5 ($T = 25$). The performance of various approaches on USPS database are depicted

Method	MNIST	USPS
CBN	1.95 (3×10^4)	4.14 (7291)
ESC [85]	5.16 (150)	6.03 (80)
Ramirez *et al.* [24]	1.26 (800)	3.98 (80)
Deep Belief Network [76]	1.25 (-)	- (-)
MMDL [86]	1.24 (150)	-(-)
Proposed	**1.12** (150)	**3.43** (80)
Improvements	9.7%	13.8%

Table 4.1: Error rate (%) on MNIST and USPS datasets. The dictionary size is shown in the parentheses. Improvements over the second best algorithm is shown in the last line.

in Table 4.1. Our algorithm achieves the lowest error rate 3.43% among other supervised learning-based methods. The experimental result validates the efficacy of our proposed algorithm on a dataset with a larger variance.

4.4 Summary

In this chapter, we present a novel sparse coding algorithm that is able to dynamically select the dictionary subatoms to adapt to the misaligned image dataset. In the proposed method, both the dictionary atoms and the input test image are represented by tensors, and each vector pixel in the tensor image is a vectorized local patch. Each tensor atom is aligned with the input tensor image using large displacement optical flow, which is highly computationally efficient. Using the fixed point differentiation, a supervised dictionary learning algorithm is developed for the proposed sparse coding framework, which significantly reduces the required dictionary size.

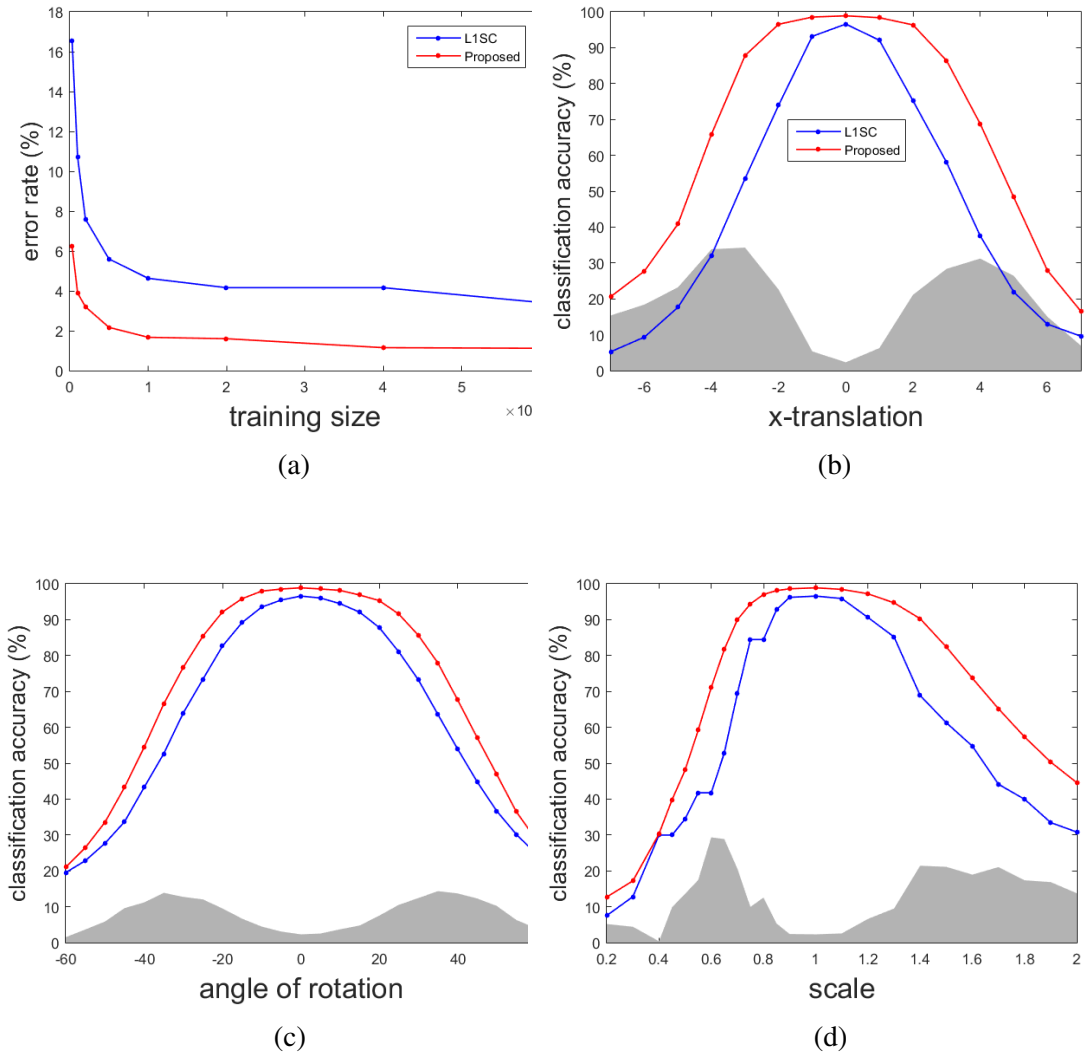


Figure 4.2: The proposed method demonstrates plausible performance on MNIST digits recognition with a small number of training samples. It also demonstrates robustness towards various image deformations. Classification accuracy of different experimental settings are shown in the above sub-figures: (a) Error rate under various sizes of training samples. (b) Translation along x direction versus classification accuracy. (c) In-plane rotation only. (d) Scale variation only. In (b)-(c), red and blue lines are the results of the proposed method and L1SC, respectively. Gray shadow area at the bottom of each figure is the accuracy difference between the proposed method and L1SC.

Chapter 5

Unsupervised Multilayer Invariant Sparse Coding for Large Dataset

Invariant feature extraction has always been pursued in object recognition algorithms due to its significance in enhancing the classifier generality towards unseen samples. In this chapter, we introduce a novel local feature-based hierarchical framework to produce invariant sparse codes for object recognition. In order to enforce the invariant property for each sample patch (local feature descriptor) in the image, its sparse code is recovered with a dedicated dictionary whose atoms are adaptively chosen from several bags of candidate atoms. The single-layer invariant sparse coding model is naturally extended to a multi-layer hierarchical architecture to further improve the invariance of the sparse codes. We show that the proposed hierarchical sparse coding model is able to generate complex invariant properties with layer-wise unsupervised dictionary learning. Experimental results on the

popular image datasets, including MNIST, CIFAR-10 and STL-10, verify the efficacy and robustness of the proposed algorithm.

5.1 Introduction

Sparse coding has been successfully applied to numerous computer vision tasks. However, when the objects in the images have noticeable variations in illumination, translation, scaling and rotation, these objects are no longer effectively captured by the atoms of the SRC dictionary. In this case, the underlying linear assumption of SRC is violated and its classification performance plunges.

To handle the misalignment problem, Wagner *et al.* [72] proposed to align the test image with dictionary atoms before performing sparse recovery. Their framework is able to handle a large range of variations in translation, scaling, rotation and even 3D pose variation. Similar methods have also been proposed in [87, 88] to handle the misalignment in face recognition or to perform dynamic scene registration. However, these methods are based on the holistic sparse coding, which is usually intractable when the size of image is large or when the scene is complex with multiple objects of interest. Besides, the performance of these methods are limited by the enormous variabilities in the image database where most of these variations cannot be parameterized into any known transformation group [89].

On the other hand, a patch set-based method in [90] represents each image with numer-

ous centers of patch clusters so that their location information is discarded. Unfortunately, since the local features are usually highly correlated, the recovered sparse codes are usually unstable, resulting in severe overfitting. Numerous works based on collaborative sparse recovery [42, 91] have also been proposed to improve the robustness of the sparse codes.

However, neither the local sparse coding nor the collaborative sparse recovery enforce any invariance. To efficiently embed the invariant property within the sparse coding-based image classifiers, max pooling process is usually employed after the sparse recovery stage, mimicking an architecture similar to the convolutional neural networks (CNN) [92]. Based on the spatial pyramid matching framework, Yang *et. al.* [20] proposed a local sparse coding model with multi-scale max pooling using local SIFT features. Extracting invariant features via group sparsity has also been proposed in [93], where each group of dictionary atoms represents complex discriminative features. In a more direct approach for enforcing invariant property, Sohn *et. al.* [94] explicitly employed linear transformation to handle the image deformation in order to achieve invariance over numerous image transformations. Another interesting approach is convolutional sparse coding [78], where the local features are reconstructed by convoluting the local sparse codes with a local dictionary. Visualization of their learned dictionary suggests that the dictionary atoms contain more complex features when the sparse coding is locally translation-invariant.

Most of these local feature-based methods achieve invariant property through either bypassing the alignment problem [42, 90, 91] or applying post processing [2, 20, 63], such as max pooling. Few attempts have been made to simultaneously achieve an inherently invari-

ant sparse code through a sparse recovery method with embedded misalignment compensation. In this chapter, we develop a novel sparse coding framework that is able to produce invariant sparse codes by adaptively constructing a dedicated dictionary for each input sample patch during the sparse recovery stage. Layer-wise unsupervised dictionary learning is also developed for the proposed sparse coding framework. The proposed invariant sparse coding model is formulated in an intuitively similar way as archetypal analysis [95, 96], where each adaptive dictionary acts as archetype patterns. For a given input sample, an adaptive dictionary is constructed from several sub-dictionaries and we refer to each sub-dictionary as a Bag of Atoms (BoA), where each of them is akin to Bag of Words (BoW) [97]. We distinguish BoA from BoW in this chapter since the latter one is usually used directly as a dictionary for sparse recovery in the literature [2, 20] while instead we use a large number of BoAs for constructing an adaptive dictionary, with which the sparse code is recovered. Moreover, the sparse recovery is performed using the collaborative representation so that the invariant property can be enforced with unsupervised dictionary learning and naturally extended to a multi-layer hierarchical architecture. Specifically, our proposed model has the following contributions:

- We propose to construct an adaptive dictionary which is dedicated to each input sample in order to produce inherently invariant sparse code under data variations. Unlike CNN-based sparse coding methods whose invariant features are obtained with max-pooling operations, the invariant property of the local sparse codes in the proposed framework is enforced and integrated into the sparse recovery stage, rendering sparse

codes that are more stable and more invariant.

- Based on a local feature descriptor matching method, we propose to use several BoAs to adaptively build dedicated dictionaries, which are highly robust and efficient in sparsely capturing the object of interest.

- We propose a novel method to adaptively construct a dedicated dictionary for each input sample from several BoAs in order to produce inherently invariant sparse codes.

- We develop a layer-wise unsupervised dictionary learning algorithm with bilevel optimization in order to simultaneously minimize the errors of local feature descriptor matching and signal reconstruction. Using our layer-wise unsupervised dictionary learning algorithm, we are able to design diverse and contextually rich BoAs.

The rest of the chapter is organized as follows: We first briefly review the local translation invariant sparse coding proposed in [63, 75] and formulate the proposed hierarchical invariant sparse coding framework in Section 5.2. Layer-wise unsupervised dictionary learning is developed in Section 5.3. Experimental results on three publicly available dataset, including MNIST, CIFAR-10 and STL-10, are demonstrated in Section 6.4. Finally, we summarize the chapter with advantages and disadvantages of the proposed model in Section 5.5.

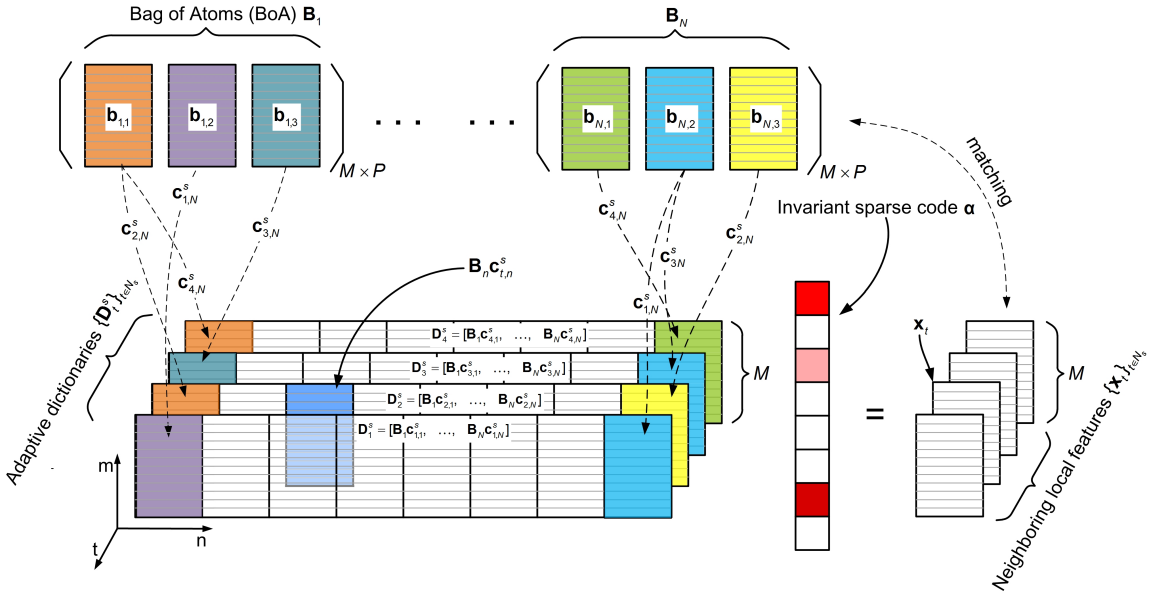


Figure 5.1: Proposed invariant sparse coding framework: N Bags of candidate atoms $\{\mathbf{B}_n\}_{n=1}^{N}$ are shown in the upper part of the figure. The input neighboring local feature vectors $\{\mathbf{x}_t\}_{t\in N_s}$ and the adaptive dictionaries $\{\mathbf{D}_t^s\}_{t\in N_s}$ are shown in the lower part of the figure. Each bag of atoms $\mathbf{B}_n$ is matched against an input local feature descriptor $\mathbf{x}_t$, resulting in an adaptive dictionary atom $\mathbf{B}_n\hat{\mathbf{c}}_{t,n}^s$. After this matching process, the invariant sparse code $\boldsymbol{\alpha}$ for neighboring local feature vectors $\{\mathbf{x}_t\}_{t\in N_s}$ is recovered using all the adaptive dictionaries $\{\mathbf{D}_t^s\}_{t\in N_s}$ simultaneously. In the above illustrative figure, the number of atoms P in each BoA is 3, the number of neighboring features vectors T is 4, and the total number of BoA N is 8.

5.2 Hierarchical Invariant Sparse Coding with Adaptive Dictionary

Suppose an image is represented by a set of local feature descriptors $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^S] \in \mathbb{R}^{M\times S}$, where $\mathbf{x}^s \in \mathbb{R}^M$ is the s^{th} local feature of the image. At each image pixel we generate a local feature vector, which is obtained by vectorizing all the pixels within a local patch or extracting a local SIFT feature descriptor from the local patch. In the rest of the

chapter, we use the term pixel to refer to our feature vector at each image pixel location. Let $\{\mathbf{x}_t\}_{t\in\mathcal{N}_s}$ be a set of neighboring local feature vectors that are extracted from a window centered at the pixel $\mathbf{x}^s$, where $\mathcal{N}_s$ is an index set that includes all the pixel indexes in the specified window. In this chapter, we assume that all the neighboring local features $\{\mathbf{x}_t\}_{t\in\mathcal{N}_s}$ have originated from a single latent invariant feature $\mathbf{z}^s \in \mathbb{R}^{M1}$, which can be encoded with a single invariant sparse code $\boldsymbol{\alpha}^s \in \mathbb{R}^N$. An interesting question is whether we can recover the archetypal invariant sparse code $\boldsymbol{\alpha}^s$ from the local warped features $\{\mathbf{x}_t\}_{t\in\mathcal{N}_s}$?

In the sparse coding-based approach [63], the neighboring local features are encoded separately and the invariant sparse code is reached by a pooling strategy. Specifically, given a dictionary $\mathbf{D} \in \mathbb{R}^{M\times N}$, the sparse codes $\{\boldsymbol{\alpha}_t^s\}_{t\in\mathcal{N}_s}$ corresponding to the local feature vectors $\{\mathbf{x}_t\}_{t\in\mathcal{N}_s}$ are recovered by solving the following problem:

$$\hat{\boldsymbol{\alpha}}_t^s = \arg\min_{\boldsymbol{\alpha}_t^s} \frac{1}{2}\|\mathbf{D}\boldsymbol{\alpha}_t^s - \mathbf{x}_t\|_2^2 + \lambda\|\boldsymbol{\alpha}_t^s\|_1, \forall t \in \mathcal{N}_s, \tag{5.1}$$

where $\lambda > 0$ is the regularization parameter and $\hat{\boldsymbol{\alpha}}_t^s \in \mathbb{R}^N$ is the recovered sparse code for input $\mathbf{x}_t^s$. Solving problem (5.1) alone cannot achieve the invariant sparse code since the neighboring local features belong to different subspaces, max pooling is therefore applied to force all neighboring sparse codes to be the same in order to generate a single invariant

[1]The invariant archetypal feature $\mathbf{z}^s$ is not unique.

sparse code $\hat{\boldsymbol{\alpha}}^s \in \mathbb{R}^N$ at image location s:

$$(\hat{\boldsymbol{\alpha}}^s)_i = \max\{|(\hat{\boldsymbol{\alpha}}^s_1)_i|, \ldots, |(\hat{\boldsymbol{\alpha}}^s_T)_i|\}, \forall i = 1, \ldots, N, \tag{5.2}$$

where $|(\hat{\boldsymbol{\alpha}}^s_t)_i|$ is the absolute value of $(\hat{\boldsymbol{\alpha}}^s_t)_i$, which is the i^{th} element of $\hat{\boldsymbol{\alpha}}^s_t$. Major drawback for this pooling-based strategy is that, it forcefully combines the subspaces that belong to every neighboring local features all together and produces a sparse code which is not inherently invariant.

In this chapter, we are trying to recover an invariant sparse code for each input during the sparse recovery stage. As we will see, instead of using a single dictionary for recovering the sparse codes of all the neighboring local features, we propose to construct a dedicated dictionary for each input local feature. Interestingly, the proposed model has a formulation similar to that of the archetypal analysis [96].

5.2.1 Invariant Sparse Coding with Adaptive Dictionaries

In pursuit of invariant sparse representation, we would like to find a set of adaptive dictionaries $\{\mathbf{D}^s_t\}_{t\in\mathcal{N}_s}$ to recover the invariant sparse code $\boldsymbol{\alpha}^s$ from the neighboring feature vectors $\{\mathbf{x}_t\}_{t\in\mathcal{N}_s}$, where

$$\mathbf{D}^s_t = [\mathbf{d}^s_{t,1}, \ldots, \mathbf{d}^s_{t,N}] \in \mathbb{R}^{M\times N}, \tag{5.3}$$

and $\mathbf{d}_{t,n}^s \in \mathbb{R}^M$ is the n^{th} atom of the adaptive dictionary $\mathbf{D}_t^s$. As shown in Fig. 5.1, we describe every adaptive dictionary atom $\mathbf{d}_{n,t}^s$ as a linear combination of atoms from a BoA, $\mathbf{B}_n = [\vec{b}_{n,1}, \ldots, \vec{b}_{n,P}] \in \mathbb{R}^{M \times P}$, where each $\mathbf{b}_{n,p} \in \mathbb{R}^M$ is a candidate atom[2]. Formally, for all the adaptive dictionaries $\{\mathbf{D}_t^s\}_{t \in \mathcal{N}_s}$, where

$$\mathbf{d}_{t,n}^s = \mathbf{B}_n \mathbf{c}_{t,n}^s, \quad \forall n \in [N], \forall t \in \mathcal{N}_s, \tag{5.4}$$

and $\mathbf{c}_{t,n}^s \in \mathbb{R}^P$ is the matching coeffecient such that $\mathbf{c}_{t,n}^s > 0$, $\|\mathbf{c}_{t,n}^s\|_0 = 1$, define the invariant sparse code $\boldsymbol{\alpha}^s$ as the minimizer of

$$\hat{\boldsymbol{\alpha}}^s = \arg\min_{\boldsymbol{\alpha}^s} \frac{1}{2} \sum_{t \in \mathcal{N}_s} \|\mathbf{D}_t^s \boldsymbol{\alpha}^s - \mathbf{x}_t\|_2^2 + \frac{\lambda}{T} \|\boldsymbol{\alpha}^s\|_1, \tag{5.5}$$

where $\|\mathbf{c}_{t,n}^s\|_0$ is the ℓ_0 or the cardinality norm, and $\|\boldsymbol{\alpha}^s\|_1 = \sum_{n=1}^N |(\boldsymbol{\alpha}^s)_n|$ is the ℓ_1-norm of the sparse codes. The above formulation is similar to that of the archetypal analysis. Combining (5.3) and (5.4), we can see that each adaptive dictionary $\mathbf{D}_t^s$ in (5.5) is formulated as

$$\mathbf{D}_t^s = [\mathbf{B}_1 \mathbf{c}_{t,1}^s, \ldots, \mathbf{B}_N \mathbf{c}_{t,N}^s]. \tag{5.6}$$

Each optimal matching coefficient $\hat{\mathbf{c}}_{t,n}^s$ is obtained by solving the following ℓ_0-minimization

[2]Each BoA may contain P different of candidate atoms, yet for illustrative purpose, we define the number P is the same for every BoA.

problem:

$$\begin{aligned} \hat{\mathbf{c}}_{t,n}^{s} &= \arg\min_{\mathbf{c}_{t,n}^{s}} \|\mathbf{B}_n \mathbf{c}_{t,n}^{s} - \mathbf{x}_t\|_2^2 \\ \text{s.t.} \quad & \|\mathbf{c}_{t,n}^{s}\|_0 = 1, \mathbf{c}_{t,n}^{s} \geq 0, \ \forall n \in [N], \forall t \in \mathcal{N}_s, , \forall s \in [S] \end{aligned} \tag{5.7}$$

where $\|\mathbf{c}_{t,n}^{s}\|_0$ is the ℓ_0-norm or the cardinality of $\mathbf{c}_{t,n}^{s}$. The matching coefficient is optimized using the nonnegative orthogonal matching pursuit (OMP) [98] by setting the sparsity level to one [3]. The formulation of the matching problem (5.7) leads to several advantages over previous works: First, it is able to handle illumination variance since the magnitude of $\mathbf{c}_{t,n}^{s}$ is unconstrained. Second, it forces the coefficients $\mathbf{c}_{t,n}^{s}$ to have exactly one nonzero element: we assume that the candidate atoms in the same BoA do not belong to the same linear subspace since they are generated by nonlinear transformation of a latent archetypal local feature. Therefore, further increase in the number of nonzero elements in $\mathbf{c}_{t,n}^{s}$ is not necessary and can cause high coherency of atoms during the dictionary design. Finally, the matching coefficient is computed through matching with the original input sample $\mathbf{x}_t$. On the other hand, in previous works such as the group sparsity-based invariant feature [93], each atom in the dictionary is implicitly matched with a residue of the original input sample, eventually leading to unstable sparse codes under data variations.

The proposed invariant sparse coding for a set of given input neighboring samples

[3] In this chapter, we are only interested in enforcing pixel-level instead of subpixel-level invariant property.

$\{\mathbf{x}_t\}_{t\in\mathcal{N}_s}$ can be formulated compactly by combining (5.5), (5.6) and (5.7):

$$\begin{aligned}
&\hat{\boldsymbol{\alpha}}^s = \min_{\boldsymbol{\alpha}^s} \frac{1}{2}\sum_{t\in\mathcal{N}_s}\|\sum_{n=1}^{N}(\boldsymbol{\alpha}^s)_n\mathbf{B}_n\hat{\mathbf{c}}^s_{t,n} - \mathbf{x}_t\|_2^2 + \frac{\lambda}{T}\|\boldsymbol{\alpha}^s\|_1, \\
&\text{s.t.} \quad \hat{\mathbf{c}}^s_{t,n} = \arg\min_{\mathbf{c}^s_{t,n}}\|\mathbf{B}_n\mathbf{c}^s_{t,n} - \mathbf{x}_t\|_2^2, \\
&\qquad \|\mathbf{c}^s_{t,n}\|_0 = 1, \mathbf{c}^s_{t,n} \geq 0,\ \forall n \in [N], t \in \mathcal{N}_s, \forall s \in [S].
\end{aligned} \tag{5.8}$$

From (5.8), we can see that the variance of the input samples is captured by the matching coefficients $\mathbf{c}^s_{t,n}$ and thus the sparse code becomes invariant. Furthermore, the employment of collaborative sparse recovery over the neighboring features in (5.8) improves the stability of the sparse codes by achieving a sparser solution. More importantly, it generates a single output feature vector by integrating the information from all neighboring input samples so that this local invariant sparse feature vector can be directly used as input to the next layer in a multi-layer architecture. The proposed sparse recovery stage for problem (5.8) is solved by using the alternating direction method of multipliers (ADMM) [67].

5.2.2 Hierarchical Invariant Sparse Coding

Extending the proposed invariant sparse coding model to a multi-layer hierarchical architecture is necessary for pursuing sparse codes that are progressively invariant and discriminative. The proposed sparse recovery (5.8) not only produce invariant sparse codes, but combines all the information within the neighborhood window into a single sparse output, which is highly stable and consists of more complicated patterns compared with its

input counterpart. Formally, at each layer $h = 1, \ldots, H$, let its input be represented by a set of local features $\mathbf{X}^h = [\mathbf{x}^{1_h}, \ldots, \mathbf{x}^{S_h}] \in \mathbb{R}^{M_h \times S_h}$. To get the output of layer h, we first apply invariant sparse recovery (5.8) on every neighborhood local features of $\mathbf{X}^{(h)}$ as shown in Fig. (5.2). Let $\mathbf{A}^{(h)} = [\boldsymbol{\alpha}_1^{(h)}, \ldots, \boldsymbol{\alpha}_S^{(h)}] \in \mathbb{R}^{M_{(h+1)} \times S_h}$, then the output of layer h is:

$$\mathbf{Y}^{(h)} = g(\mathbf{A}^{(h)}), \tag{5.9}$$

where $g(\cdot)$ is a pooling function on $\mathbf{A}$, $\mathbf{X}^{(h+1)} = \mathbf{Y}^{(h)} \in \mathbb{R}^{M_{(h+1)} \times S_{(h+1)}}$. For the initial input layer $h = 0$, each column vector $\mathbf{x}_s^{(0)}$ of $\mathbf{X}^{(0)}$ is a vectorized local patch. The output sparse codes are concatenated together and fed into a linear classifier, which is chosen as the linear SVM in this chapter.

Similar to any other multi-layer architecture, overfitting can be a serious problem if not being dealt with properly. It has been observed that enforcing simply enforcing sparsity can lead to severe overfitting [20, 75] due to the intense variance of the output sparse codes at each layer. To alleviate the overfitting problem, we apply two strategies for the unsupervised dictionary learning and invariant sparse recovery. First, we adopt the dropout scheme that has been widely used in deep learning to avoid the scenario that certain atoms always being active. For each given input samples, we randomly drop a rate p of all the BoA. We found that this heuristic scheme is able to efficiently alleviate the coadaptation problem among BoA. Second, we only enforce the ℓ_0-norm constraint on $\mathbf{c}_{t,n}^s$ as already shown in problem (5.7), leaving its magnitude to be unconstrained so that the sparse code can be illumination invariant without normalizing the local sparse code descriptors. In

order to alleviate the overfitting for the proposed multi-layer architecture, avoiding the normalization of local descriptors is critical since it can make the hidden layer focus on learning the robust and stable inputs, which usually corresponds to a higher ℓ_2-norm due to the small reconstruction error. On the contrary, the hidden layer would try to absorb more information from the unstable inputs if they have the same ℓ_2-norm as the stable ones.

5.3 Layer-wise Unsupervised Dictionary Learning

In order to enhance the performance of invariant sparse coding, we employ the layer-wise unsupervised dictionary learning to acquire the BoAs. Formulated as a bilevel optimization problem, the dictionary is updated using back propagation to minimize both the reconstruction error (5.5) and the matching error (5.7). Given K number of input neighboring samples for layer h, the unsupervised dictionary learning problem can be formulated as follows:

$$\min_{\mathcal{B}} \sum_{i=1}^{K} \sum_{s=1}^{S} \ell(\{\mathbf{x}_t\}_{t\in\mathcal{N}_s}, \mathcal{B}, \{\mathbf{c}_{t,n}^s\}_{t\in\mathcal{N}_s}), \tag{5.10}$$

where $\mathcal{B} = \{\mathbf{B}_n\}_{n=1}^N$. We have dropped the layer index h and sample index i for simplicity. Problem (5.10) is separable across different neighboring features and can be updated by independently minimizing $\ell(\cdot)$. Therefore, we only focus on optimizing $\ell(\cdot)$ from now

on. Optimizations (5.8) and (5.10) suggest that the update of the dictionary is driven by reducing reconstruction error. The local optima can be solved by using gradient descent method [62] which is based on the error backpropagation algorithm. Each matching coefficient $\mathbf{c}_{t,n}^s$ is an implicit function of $\mathbf{B}_n$ and $\mathbf{x}_t, \forall t \in \mathcal{N}_s$. More specifically, we have $\mathbf{c}_{t,n}^s = \mathbf{c}_{t,n}^s(\mathbf{B}_n, \mathbf{x}_t), \forall t \in \mathcal{N}_s$. Given input neighboring samples $\{\mathbf{x}_t\}_{t\in\mathcal{N}_s}$, we apply the chain rule of differentiation, the direction along which the upper-level cost decreases can be formulated as:

$$\frac{\partial \ell}{\partial \mathbf{B}} = \frac{\partial \ell}{\partial \mathbf{B}} + \sum_{t=1}^{T}\sum_{n=1}^{N} \frac{\partial \ell}{\partial \mathbf{c}_{t,n}^s} \frac{\partial \mathbf{c}_{t,n}^s}{\partial \mathbf{B}_n}. \tag{5.11}$$

The derivation for the first term of (5.11) is trivial. We now focus on the derivation of the second term. Similar to the strategy adopted in [59, 63, 81, 86], we apply the fixed point differentiation to solve for $\partial \mathbf{c}_{t,n}^s / \partial \mathbf{B}_n$. Assuming $\max_{j\in\Lambda} |\mathbf{b}_{n,j}^\top \mathbf{c}_{t,n}^s|$ is strictly larger than $\max_{i\in\Lambda^c} |\mathbf{b}_{n,j}^\top \mathbf{c}_{t,n}^s|$, then the optimal solution $\hat{\mathbf{c}}_{t,n}^s$ of (5.8) which is given by nonnegative OMP satisfies:

$$\begin{cases} \left(\mathbf{B}_n^\top (\mathbf{B}_n \hat{\mathbf{c}}_{t,n}^s - \mathbf{x}_t)\right)_\Lambda = 0, \\ \left(\mathbf{B}_n^\top (\mathbf{B}_n \hat{\mathbf{c}}_{t,n}^s - \mathbf{x}_t)\right)_{\Lambda^c} \neq \mathbf{0}, \end{cases} \tag{5.12}$$

where $\Lambda \subset [N]$ is the active set, Λ^c is the complementary set. Here we have assumed that the matching error $\|\mathbf{B}_n \hat{\mathbf{c}}_{t,n}^s - \mathbf{x}_t\|_2^2 \neq 0$ which is a rather reasonable assumption. It is not difficult to see that the implicit function $\mathbf{c}_{t,n}^s(\mathbf{B}_n)$ is only continuous and differentiable at its only nonzero point. We set the gradient $\partial(\hat{\mathbf{c}}_{t,n}^s)_\Lambda / \partial(\mathbf{b}_{n,i})_m = \mathbf{0}$ to avoid having unstable

update for inactive candidate atoms, and only the active candidate atoms are updated, where $(\mathbf{b}_{n,i})_m \in \mathbb{R}$ is the m^{th} element of $\mathbf{b}_{n,i}$. Applying differentiation on both sides of (5.12) at the nonzero points of the matching coefficient:

$$\frac{\partial \mathbf{b}_{n,j}^\top \left(\mathbf{b}_{n,j}(\mathbf{c}_{t,n}^s)_j - \mathbf{x}_t\right)}{\partial (\mathbf{b}_{n,j})_m} = 0, \tag{5.13}$$

where $(\mathbf{c}_{t,n}^s)_j$ is the j^{th} element of $\mathbf{c}_{t,n}^s$. Expand the above equation yields:

$$\frac{\partial \mathbf{b}_{n,j}^\top \mathbf{b}_{n,j}}{\partial (\mathbf{b}_{n,j})_m}(\mathbf{c}_{t,n}^s)_j + \mathbf{b}_{n,j}^\top \mathbf{b}_{n,j} \frac{\partial (\mathbf{c}_{t,n}^s)_j}{\partial (\mathbf{b}_{n,j})_m} - \frac{\partial \mathbf{b}_{n,j}^\top \mathbf{x}_t}{\partial (\mathbf{b}_{n,j})_m} = 0. \tag{5.14}$$

Therefore, we reach the desired gradient:

$$\frac{\partial (\mathbf{c}_{t,n}^s)_j}{\partial (\mathbf{b}_{n,j})_m} = (\mathbf{b}_{n,j}^\top \mathbf{b}_{n,j})^{-1} \left(\frac{\partial \mathbf{b}_{n,j}^\top \mathbf{x}_t}{\partial (\mathbf{b}_{n,j})_m} - \frac{\partial \mathbf{b}_{n,j}^\top \mathbf{b}_{n,j}}{\partial (\mathbf{b}_{n,j})_m}(\mathbf{c}_{t,n}^s)_j \right). \tag{5.15}$$

After combining (5.15) with (5.11) for each element of candidiate dictionary atoms, the gradient for updating the dictionary can be achieved. For large datasets, the dictionary and the classifier parameters are updated in an online manner.

5.4 Experimental Verification

In this section, we evaluate our proposed invariant sparse coding algorithm on three publicly available datasets, including the MNIST, CIFAR-10 and STL-10. The classifica-

tion is performed using linear SVM with LIBSVM toolbox [99]. The step size used in the unsupervised dictionary learning is based on the classical neural network method:

$$\rho^t = \frac{\rho_0}{\sqrt{(Bt)/K + 1}}, \tag{5.16}$$

where t is the iteration counter, ρ_0 is the initial step size and B is the batch size which is set to 2048 in our experiments. Parameters are chosen with a 5-fold cross-validation. We start searching the step size from 10^{-2} and increase the step size by a factor of 10 until the performance on the validation set decreases. The regularization parameter λ is optimized over the set $\{10^{-5}, 10^{-4}, \ldots, 10^{-1}, 10^{0}\}$. The searching window size is chosen from the set $\{2 \times 2, 3 \times 3, 4 \times 4\}$. Patch size for the first layer is in the set $\{3 \times 3, 4 \times 4, \ldots, 8 \times 8\}$. Using a larger number of BoA in this chapter is chosen from the set $\{100, 200, 400, 800\}$. Larger number of BoA may produce better performance, but is right now beyond our computational capacity. In all the experiments, the dropout rate is set to 0.5.

5.4.1 Evaluation on the MNIST Database

The MNIST dataset [82] consists of a total number 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.

Before evaluating the object recognition performance, we first examine the patterns of the atoms within the BoAs obtained by the unsupervised dictionary learning in order to help

Method	**Error (%)**
Yang *et. al.* [63]	0.84
CNN [92]	0.53
CKN-PM1	0.63
CKN-PM2	0.53
Proposed - Single Layer	0.66
Proposed - Two Layers	**0.51**

Table 5.1: MNIST Classification Error.

us understand how the proposed method produces the invariant sparse codes. We randomly select $10,000$ samples from the training set for the dictionary learning. All atoms in BoAs are initialized with Gaussian random samples and the patch size is set to 12×12. The number of atoms in each BoA is set to $P = 5$ and the searching window size is set to 4×4. The total number of BoA is set to $N = 96$, i.e., each adaptive dictionary has 128 atoms. We show all the learned BoAs in Fig. 5.3. It is obvious that the atoms in the same BoA are highly similar and most of them are slightly translated or rotated versions of each other, which clearly shows that the proposed sparse coding model can achieve invariant property towards a number of image variations. On the other hand, this similarity pattern property also demonstrates that the learned BoAs are highly redundant, which makes it possible to construct redundant dedicated adaptive dictionaries $\{\mathbf{D}_t^s\}_{t\in\mathcal{N}_s}$ in order to produce stable sparse codes.

The performance of MNIST classification is shown in Table 5.1. The dictionary is trained using the raw input images without any preprocessing. The number of BoA for the single layer architecture is set to 200. For the multi-layer architecture, the number

of layers is 2, the number of BoA is set to be 200 and 400 for the first and the second layer, respectively. The number of atoms in each bag is set to 9 for every layer. The patch size for the first layer is 8×8. We use max pooling over 2×2 regions with stride 2 for the multi-layer architecture. The proposed single layer invariant sparse coding reaches an error rate of 0.66%, which gains 21.43% improvement over the supervised sparse coding-based method [63]. With a two-layer architecture, the proposed method further reduced the classification error to 0.51%, which is competitive comparing to the state-of-art supervised CNN [92]. This performance is also slightly better than the convolutional kernel networks.

5.4.2 Evaluation on the CIFAR-10 Database

The CIFAR-10 dataset consists of a total number of $60,000$ color images that belongs to 10 classes. The database is split into $50,000$ training samples and $10,000$ test samples. Each class has 5000 training images and 1000 testing images, the size of each image is 32×32. Comparing with MNIST, the variance of this image data is significantly larger, which makes it a more difficult classification task. On the other hand, the local patch descriptors extracted from CIFAR-10 are highly correlated, which can lead to the production of unstable matching coefficients and sparse codes.

We perform 5-fold cross-valition on $10,000$ samples of the training set. All results are reported without using data augmentation. The patch size is 6×6 for constructing the first layer. The local patch feature descriptors are extracted from the whitened image patches. The number of BoA is set to 800 for the single layer model. For multi-layer model, the

Method	**Accuracy (%)**
CKN-PM	78.30
Lin *et. al.* [100]	80.90
Coates *et. al.* [75]	81.50
Coates *et. al.* [101]	82.00
Sohn *et. al.* [94]	82.20
Proposed - Single Layer	80.53
Proposed - Two Layers	**82.27**

Table 5.2: CIFAR-10 Classification Accuracy.

first and second layer has a number of 200 and 800 BoAs, respectively. Number of atoms in each BoA is 9. The searching window size for the first and second layers are set to 3×3 and 4×4. Max pooling over 3×3 with stride 2 is applied for the multi-layer model.

We report the performance in Table 6.2 in comparison with several state-of-art deep learning-based algorithms. The proposed multi-layer architecture achieves 82.27% classification accuracy, which is almost 4% higher than the performance of convolutional kernel networks. Our performance on the CIFAR-10 is also competitive with other state-of-art methods, which usually use significantly more filters (dictionaries) as well as data augmentation, such as the ones used in [94, 101, 102]. In the work of [75, 101], a total number of 1600 or 4000 filters are used for sparse coding, which is twice the number used for our proposed model. Even more features are used in the work of [100], where 3200 and 6400 number of dictionary atoms are used for sparse recovery. Therefore, our algorithm is highly efficient in terms of exploiting the dictionary expressiveness.

Method	**Accuracy (%)**
CKN-PM	60.25
Lin *et. al.* - 1 Layer [100]	59.00
Lin *et. al.* - 2 Layers [100]	**60.40**
Sohn *et. al.* [94]	58.70
Proposed - 1 Layer	58.03
Proposed - 2 Layers	58.96

Table 5.3: STL-10 Classification Accuracy.

5.4.3 Evaluation on the STL-10 Database

The dataset STL-10 is similar to CIFAR-10, but the number of labeled training samples are even more limited. With a total number of 10 classes, each class in STL-10 has 500 labeled training images and 800 test images, where each image has a size of 96×96 pixels. Due to its relatively large image size, we have downsampled the images to 32×32 although most of the compared algorithms are evaluated on the original 96×96 images. This dataset is more challenging than CIFAR-10 due to the small number of labeled training samples.

The patches extracted from the dataset are preprocessed with zero-phase whitening. The patch size is set to 6×6. The searching window size is 3×3 and 5×5 for the first and second layers, respectively. The number of BoA is set to 800 for the single layer architecture. For the two-layer architecture, the number of BoA for the first and the second layers are set to 100 and 800, respectively. Max pooling is processed over 3×3 region with stride 2.

The performance of the proposed invariant sparse coding algorithm on STL-10 is shown in Table 5.3. Our method achieves classification accuracy of 58.03% and 58.96% for the

one-layer and two-layers architecture, respectively. The standard deviation of our method is smaller than 0.6%. Our proposed algorithm demonstrates a performance better than the invariant neural network [94]. The OMP-based method [100] gives a state-of-art performance with an accuracy of 60.40% using a two-layer architecture. Our performance is competitive while using a significantly smaller number of features. The number of dictionary atoms N used in the work [100] is set to 3200 and 6400 for the first and second layers, respectively, while we only use a dictionary with at most 800 atoms for sparse recovery. On the other hand, CKN-PM is performed on the original image dataset without downsampling. The major obstacle for further improvement in performance of our algorithm is that we are not able to train model with enough number of BoAs or with large image size due to the intense computation complexity of sparse recovery. In the future work, we will try to exploit some more computationally efficient coding scheme to substitute the labored sparse recovery.

5.5 Summary

In this chapter, we have proposed a novel invariant sparse coding algorithm that is able to build adaptive dictionaries for each input sample, which shares a similar formulation with the archetypal analysis. By capturing the image misalignment through the matching coefficients, the sparse codes are set free from the variabilities in the local features. We have also extended the invariant sparse coding model to a multi-layer architecture in

order to produce sparse codes that are progressively invariant. Different from the CNN-based sparse coding architectures, the proposed invariant sparse coding framework is able to achieve inherently invariant sparse codes even without using max-pooling operations. Our proposed algorithm is also advantageous to CNN when dealing with small training set due to a stronger generality towards unseen samples. In addition, using the fixed point differentiation, a layer-wise unsupervised dictionary learning algorithm is developed for the proposed invariant sparse coding framework, which is able to simultaneously reduce the reconstruction errors of both the sparse recovery and the local feature descriptor matching. Experiments on the MNIST, CIFAR-10 and STL-10 datasets show that the performance of the proposed method is competitive with the state-of-art methods. We have discovered that the learned atoms in each BoA display similar patterns even if we do not explicitly enforce the similarity constraints over these atoms.

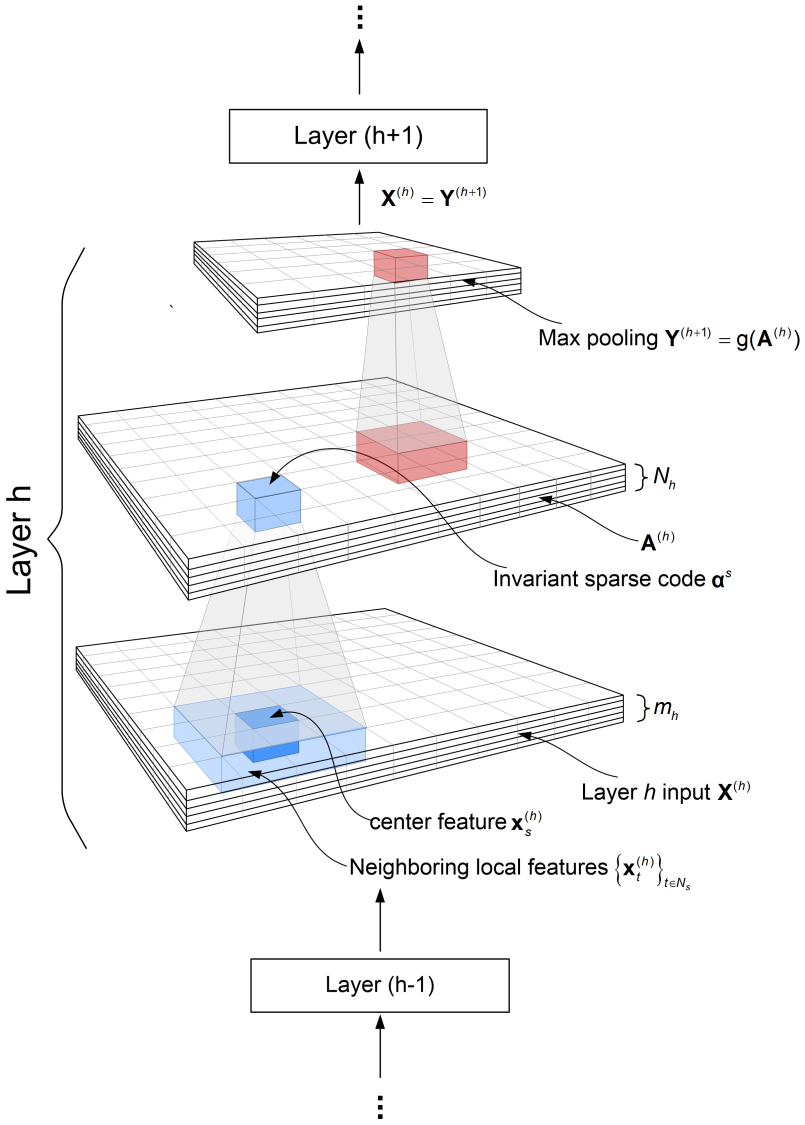


Figure 5.2: Multi-layer invariant sparse coding architecture: The input of layer h is $\mathbf{X}^{(h)}$. Invariant sparse code for every location s is computed by solving problem (5.8), producing a sparse output $\mathbf{A}^{(h)}$, which can be used directly as the layer output or further processed through a max pooling layer.

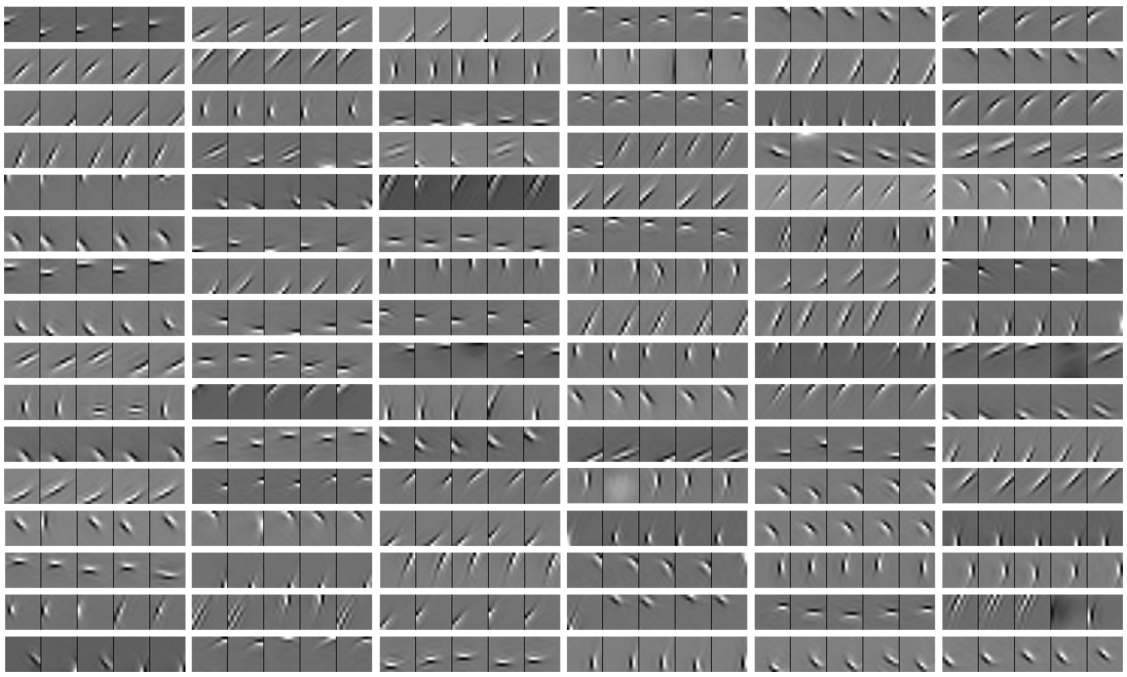


Figure 5.3: Visualization of BoAs learned from the MNIST dataset. Number of BoA is set to 96, each BoA contains 5 candidate atoms. The searching window size is set to 4×4. The training patch size is 12×12. All training patches have been whitened. We can see that each BoA contains atoms with similar patterns in that the atoms in the same BoA are slightly translated or rotated versions of each other.

Chapter 6

Supervised Multilayer Sparse Coding Networks

In this chapter, 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.

6.1 Introduction

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 [63, 72]. 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 [3, 11, 103, 104, 105]. This feat has encouraged researchers to try and augment the learning capacity of traditionally shallow sparse coding methods by adding layers [100, 106, 107]. Theoret-

ically, 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 [108] ameliorates information degeneracy issues within deep architectures [109, 110]. 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 [100, 107].

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 6.2, there has been inspiring breakthroughs in adapting supervised dictionary learning algorithms for use in shallow sparse coding frameworks [59, 63], 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 sig-

nal 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 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 [111] and SqueezeNet [112] 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 chapter 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 ℓ_1 regularization parameters with *task-driven regularization*.
- Supervised, end-to-end training of a multilayer sparse coding network with the aforementioned features.

In Section 6.2, we briefly review the works related to supervised dictionary learning and adaptive regularization. In Section 6.3, 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 6.4, 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 [103] in terms of both classification accuracy and convergence rate.

6.2 Supervised Learning and Adaptive Regularization

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 [113] 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 [106] 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.* [66] propose to directly compute the gradient of the dictionary by switching the ℓ_1 regularizor with the smoothed Kullback-Leibler divergence. More thorough study on task-driven dictionary learning algorithms with various applications are carried out in [59]. Applying fixed point differntiation and error backpropagation, a supervised dictionary learning scheme for the shallow sparse coding model is proposed in [63].

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 [114] and has been proved to satisfy the oracle property [115]. Do *et. al.* [116] 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 [117] 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 [118]. 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.

6.2.1 Multilayer Architecture

To begin, we formulate a generalized, multilayer sparse coding architecture as illustrated Fig. 6.1. Let an image with $H \times W$ pixels and C channels be represented by a 3-dimensional tensor $\mathcal{X} \in \mathbb{R}^{H \times W \times C}$. Denote a single C-channel pixel as $x_i \in \mathbb{R}^C$, where $i \in [HW]$[1] is the linear index of the pixel. We define the *hyperpixel* $\mathbf{x}_i \in \mathbb{R}^M$ as the concatenation of neighboring pixels of x_i within a $K \times K$ receptive field (a patch of neighboring pixels) such that $\mathbf{x}_i = [y_i^\top, \ldots, x_{i+K^2}^\top]^\top$ and $M = CK^2$. We denote the sparse coding as a nonlinear function $f : \mathbb{R}^M \to \mathbb{R}^N$ such that the sparse code at the location i can be recovered as

$$\boldsymbol{\alpha}_i^* = f(\mathbf{x}_i, \Theta), \tag{6.1}$$

where Θ represents the parameters for a given sparse coding layer.

The sparse code $\boldsymbol{\alpha}^* \in \mathbb{R}^N$ is generally of much higher dimension than the input signal. Thus, if output sparse codes are naively and repeatedly fed into successive sparse

[1] $[N]$ denotes the set of nonnegative integers no larger than N.

coding layers, computational complexity quickly explodes. Inspired by Network in Network [111] and SqueezeNet [112], 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. 6.1a. 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.

6.3 Multilayer Sparse Coding networks

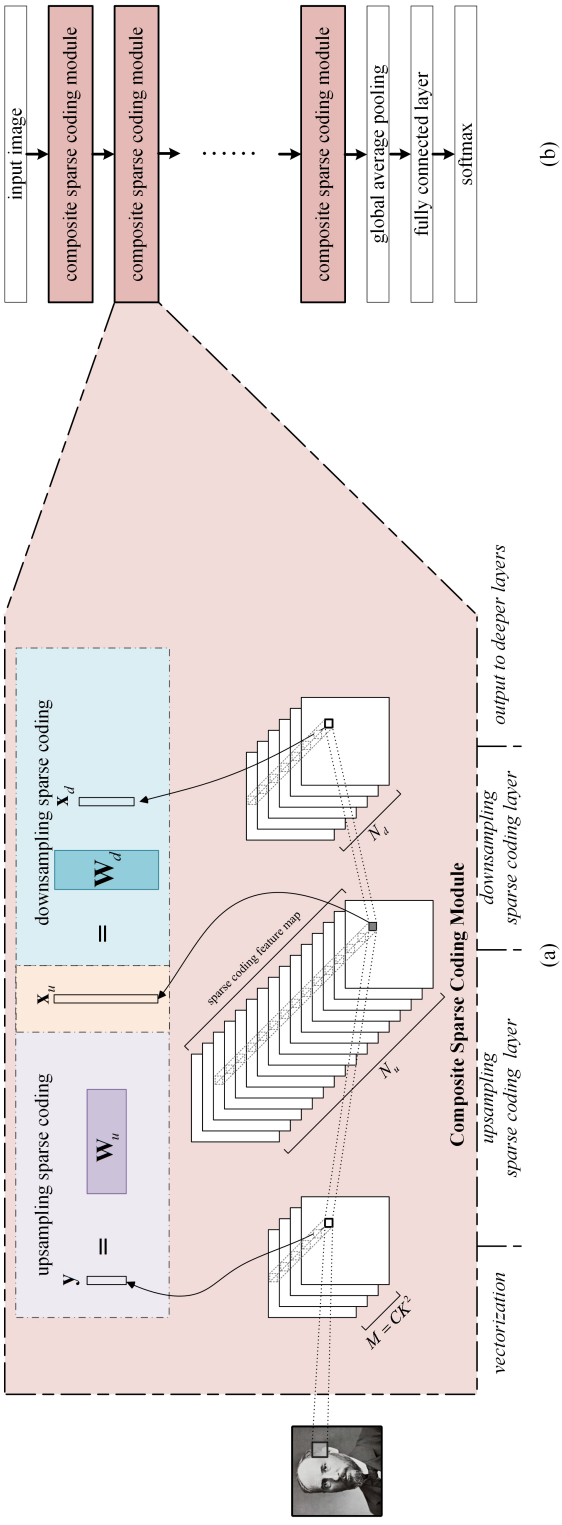


Figure 6.1: Architecture of our multilayer sparse coding network: (a) Composite sparse coding module consists of vectorization operation on a patch of pixels for generating a hyperpixel and followed by consecutively stacking upsampling and downsampling sparse coding layers. In our experiment, vectorization 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.

Our multilayer sparse coding network is constructed by the repeated stacking of our composite sparse coding modules, as depicted in Fig. 6.1b. 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

$$\boldsymbol{\alpha}^* = f(f(\mathbf{x}, \Theta_u), \Theta_d), \tag{6.2}$$

where we have dropped the subscript indices for simplicity. Θ_u, Θ_d are the parameter sets of the upsampling and downsampling sparse coding layers, respectively. In this chapter, 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.

6.3.1 Weighted Nonnegative Sparse Coding

Sharing a formulation similar to adaptive Lasso [114], our loss function is designed to increase the efficiency of the proposed multilayer sparse coding network. Formally, given

an input signal $\mathbf{x} \in \mathbb{R}^M$ and a dictionary $\mathbf{D} \in \mathbb{R}^{M \times N}$ with N atoms and M measurements, we would like to represent the local feature $\mathbf{x}$ with a sparse signal $\boldsymbol{\alpha} \in \mathbb{R}^N$ by solving the following problem

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha} > \mathbf{0}} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \sum_{i=1}^{N} |\lambda_{1i} x_i| + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}\|_2^2, \tag{6.3}$$

where $\{\lambda_{1i}\}_{i=1}^N, \lambda_2$ are the regularization parameters, $\lambda_{1i} \neq 0, \forall i \in [N]$ and $\lambda_2 > 0$. Eq. (6.3) can be simplified as

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha} > \mathbf{0}} \frac{1}{2}\|\mathbf{y} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \|\boldsymbol{\Gamma}\boldsymbol{\alpha}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}\|_2^2, \tag{6.4}$$

by denoting $\boldsymbol{\Gamma} = diag(\lambda_{11}, \ldots, \lambda_{1N}) \in \mathbb{R}^{N \times N}$.

There are two major differences between Eq. (6.4) and the adaptive Lasso models previously reported in [114, 119]. First, in the case of the adaptive Lasso, the initial regularization parameter is set with a nonzero estimation. In our case, the adaptive ℓ_1 regularization parameters need not start from a nonzero point since we train these parameters with back-propagation. The nonnegative constraint on the sparse code prevents our network from getting trapped into a linear system. Second, the ℓ_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 chapter, nonnegativity is enforced upon sparse codes to promote stability and efficiency during sparse recovery [100, 120, 121, 122]. At the outset of training, the near-

zero initialized ℓ_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 (6.4) using the algorithm of alternative direction method of multipliers (ADMM) [67]. For clarity, we describe the ADMM algorithm for our multilayer sparse coding network in Algorithm 4. 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 [43]. The parameter ρ in Algorithm 4 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 ℓ_2-norm.

Algorithm 4 ADMM for multilayer sparse coding network

Require: Dictionary $\mathbf{D} \in \mathbb{R}^{M \times N}$, input feature $\mathbf{x} \in \mathbb{R}^M$, regularization parameters $\mathbf{\Gamma}$,λ_2, $\rho = \rho_0 \|A_i\|_2^2$, $\rho_0 = 0.1$, precomputed $\mathbf{C}^{-1} = (\mathbf{D}^\top \mathbf{D} + \rho \mathbf{I}_N + |\mathbf{\Gamma}|)^{-1}$, $\mathbf{B} = \mathbf{C}^{-1}\mathbf{A}^\top \mathbf{y}$, $\mathbf{u}, \mathbf{z} = \mathbf{0} \in \mathbb{R}^N$.

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

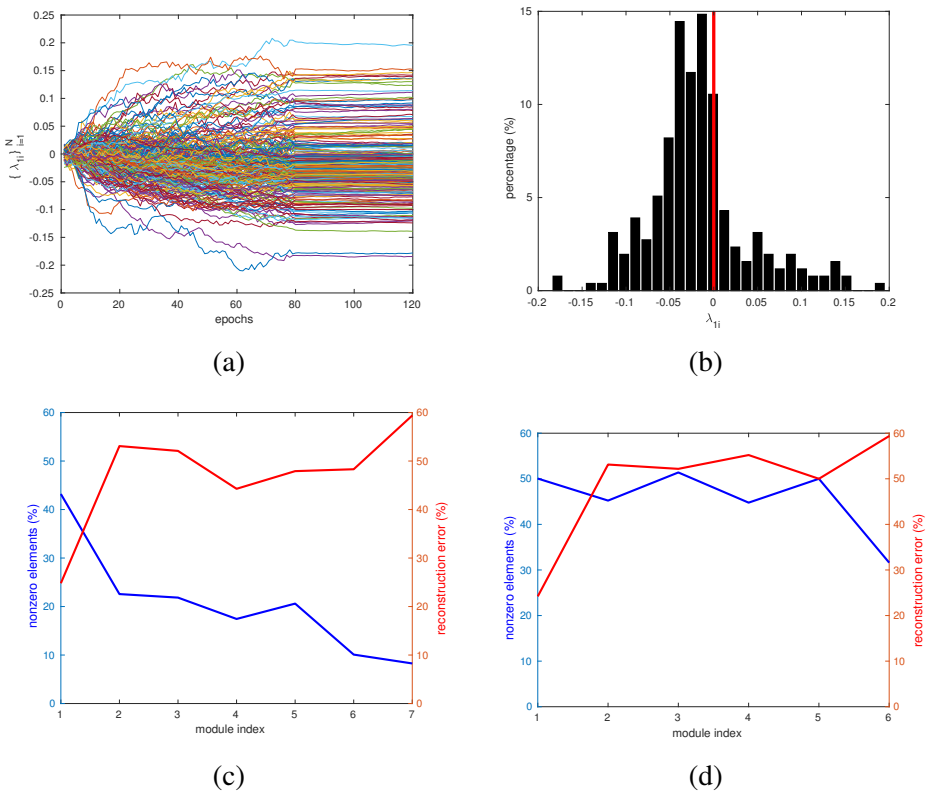


Figure 6.2: (a) - (b): Evolution and distribution of the regularization parameters, respectively. The parameters are extracted from the last sparse coding layer. (c)-(d): Evaluation of the behavior of upsampling and downsampling layer, respectively. The blue and red lines indicate the nonzero elements and reconstruction error in percentage, respectively. Layer index specified by the module index.

6.3.2 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 $\mathbf{\Gamma}$ with some small value to avoid numerical issues (set to be 10^{-5} in this chapter) 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

$$\lambda_{1i} \leftarrow \lambda_{1i} - \eta \frac{\partial L}{\partial \boldsymbol{\alpha}^*} \frac{\partial \boldsymbol{\alpha}^*}{\partial \lambda_{1i}}, \text{ s.t. } \lambda_{1i} \neq 0, \tag{6.5}$$

where $\eta > 0$ is the learning rate, L is the total task-driven loss function defined in Eq. (6.6). The detailed updating rule for regularization parameters will be discussed in the next section. As we shall see in the experiment, Eq. (8.25) causes the regularization parameters to adjust during training in order to render sparse outputs.

6.3.3 Supervised Dictionary Learning

Most of the dictionary learning methods confine the dictionary atoms within a closed unit ℓ_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 ℓ_2-norm, otherwise known as weight decay in neural networks. More specifically, we have

$$L(\Theta) = V(\Theta) + \frac{\mu}{2}\sum_{i=1}^{H}(\|\mathbf{D}^h\|_F^2 + \|\mathbf{\Gamma}^h\|_F^2), \tag{6.6}$$

where $\mu > 0$ is the weight decay, h is the layer index, Θ 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 $\mathbf{x}$, output $\boldsymbol{\alpha}$ 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:

$$\begin{aligned}
\frac{\partial L}{\partial D_{ij}} &= \frac{\partial L}{\partial \boldsymbol{\alpha}} \cdot (\mathbf{D}^\top\mathbf{D} + \lambda_2\mathbf{I})_\Lambda^{-1}(\frac{\partial \mathbf{D}_\Lambda^\top \boldsymbol{\alpha}}{\partial D_{ij}} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial D_{ij}}\mathbf{y}_\Lambda) \\
\frac{\partial L}{\partial x_i} &= \frac{\partial L}{\partial \boldsymbol{\alpha}} \cdot (\mathbf{D}^\top\mathbf{D} + \lambda_2\mathbf{I})_\Lambda^{-1}\frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i} \\
\frac{\partial L}{\partial \lambda_{1j}} &= \frac{\partial L}{\partial \boldsymbol{\alpha}} \cdot -(\mathbf{D}^\top\mathbf{D} + \lambda_2\mathbf{I})_\Lambda^{-1}\text{sign}(\boldsymbol{\alpha}_\Lambda)_j, \text{ s.t. } \lambda_{1j} \neq 0,
\end{aligned} \tag{6.7}$$

where the subscript Λ denotes the active set of the sparse code $\boldsymbol{\alpha}$, $\mathbf{D}_\Lambda$ is composed of the

active columns of $\mathbf{D}$ and $\boldsymbol{\alpha}_\Lambda$ is the active elements of the sparse code. During training, the computation of $(\mathbf{D}^\top\mathbf{D} + \lambda_2\mathbf{I})_\Lambda^{-1}$ 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 $\{\alpha_i : |x_i| > \epsilon, \forall i \in [N]\}$, where ϵ is a small constant value to avoid numerical instability and α_i is the ith element of the sparse code $\boldsymbol{\alpha}$. In multilayer sparse coding networks, a fixed threshold ϵ 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 atoms. To compensate for this effect, we set the threshold as

$$\epsilon = \epsilon_0 \frac{\|\alpha\|_2}{\|D_i\|_2}, \tag{6.8}$$

where i is the index of a dictionary atom with largest ℓ_2 norm and ϵ_0 is the threshold when both dictionary atoms and input signal are ℓ_2-normalized. We set $\epsilon_0 = 10^{-3}$ throughout our chapter.

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

Table 6.1: Network Configuration

6.4 Experimental Verification

We evaluate our multilayer sparse coding network on the benchmark dataset of CIFAR-10, CIFAR-100, SVHN and MNIST. All programs [2] are written in MATLAB with C++ and CUDA based on the MatConvNet framework [123]. 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 6.1. For the CNN baseline, each convolutional layer is followed by a ReLU layer [14], which is omitted in the table. The configuration of the network architecture is inspired by the residual network [103]. The network structure consists of two key features. First, there are no maxpooling layers. The spatial subsampling operation is fulfilled by specific

[2]Codes used in this chapter will become publicly available soon.

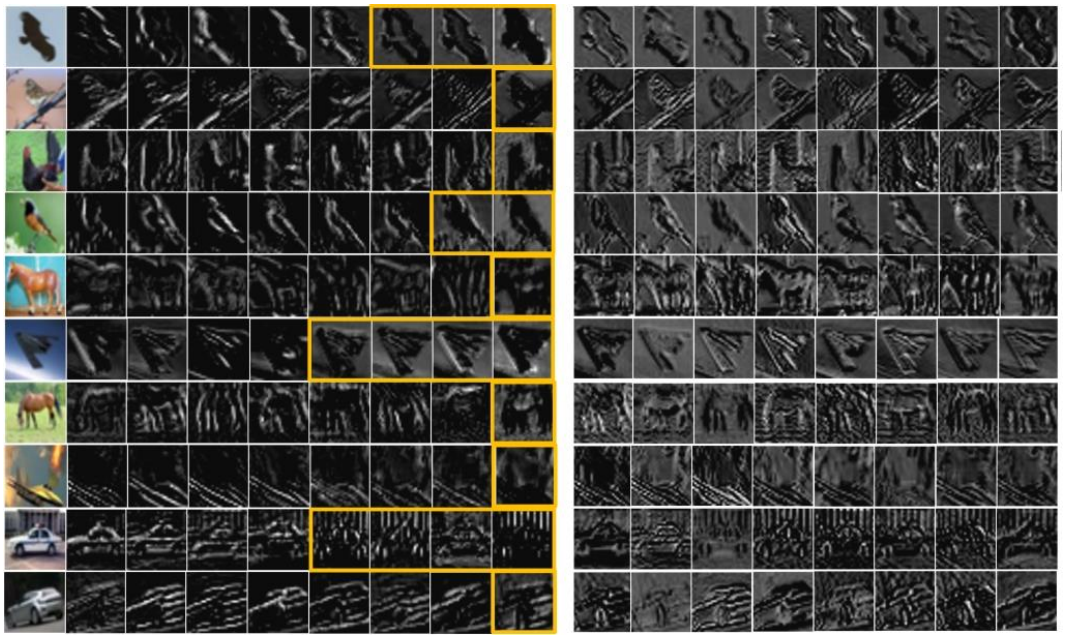


Figure 6.3: Visualization of feature map: From left to right: Original image; feature maps of our sparse coding network - feature maps contain mostly background are labeled with yellow rectangles; and feature map of the baseline CNN.

sparse coding layers with a stride of 2, denoted as '$/2$'. 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 [103, 124]. Our sparse coding network consists of six composite sparse coding modules with a total number of 13 sparse coding layers, 1 global spatial average pooling layer [111] and 1 fully connected layer as shown in Table 6.1. 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.

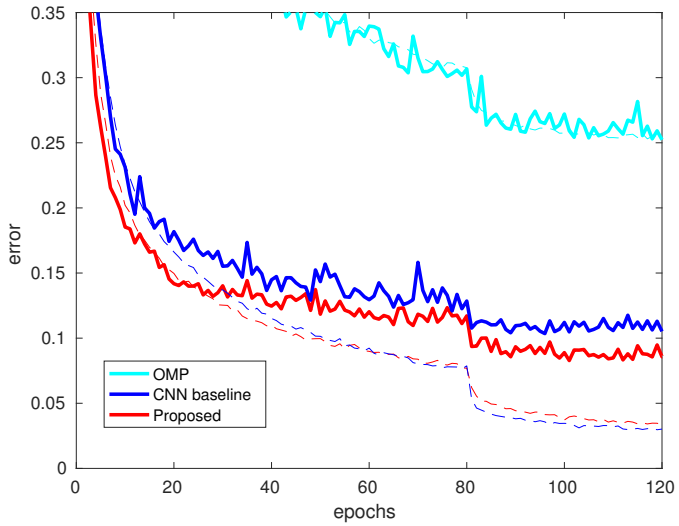


Figure 6.4: Performance comparison with the CNN baseline on CIFAR-10. Dashed and bold lines denote training and testing error, respectively.

6.4.1 Evaluation on CIFAR-10 Database

Our most extensive experiment is conducted on the CIFAR-10 dataset [125], 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.

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 [75, 103, 111, 126]. 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 not guarantee that our multilayer sparse coding network or the baseline CNN can reach its best performance.

In Fig. 6.3, we display the feature maps of both the sparse coding network and the baseline CNN, which are produced by the output of the 5^{th} 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 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.

6.4.1.1 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. 6.2.

Optimization of regularization parameters: The evolution of the regularization parameter with respect to epochs is shown in Fig. 6.2a. 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. 6.2b, 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. 6.2c, less than 10% of the output elements of the last sparse coding layer are nonzero.

Upsampling layer: Illustrated in Fig. 6.2c, 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. 6.2d. Except for the last downsampling layer that reaches a sparsity level of 20%, all others have $40\% - 50\%$ nonzero sparse coefficients.

6.4.1.2 Fast Convergence with Task-driven Regularization

We now demonstrate the advantage of using weighted Lasso over the heuristic ℓ_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. 6.4. 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.

6.4.1.3 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 [100] and nonnegative-OMP based models [75], by a margin of 10%.

[3] Including 13 sparse coding layers and 1 fully connected layer.

Method	# params	# layers	Accuracy (%)
Maxout [75]	-	-	90.62
SCKN [127]	10.50M	10	90.80
NIN [111]	-	-	91.19
DSN [126]	1.34M	7	92.03
ResNet [103]	0.27M	20	91.75
OMP [75]	0.70M	2	81.50
PCANet [1]	0.28B	3	78.67
NOMP [100]	1.09B	4	81.40
CNN-baseline	0.27M	14	88.56
Proposed	**0.27M**	**14**	**91.43**

Table 6.2: CIFAR-10 Classification Accuracy.

6.4.2 Evaluation on CIFAR-100 Database

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 6.3. Comparison of the convergence rate with our CNN baseline is shown in Fig. 6.5. 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

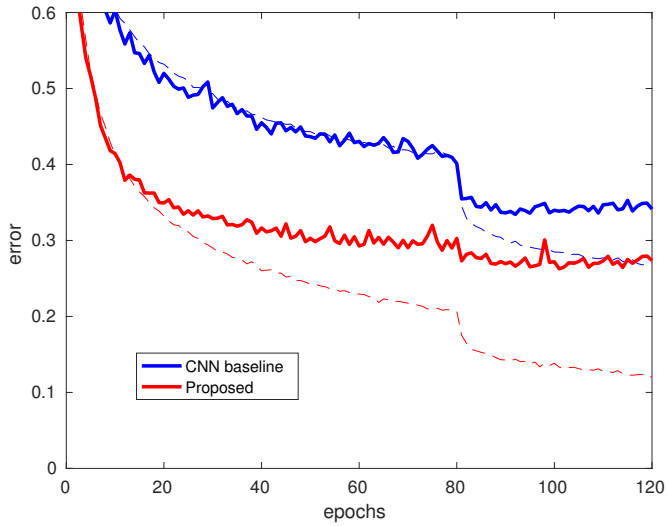


Figure 6.5: Performance comparison with the CNN baseline on CIFAR-100. Dashed and bold lines denote training and testing error, respectively.

coding network when dealing with a relatively small number of training samples per class.

Method	**# params**	**# layers**	**Accuracy (%)**
NOMP [100]	1.09B	4	60.08
Maxout [75]	-	-	63.46
NIN [111]	-	-	64.32
DSN [126]	1.34M	7	63.46
All-CNN [124]	1.40M	10	66.29
Highway [128]	2.3M	19	67.76
ResNet [129]	0.46M	32	68.10
CNN-baseline	0.27M	14	67.58
Proposed	**0.27M**	**14**	**72.64**

Table 6.3: CIFAR-100 Classification Accuracy.

6.4.3 Evaluation on SVHN Database

SVHN [130] 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 6.4. 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 6.4. 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.

Method	**# params**	**# layers**	**Error (%)**
RCNN [12]	2.67M	-	1.77
ReNet [13]	23.12M	7	2.38
DSN [126]	1.34M	7	1.92
Maxout [75]	-	-	2.37
NIN [111]	-	-	2.35
CNN-baseline	0.27M	14	2.95
Proposed	**0.27M**	**14**	**2.16**

Table 6.4: SVHN Classification Error.

6.4.4 Evaluation on MNIST Database

The MNIST [82] 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 6.5. 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%.

Method	**# params**	**# layers**	**Error (%)**
ScatNet [89]	-	3	0.43
PCANet [1]	-	3	0.62
Maxout [75]	-	-	0.45
NIN [111]	-	-	0.47
CKN	0.44M	3	0.39
DSN [126]	0.35M	3	0.39
Highway [128]	0.15M	20	0.45
ResNet [103, 131]	-	100	0.51
CNN-baseline	0.27M	14	0.47
Proposed	**0.27M**	**14**	**0.39**

Table 6.5: MNIST Classification Error.

6.5 Summary

In this chapter, 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.

Chapter 7

Conclusion and Future Research

7.1 Conclusion

In this thesis, we have exploited and evaluated four key components in sparse coding with application on image classification in order to: i) enforce various structured sparsity priors for achieving more stable sparse codes; ii) train the dictionary using task-driven dictionary learning in order to improve the coding efficiency; iii) enforce the invariant property for gaining more generality toward the variability, and iv) extend the sparse coding model to a multilayer architecture so that its learning capacity can be substantially augmented.

In Chapter 2 and Chapter 3 we have exploited various structured sparsity priors and developed supervised dictionary learning algorithms for efficient representation. We propose a new dictionary learning algorithm for task-driven dictionary learning with joint or Laplacian sparsity in order to exploit the spatial-spectral information of HSI neighboring pixels.

We show experimentally that the proposed dictionary learning methods have a significantly better performance than SRC even when the dictionary is highly compact. We also describe an optimization algorithm for solving the Laplacian sparsity recovery problem. The proposed optimization method is much faster than the modified feature sign search used in [42].

We present a novel sparse coding framework in Chapter 4 that is robust to image transformation. In the proposed model, each dictionary atom is constructed in the form of a tensor and is aligned with the test image using the large displacement optical flow concept [79]. The proposed algorithm does not require the training dataset to be pre-aligned. Adapting the dictionary to the input test image is highly efficient: requiring only $O(PT)$ operations for adapting each dictionary atom, where T is the number of pixels in a searching window and P is the total number of subatoms to be aligned. Supervised dictionary learning algorithm is developed for the proposed sparse coding framework.

Extending invariant sparse coding model to multilayer architecture is discussed in Chapter 5 and Chapter 6. We first develop a layer-wise unsupervised dictionary learning algorithm with bilevel optimization in order to simultaneously minimize the errors of local feature descriptor matching and signal reconstruction. Using our layer-wise unsupervised dictionary learning algorithm, we are able to design diverse and contextually rich BoAs. We have extended the sparse coding model to a multilayer architecture with as many as 13 sparse coding layers. 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.

7.2 Future Research

Our sparse coding networks confront with a scalability issue due to the high computational demand of the sparse recovery and dictionary learning. We intend to address the scalability issue in the our future work, which is discussed in more details as follows.

7.2.1 Integrating Sparse Coding Networks with Convolutional Neural Networks

One should not ignore either the benefit of applying convolutional layer or sparse coding layer. On one hand, convolutional layer is highly computationally efficient though requiring more samples to regularize the learnable parameters. On the other hand, sparse coding layer demands much fewer training samples at the sacrifice of high computational cost for recovering the sparse codes. For a given deep architecture, the shallower layer comes with a smaller receptive field and the local features in the corresponding layer have much smaller variations. Therefore, the shallower layers require few labeled training samples to avoid overfitting. Hence, a deep network that has convolutional layers in the shallower part and sparse coding layers in the deeper part should be able to enjoy both the low computational cost and less severe data hunger issue.

In addition, unlike our previous work that uses sparse coding layer with 1×1 kernels for dimension reduction, we propose to use the convolutional layer with 3×3 kernels to reduce the dimension of the input sparse code. The motivations are 1) The dimension reduction layer contains only few number of learnable parameters and therefore these parameters should be able to be regularized by the data itself. 2) The output of the convolutional layer is highly sparse and also nonnegative, i.e., the input for each sparse coding layer is highly sparse and nonnegative. Since the dictionary atoms and the input feature lie in the same subspace, we should be able to naturally enforce nonnegativity constraint on the dictionary. Such nonnegativity regularization on the dictionary atoms can further allow us to use the theorems related with nonnegative matrix factorization (NMF) to explain the behavior of the sparse coding layer.

7.2.2 Fast Approximation of Sparse Coding

The success of multilayer sparse coding network has reaffirmed that multilayer architecture can escape from degeneracy issues by resorting to a generative model. In our deep coding network, both sparse coding and sparse clustering are critical for a stabilizing multilayer system. However, there is one major issue that frustrates our sparse coding network, namely the high computation cost as well as memory requirement. One natural question is how to design highly scalable deep coding network that can largely reduce computational demand by stop chasing unnecessary recovery accuracy while encouraging clustering. As validated in neural network, network with binary activation is able to produce reasonable

performance on relatively shallow networks. Such discovery confirms that the sparse-code support is already accurate enough if we directly use the activated neurons. Again, maybe such support is already accurate enough for clustering purposes.

Denote $\mathbf{D}$ as the weights for the neural network or the deep coding dictionary, $\mathbf{b}$ be the bias, $\mathbf{x}$ as the signal observation and $\boldsymbol{\alpha}$ as the desired sparse code. For illustration purpose, we only consider single signal case and the convolutional case can be naturally extended. We select the support of the sparse codes using ReLU activation function $\Lambda = logical(\text{relu}(\mathbf{D}(\mathbf{x}+\mathbf{b})))$ where $logical(\cdot)$ is the binarization operation and Λ is the active set. This clever approach allows us to compute the gradient of the loss function w.r.t. the bias using backpropagation. To reconstruct the magnitude of the sparse code $\boldsymbol{\alpha}$, we propose to solve the following nonnegative least square problem:

$$\boldsymbol{\alpha}^*_\Lambda = argmin_{\boldsymbol{\alpha}_\Lambda>0}\|\mathbf{D}_\Lambda\boldsymbol{\alpha}_\Lambda - (\mathbf{y}+\mathbf{b})\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_2^2, \tag{7.1}$$

where $\boldsymbol{\alpha}_\Lambda$ is the active elements of the sparse code, $\lambda > 0$ is the ℓ_2 regularizor used to stabilize the model when the code is not sparse enough. Problem (7.1) can be solved much more efficiently. When we add the bias to the input signal $\mathbf{y}$, the sparse code also tries to recover the bias component, therefore preserving the gradient information. We intend to explore gradient descent techniques to optimize $\mathbf{D}$, $\mathbf{b}$, and $\boldsymbol{\alpha}$ alternatingly.

7.2.3 Construction of Wide Sparse Coding Network

Wide neural network, where each layer contains large number of filters, have shown its effectiveness in reducing the required number of layers, hence significantly increase the computational efficiency of the network. Unfortunately, previous works have shown that extending the network to a wider architecture is not trivial. As the network becomes wider, the overfitting issue become more and more severe.

Sparse coding layer is a perfect building block for such wide network: First, the dictionary for sparse recovery is naturally designed to have a wide architecture so as to gain the redundancy and robustness. In most cases, the number of dictionary atoms is $4 - 6$ times larger than the number of the measurement, rendering a network with extremely wide structure. Second, sparsity regularizor acts as a strong prior that is able to largely alleviate the overfitting issue when the network becomes extremely wide. Last but not least, as the network goes wider, the dictionary atoms naturally becomes sparser as we have already discussed in Section 7.2.1. In the future, we will investigate how to efficiently employ highly redundant dictionary in the sparse coding network.

Chapter 8

Appendix

8.1 Appendix A

We can infer from Eq. (3.18) that $vec(\partial \mathbf{A}/\partial D_{mn}) = \mathbf{0}$, $\forall n \in \Lambda^c$, which indicates $\partial \mathcal{L}/\partial D_{mn} = \mathbf{0}, \forall n \in \Lambda^c$. Therefore, we only need to take the gradient $\partial \mathcal{L}/\partial D_{mn}, \forall n \in \Lambda$ into account.

From the Eq. (3.13) and Eq. (3.18), we achieve the gradient for every element of $\tilde{\mathbf{D}}$,

$$\frac{\partial \mathcal{L}}{\partial \tilde{D}_{mn}} = vec\left(\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{A}}}\right)^{\top} \cdot vec\left(\frac{\partial \tilde{\mathbf{A}}}{\partial \tilde{D}_{mn}}\right), \tag{8.1}$$

where $m = 1, \ldots, M$ and $n = 1, \ldots, N_\Lambda$. Let $\mathbf{g} = vec\left(\frac{\partial f}{\partial \tilde{\mathbf{A}}^{\top}}\right) = vec\left(\left(\mathbf{W}\hat{\mathbf{A}} - \hat{\mathbf{Y}}\right)^{\top}\tilde{\mathbf{W}}\right)$ and $\tilde{\mathbf{W}} = \mathbf{W}^{\Lambda}$ is the Λ columns of $\mathbf{W}$. Expand Eq. (3.18) and combine it with Eq. (8.1),

we have

$$\frac{\partial \mathcal{L}}{\partial \tilde{D}_{mn}} = U_{mn} - V_{mn} \text{ and } \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{D}}} = \mathbf{U} - \mathbf{V}, \tag{8.2}$$

where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{m \times N_\Lambda}$ and every element U_{mn}, V_{mn} are defined as

$$\begin{aligned} U_{mn} &= \mathbf{g}^\top \left(\tilde{\mathbf{D}}^\top \tilde{\mathbf{D}} \otimes \mathbf{I}_P + \lambda \boldsymbol{\Gamma} \right)^{-1} vec\left((\mathbf{X} - \mathbf{D}\mathbf{A})^\top \tilde{\mathbf{E}}_{mn} \right), \\ V_{mn} &= \mathbf{g}^\top \left(\tilde{\mathbf{D}}^\top \tilde{\mathbf{D}} \otimes \mathbf{I}_P + \lambda \boldsymbol{\Gamma} \right)^{-1} vec\left(\tilde{\mathbf{A}}^\top \tilde{\mathbf{E}}_{mn}^\top \tilde{\mathbf{D}} \right), \end{aligned}$$

where $\tilde{\mathbf{E}}_{mn} \in \mathbb{R}^{M \times N_\Lambda}$ is the indicator matrix that element (m, n) of $\tilde{\mathbf{E}}_{mn}$ is 1 and all other elements are zero.

Consider the simplification for $\mathbf{U}$ first

$$\begin{aligned} U_{mn} &= \mathbf{g}^\top \left(\tilde{\mathbf{D}}^\top \tilde{\mathbf{D}} \otimes \mathbf{I}_P + \lambda \boldsymbol{\Gamma} \right)^{-1} \left(\tilde{\mathbf{E}}_{mn}^\top \otimes \mathbf{I}_P \right) vec\left((\mathbf{X} - \mathbf{D}\mathbf{A})^\top \right) \\ &= \mathbf{g}^\top \mathbf{F}^{\tilde{n}} vec\left((\mathbf{X} - \mathbf{D}\mathbf{A})^\top \right)_{\tilde{m}}, \end{aligned} \tag{8.3}$$

where $\mathbf{F} = \left(\tilde{\mathbf{D}}^\top \tilde{\mathbf{D}} \otimes \mathbf{I}_P + \lambda \boldsymbol{\Gamma} \right)^{-1}$; $\tilde{m}(m) = \{(m-1)P + 1, \ldots, mP\}$, $\tilde{n}(n) = \{(n-1)P + 1, \ldots, nP\}$ denote the index sets; $\mathbf{F}^{\tilde{n}}$ are the $\tilde{n}$ columns of $\mathbf{F}$.

Let $\boldsymbol{\xi}_m = vec\left((\mathbf{X} - \mathbf{D}\mathbf{A})^\top \right)_{\tilde{m}}$. It can be shown that $\boldsymbol{\xi}_m^\top$ is the m^{th} row of $(\mathbf{X} - \mathbf{D}\mathbf{A})$. Now the (m, n) element U_{mn} of the first part $\mathbf{U}$ can be written as

$$U_{mn} = \left(\mathbf{g}^\top \mathbf{F} \right)^{\tilde{n}} \boldsymbol{\xi}_m, \tag{8.4}$$

Stacking all elements of $\mathbf{U}$

$$\begin{aligned}\mathbf{U} &= \begin{bmatrix} \left(\mathbf{g}^\top \mathbf{F}\right)^{\tilde{1}} \boldsymbol{\xi}_1 & \cdots & \left(\mathbf{g}^\top \mathbf{F}\right)^{\widetilde{N_\Lambda}} \boldsymbol{\xi}_1 \\ \vdots & \ddots & \vdots \\ \left(\mathbf{g}^\top \mathbf{F}\right)^{\tilde{1}} \boldsymbol{\xi}_M & \cdots & \left(\mathbf{g}^\top \mathbf{F}\right)^{\widetilde{N_\Lambda}} \boldsymbol{\xi}_M \end{bmatrix} \\ &= \boldsymbol{\xi} \left[(\mathbf{g}^\top \mathbf{F})^{\tilde{1}^\top} \cdots (\mathbf{g}^\top \mathbf{F})^{\tilde{N_\Lambda}^\top} \right], \end{aligned} \tag{8.5}$$

where Λ_n denotes the n^{th} element of set Λ.

Now consider simplification for $\mathbf{V}$. Each element V_{mn} of $\mathbf{V}$ can be written as

$$\begin{aligned} V_{mn} &= \mathbf{g}^\top \mathbf{F} \cdot vec\left(\tilde{\mathbf{A}}^\top \tilde{\mathbf{E}}_{mn}^\top \tilde{\mathbf{D}}\right) \\ &= \mathbf{g}^\top \mathbf{F} \left(\tilde{\mathbf{D}}^\top \tilde{\mathbf{E}}_{mn} \otimes \mathbf{I}_P\right) vec\left(\tilde{\mathbf{A}}^\top\right) \\ &= \mathbf{g}^\top \mathbf{F} \left(\tilde{\mathbf{D}}_m^\top \otimes \mathbf{I}_P\right) \tilde{\mathbf{A}}_n^\top, \end{aligned} \tag{8.6}$$

where $\tilde{\mathbf{A}}_n$ is the n^{th} row of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{D}}_m$ is the m^{th} row of $\mathbf{D}$.

Stacking every element of $\mathbf{V}$, such that

$$\mathbf{V} = \begin{bmatrix} \mathbf{g}^\top \mathbf{F} \left(\tilde{\mathbf{D}}_1^\top \otimes \mathbf{I}_P\right) & \cdots & \mathbf{g}^\top \mathbf{F} \left(\tilde{\mathbf{D}}_1^\top \otimes \mathbf{I}_P\right) \\ \vdots & \ddots & \vdots \\ \mathbf{g}^\top \mathbf{F} \left(\tilde{\mathbf{D}}_M^\top \otimes \mathbf{I}_P\right) & \cdots & \mathbf{g}^\top \mathbf{F} \left(\tilde{\mathbf{D}}_M^\top \otimes \mathbf{I}_P\right) \end{bmatrix} \mathbf{A}^\top$$

$$= \begin{bmatrix} \sum_{n=1}^{N} \mathbf{D}_{1n}^{\top} \left(\mathbf{g}^{\top}\mathbf{F}\right)_{\bar{n}} \mathbf{A}_{1}^{\top} & \cdots & \sum_{n=1}^{N} \mathbf{D}_{1n}^{\top} \left(\mathbf{g}^{\top}\mathbf{F}\right)_{\bar{n}} \mathbf{A}_{N}^{\top} \\ \vdots & \ddots & \vdots \\ \sum_{n=1}^{N} \mathbf{D}_{Mn}^{\top} \left(\mathbf{g}^{\top}\mathbf{F}\right)_{\bar{n}} \mathbf{A}_{1}^{\top} & \cdots & \sum_{n=1}^{N} \mathbf{D}_{Mn}^{\top} \left(\mathbf{g}^{\top}\mathbf{F}\right)_{\bar{n}} \mathbf{A}_{N}^{\top} \end{bmatrix}$$

$$= \mathbf{D} \left[\left(\mathbf{g}^{\top}\mathbf{F}\right)_{\bar{1}}^{\top} \cdots \left(\mathbf{g}^{\top}\mathbf{F}\right)_{\bar{P}}^{\top} \right] \mathbf{A}^{\top}, \tag{8.7}$$

where $\bar{p}(p) = \{p, p+P, \ldots, p+(N-1)P\}$. Combining Eq. (8.5) and Eq. (8.7)

$$\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{D}}} = \mathbf{U} - \mathbf{V} = \boldsymbol{\xi}\boldsymbol{\beta}_{\Lambda}^{\top} - \tilde{\mathbf{D}}\boldsymbol{\beta}_{\Lambda}\tilde{\mathbf{A}}^{\top} \text{ and } \frac{\partial \mathcal{L}}{\partial \mathbf{D}} = \boldsymbol{\xi}\boldsymbol{\beta}^{\top} - \mathbf{D}\boldsymbol{\beta}\mathbf{A}^{\top}, \tag{8.8}$$

where $\boldsymbol{\beta}_{\Lambda_c} = \mathbf{0}$ and $\boldsymbol{\beta}_{\Lambda} = [\left(\mathbf{g}^{\top}\mathbf{F}\right)_{\tilde{1}}^{\top}, \cdots, \left(\mathbf{g}^{\top}\mathbf{F}\right)_{\widetilde{N_{\Lambda}}}^{\top}]^{\top}$. More generally, we have defined $\boldsymbol{\beta}_{\Lambda} \in \mathbb{R}^{N_{\Lambda} \times P}$ such that $vec(\boldsymbol{\beta}_{\Lambda}^{\top}) = \mathbf{F}\mathbf{g}$.

8.2 Appendix B

The gradient for updating the dictionary can be written as

$$\frac{\partial \mathcal{L}}{\partial D_{mn}} = vec\left(\frac{\partial \mathcal{L}}{\partial \mathbf{A}}\right)^{\top} \cdot vec\left(\frac{\partial \mathbf{A}}{\partial D_{mn}}\right)$$
$$= vec\left(\frac{\partial \mathcal{L}}{\partial \mathbf{A}}\right)_{\Lambda}^{\top} \cdot vec\left(\frac{\partial \mathbf{A}}{\partial D_{mn}}\right)_{\Lambda}, \tag{8.9}$$

Expand Eq. (3.31) and combine it with Eq. (8.9), the desired gradient is

$$\frac{\partial \mathcal{L}}{\partial D_{mn}} = U_{mn} - V_{mn} \text{ and } \frac{\partial \mathcal{L}}{\partial \mathbf{D}} = \mathbf{U} - \mathbf{V}, \tag{8.10}$$

where

$$\begin{aligned} U_{mn} &= \mathbf{g}^\top \mathbf{F}^{-1} vec\left(\mathbf{E}_{mn}^\top (\mathbf{X} - \mathbf{DA})\right)_\Lambda, \\ V_{mn} &= \mathbf{g}^\top \mathbf{F}^{-1} vec\left(\mathbf{D}^\top \mathbf{E}_{mn} \mathbf{A}\right)_\Lambda, \\ \mathbf{F} &= \left(\mathbf{I}_P \otimes \mathbf{D}^\top \mathbf{D} + \gamma \mathbf{L} \otimes \mathbf{I}_N\right)^{-1}_{\Lambda,\Lambda}. \end{aligned}$$

Let $\mathbf{g}$ has the same definition as that in Section 8.1. The first part $\mathbf{U}$ of $\frac{\partial f}{\partial D_{mn}}$ is

$$\begin{aligned} U_{mn} &= \mathbf{g}^\top \mathbf{F} vec\left(\mathbf{E}_{mn}^\top (\mathbf{X} - \mathbf{DA})\right)_\Lambda \\ &= \left(\mathbf{g}^\top \mathbf{F}\right)_{\tilde{n}} vec\left(\mathbf{X} - \mathbf{DA}\right)_{\tilde{m}(m,n)} \end{aligned} \tag{8.11}$$

$\mathbf{E}_{mn} \in \mathbb{R}^{M \times N}$ is the indicator matrix that the (m, n) element is 1 and all other elements are zero. $\tilde{m}$ and $\tilde{n}$ are defined as the following index sets,

$$\begin{aligned} \tilde{m}(m,n) &= \{m, \ldots, m + pM, \ldots\}, \forall p \text{ s.t. } n + pN \in \Lambda \\ \tilde{n}(n) &= \{n, n + N, \ldots, n + (P-1)N\} \cap \Lambda \end{aligned}$$

Eq. (8.11) can be further simplified by introducing $\mathbf{h}^{(n)} \in \mathbb{R}^P$, such that

$$\mathbf{h}^{(n)} = \begin{cases} \left(\mathbf{g}^\top \mathbf{F}\right)_{n+pN}, \text{ if } n + pN \in \tilde{n}(n), \forall p \\ 0, \text{ otherwise} \end{cases} \tag{8.12}$$

Now Eq. (8.11) can be rewritten as,

$$U_{mn} = \mathbf{h}^{(n)\top} \boldsymbol{\xi}_m, \tag{8.13}$$

where $\boldsymbol{\xi}_m^\top$ is the m^{th} row of $\mathbf{X} - \mathbf{DA}$. The first part $\mathbf{U}$ of the gradient $\frac{\partial f}{\partial \mathbf{D}}$ can be obtained by stacking all U_{mn} in Eq. (8.13)

$$\begin{aligned} \mathbf{U} &= \begin{bmatrix} \boldsymbol{\xi}_1^\top \mathbf{h}^{(1)} & \cdots & \mathbf{x}_1^\top \mathbf{h}^{(N)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\xi}_M^\top \mathbf{h}^{(1)} & \cdots & \boldsymbol{\xi}_M^\top \mathbf{h}^{(N)} \end{bmatrix} \\ &= \boldsymbol{\xi} \left[\mathbf{h}^{(1)} \cdots \mathbf{h}^{(N)}\right] \\ &= \boldsymbol{\xi} \boldsymbol{\beta}^\top, \end{aligned} \tag{8.14}$$

where we define $\boldsymbol{\beta} = \left[\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(N)}\right]^\top \in \mathbb{R}^{N \times P}$. By examining the nonzero elements position of $\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(N)}$, it is not difficult to find the relation between $\boldsymbol{\beta}$ and $\mathbf{g}^\top \mathbf{F}$

$$vec\left(\boldsymbol{\beta}\right)_\Lambda = \mathbf{Fg} \text{ and } vec\left(\boldsymbol{\beta}\right)_{\Lambda^c} = \mathbf{0}. \tag{8.15}$$

Now consider the second term V_{mn} of $\frac{\partial f}{\partial D_{mn}}$

$$\begin{aligned} V_{mn} &= \left(\mathbf{g}^\top \mathbf{F}\right) \left(\mathbf{I}_P \otimes \mathbf{D}^\top \mathbf{E}_{mn}\right)_{\Lambda,\Lambda} vec\left(\mathbf{A}\right)_\Lambda \\ &= \left(\mathbf{g}^\top \mathbf{F}\right) \left(\left[\mathbf{D}_m vec(\mathbf{A})_n \dots \mathbf{D}_m vec(\mathbf{A})_{(n+(P-1)N)}\right]^\top\right)_\Lambda \\ &= \mathbf{D}_m \sum_{p=1}^{P} \mathbf{A}_{n,p} \boldsymbol{\beta}^p, \end{aligned} \tag{8.16}$$

where $\boldsymbol{\beta}^p$ is the p^{th} column of $\boldsymbol{\beta}$. The differentiation $\frac{\partial f}{\partial \mathbf{D}}$ can be derived from V_{mn} in Eq. (8.16)

$$\begin{aligned} \mathbf{V} &= \begin{bmatrix} \mathbf{D}_1 \left[\sum_{p=1}^{P} \mathbf{A}_{1,p} \boldsymbol{\beta}^p, \cdots, \sum_{p=1}^{P} \mathbf{A}_{N,p} \boldsymbol{\beta}^p\right] \\ \vdots \\ \mathbf{D}_M \left[\sum_{p=1}^{P} \mathbf{A}_{1,p} \left(\mathbf{F}\mathbf{g}\right)_{\hat{p}} \cdots \sum_{p=1}^{P} \mathbf{A}_{N,p} \left(\mathbf{F}\mathbf{g}\right)_{\hat{p}}\right] \end{bmatrix} \\ &= \mathbf{D} \left[\sum_{p=1}^{P} \mathbf{A}_{1,p} \boldsymbol{\beta}^p \cdots \sum_{p=1}^{P} \mathbf{A}_{N,p} \boldsymbol{\beta}^p\right] \\ &= \mathbf{D}\boldsymbol{\beta}\mathbf{A}^\top, \end{aligned} \tag{8.17}$$

Combining Eq. (8.14) and Eq. (8.17), we reach the gradient of the dictionary

$$\frac{\partial \mathcal{L}}{\partial \mathbf{D}} = \boldsymbol{\xi}\boldsymbol{\beta}^\top - \mathbf{D}\boldsymbol{\beta}\mathbf{A}^\top. \tag{8.18}$$

8.3 Appendix C

We now demonstrate the details for the derivation of Eq. (7). Let $\boldsymbol{\alpha}^* \in \mathbb{R}^N$ be the optimal point of Lasso problem, which satisfies the first order optimality condition

$$\mathbf{D}^\top \mathbf{D}\boldsymbol{\alpha} - \mathbf{D}^\top \mathbf{x} + \boldsymbol{\Gamma}\text{sign}(\boldsymbol{\Gamma}\boldsymbol{\alpha}) + \lambda_2 \boldsymbol{\alpha} = \mathbf{0}, \tag{8.19}$$

where we have omitted the superscript '$*$' for simplicity. We first derive the gradient $\partial\boldsymbol{\alpha}/\partial D_{ij}$ for a single dictionary element D_{ij}. Differentiate both sides of Eq. (8.19) w.r.t. D_{ij}

$$\frac{\partial \mathbf{D}^\top \mathbf{D}}{\partial D_{ij}}\mathbf{x} + \mathbf{D}^\top \mathbf{D}\frac{\partial \boldsymbol{\alpha}}{\partial D_{ij}} - \frac{\partial \mathbf{D}^\top \mathbf{x}}{\partial D_{ij}} + \boldsymbol{\Gamma}\frac{\partial \text{sign}(\boldsymbol{\Gamma}\boldsymbol{\alpha})}{\partial D_{ij}} = \mathbf{0}. \tag{8.20}$$

The inactive atoms are not updated since the gradient $\partial\text{sign}(\boldsymbol{\Gamma}\boldsymbol{\alpha})/\partial D_{ij}$ on which $\alpha_j = 0$ is not well defined [59, 63]. Let Λ be the active set of the sparse code and only the active atoms are updated

$$\frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial D_{ij}}\mathbf{x} + \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda\frac{\partial \boldsymbol{\alpha}}{\partial D_{ij}} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial D_{ij}} + \boldsymbol{\Gamma}_\Lambda\frac{\partial \text{sign}(\boldsymbol{\Gamma}\boldsymbol{\alpha})_\Lambda}{\partial D_{ij}} = \mathbf{0}, \tag{8.21}$$

where $j \in \Lambda$, $\boldsymbol{\Gamma}_\Lambda \in \mathbb{R}^{K\times K}$ is the submatrix of $\boldsymbol{\Gamma}$ selected by the active set Λ and K is the cardinality of the active set Λ. We reach the updating rule for a single dictionary element

$$\frac{\partial L}{\partial D_{ij}} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot (\mathbf{D}^\top \mathbf{D} + \lambda_2 \mathbf{I})_\Lambda^{-1}(\frac{\partial \mathbf{D}_\Lambda^\top \boldsymbol{\alpha}}{\partial D_{ij}} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda}{\partial D_{ij}}\boldsymbol{\alpha}_\Lambda), \tag{8.22}$$

where $(\mathbf{D}^\top \mathbf{D} + \lambda_2 \mathbf{I})_\Lambda \in \mathbb{R}^{K\times K}$ is the submatrix of $(\mathbf{D}^\top \mathbf{D} + \lambda_2 \mathbf{I})$ selected by the active set Λ. Similarly, the gradient of sparse code $\boldsymbol{\alpha}$ w.r.t to each input signal element x_i can be reached by

differentiating both sides of Eq. (8.19) w.r.t. x_i

$$\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda \frac{\partial \boldsymbol{\alpha}}{\partial x_i} - \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i} + \boldsymbol{\Gamma}_\Lambda \frac{\partial \text{sign}(\boldsymbol{\Gamma}\boldsymbol{\alpha})_\Lambda}{\partial x_i} = \mathbf{0}. \tag{8.23}$$

Eq. (8.23) is equivalent with

$$\frac{\partial L}{\partial x_i} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot (\mathbf{D}^\top \mathbf{D} + \lambda_2 \mathbf{I})_\Lambda^{-1} \frac{\partial \mathbf{D}_\Lambda^\top \mathbf{x}}{\partial x_i}. \tag{8.24}$$

Finally, differentiating both sides of Eq. (8.19) w.r.t. λ_j

$$\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda \frac{\partial \boldsymbol{\alpha}}{\partial \lambda_j} + \mathbf{E}_j \text{sign}(\boldsymbol{\Gamma}\boldsymbol{\alpha})_\Lambda = \mathbf{0}, \tag{8.25}$$

where $\mathbf{E}_j \in \mathbb{R}^{K \times K}$ is the indicator matrix such that only the element (j, j) equals to one and zeros elsewhere. Eq. (8.25) can be further simplified as

$$\frac{\partial L}{\partial \lambda_{1j}} = \left(\frac{\partial L}{\partial \boldsymbol{\alpha}}\right)_\Lambda^\top \cdot -(\mathbf{D}^\top \mathbf{D} + \lambda_2 \mathbf{I})_\Lambda^{-1} \text{sign}(\boldsymbol{\Gamma}_\Lambda \boldsymbol{\alpha}_\Lambda)_j, \text{ s.t. } \lambda_{1j} \neq 0, \tag{8.26}$$

where $\text{sign}(\boldsymbol{\Gamma}_\Lambda \boldsymbol{\alpha}_\Lambda)_j = \mathbf{E}_j \text{sign}(\boldsymbol{\Gamma}\boldsymbol{\alpha})_\Lambda \in \mathbb{R}^K$.

Bibliography

[1] 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*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.

[2] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2010.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2012.

[4] G. Camps-Valls, D. Tuia, L. Bruzzone, and J. Atli, "Advances in hyperspectral image classification: Earth monitoring with statistical learning methods," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 45–54, Jan. 2014.

[5] X. Mei and H. Ling, "Robust visual tracking and vehicle classification via sparse

representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2259–2272, Nov. 2011.

[6] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[7] J. Benediktsson, P. Swain, and O. Ersoy, "Neural network approaches versus statistical methods in classification of multisource remote sensing data," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 540–552, Jul. 1990.

[8] L. Ma, M. M. Crawford, and J. Tian, "Local manifold learning-based k-nearest-neighbor for hyperspectral image classification," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4099–4109, Nov. 2010.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Apr. 2004.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2005.

[11] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[12] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[13] 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, May 2015.

[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," *in International Conference on Machine Learning (ICML)*, Jun. 2010.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *in International Conference on Machine Learning (ICML)*, Jul. 2015.

[17] Z. Jiang, Z. Lin, and L. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, Nov. 2013.

[18] 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*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[19] Y. Boureau, F. R. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2010, pp. 2559–2566.

[20] J. Yang, K. Yu, Y. Gong, and T. S. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2009, pp. 1794–1801.

[21] J. Yang, K. Yu, and T. S. Huang, "Efficient highly over-complete sparse coding using a mixture model," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Sept. 2010, pp. 113–126.

[22] S. Agarwal and D. Roth, "Learning a sparse representation for object detection," in *ECCV*, vol. 4, May 2002, pp. 113–130.

[23] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2008, pp. 1–8.

[24] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *in IEEE Computer*

Society Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2010, pp. 3501–3508.

[25] X. Sun, Q. Qu, N. M. Nasrabadi, and T. D. Tran, "Structured priors for sparse-representation-based hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 7, pp. 1235–1239, Jul. 2014.

[26] X. Sun, N. M. Nasrabadi, and T. D. Tran, "Task-driven dictionary learning for hyperspectral image classification with structured sparsity constraints," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 8, pp. 4457–4471, Aug. 2015.

[27] Q. Qu, X. Sun, N. M. Nasrabadi, and T. D. Tran, "Subspace vertex pursuit for separable non-negative matrix factorization in hyperspectral unmixing," *in ICASSP*, May 2014.

[28] X. Sun, N. M. Nasrabadi, and T. D. Tran, "Task-driven dictionary learning for hyperspectral image classification with structured sparsity priors," *in ICIP*, Oct. 2014.

[29] Xiaoxia, N. M. Nasrabadi, and T. D. Tran, "Sparse coding with fast image alignment via large displacement optical flow," *in ICASSP*, Mar. 2016.

[30] L. Liu, J. Glaister, X. Sun, A. Carass, T. D. Tran, and J. L. Prince, "Segmentation of thalamus from mr images via task-driven dictionary learning," *in Proceedings of SPIE*, Feb. 2016.

[31] X. Sun, N. M. Nasrabadi, and T. D. Tran, "Supervised multilayer sparse coding

networks for image classification," *CoRR*, vol. abs/1701.08349, 2017. [Online]. Available: http://arxiv.org/abs/1701.08349

[32] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni, "Recent advances in techniques for hyperspectral image processing," *Remote Sensing of Environment*, vol. 113, no. 6, pp. S110 – S122, 2009.

[33] G. Camps-Valls, L. Gomez-Chova, J. Munoz-Mari, J. Vila-Frances, and J. Calpe-Maravilla, "Composite kernels for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 1, pp. 93–97, Jan. 2006.

[34] L. Yang, S. Yang, P. Jin, and R. Zhang, "Semi-supervised hyperspectral image classification using spatio-spectral laplacian support vector machine," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 3, pp. 651–655, March 2014.

[35] J. Zhu, S. Rosset, R. Tibshirani, and T. J. Hastie, "1-norm support vector machines," in *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, S. Thrun, L. K. Saul, and P. B. Schölkopf, Eds., Dec. 2004, pp. 49–56.

[36] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.

[37] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification us-

ing dictionary-based sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 10, pp. 3973–3985, Oct. 2011.

[38] Q. Haq, L. Tao, F. Sun, and S. Yang, "A fast and robust sparse approach for hyperspectral data classification using a few labeled samples," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 6, pp. 2287–2302, Jun. 2012.

[39] R. Ji, Y. Gao, R. Hong, Q. Liu, D. Tao, and X. Li, "Spectral-spatial constraint hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 3, pp. 1811–1824, March 2014.

[40] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza, "Sparse unmixing of hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 6, pp. 2014–2039, Jun. 2011.

[41] J. Chen and X. Huo, "Theoretical results on sparse representations of multiple-measurement vectors," *IEEE Transactions on Signal Processing*, vol. 54, no. 12, pp. 4634–4643, Dec. 2006.

[42] S. Gao, I. Tsang, and L. Chia, "Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications," *IEEE Transactions on Pattern Analysis Machine Intelli- gence*, vol. 35, no. 1, pp. 92–104, Jan. 2013.

[43] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace

structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, Jan. 2013.

[44] A. Rakotomamonjy, "Surveying and comparing simultaneous sparse approximation (or group-lasso) algorithms." *Signal Processing*, vol. 91, no. 7, pp. 1505–1526, Jul. 2011.

[45] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-task regression with structured sparsity," *in International Conference on Machine Learning (ICML)*, pp. 543–550, Jun. 2010.

[46] P. Sprechmann, I. Ramirez, G. Sapiro, and Y. Eldar, "C-Hilasso: A collaborative hierarchical sparse modeling framework," *IEEE Transaction on Signal Processing*, vol. 59, no. 9, pp. 4183–4198, Sept. 2011.

[47] Y. Qian, M. Ye, and J. Zhou, "Hyperspectral image classification based on structured sparse logistic regression and three-dimensional wavelet texture features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 4, pp. 2276–2291, Apr. 2013.

[48] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit," *Signal Processing*, vol. 86, no. 3, pp. 572–588, Mar. 2006.

[49] S. Cotter, B. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear

inverse problems with multiple measurement vectors," *IEEE Transaction on Signal Processing*, vol. 53, no. 7, pp. 2477–2488, Jul. 2005.

[50] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.

[51] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Journal FTML*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[52] S. Wright, R. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transaction on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, Jul. 2009.

[53] F. J. Herrmann, M. P. Friedlander, and O. Yilmaz, "Fighting the curse of dimensionality: compressive sensing in exploration seismology," *IEEE Signal Processing Magazine*, vol. 29, no. 3, pp. 88–100, May 2012.

[54] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza, "Sparse unmixing of hyperspectral data," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 49, no. 6, pp. 2014–2039, Jun. 2011.

[55] J. Li, J. Bioucas-Dias, and A. Plaza, "Semisupervised hyperspectral image segmen-

tation using multinomial logistic regression with active learning," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4085–4098, Nov. 2010.

[56] K. Engan, S. Aase, and J. Hakon Husoy, "Method of optimal directions for frame design," *in ICASSP*, vol. 5, pp. 2443–2446, Mar. 1999.

[57] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transaction on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[58] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," *in International Conference on Machine Learning (ICML)*, pp. 689–696, Jun. 2009.

[59] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, Apr. 2012.

[60] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. Bach, "Supervised dictionary learning," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, pp. 1033–1040, Dec. 2008.

[61] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE Transaction on Image Processing*, vol. 21, no. 8, pp. 3467–3478, Aug. 2012.

[62] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annuals of Operation Research*, vol. 153, no. 1, pp. 235–256, Apr. 2007.

[63] J. Yang, K. Yu, and T. Huang, "Supervised translation-invariant sparse coding," *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2010.

[64] J. Zheng and Z. Jiang, "Tag taxonomy aware dictionary learning for region tagging," *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 369–376, Jun. 2013.

[65] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society, Series B*, vol. 67, pp. 301–320, Dec. 2005.

[66] D. M. Bradley and J. A. Bagnell, "Differentiable sparse coding," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2008.

[67] 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*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

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

[69] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[70] J. Mairal, R. Jenatton, R. B. Francis, and R. O. Guillaume, "Network flow algorithms for structured sparsity," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2010.

[71] Y. Chen, N. Nasrabadi, and T. Tran, "Hyperspectral image classification via kernel sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 1, pp. 217–231, Jan. 2013.

[72] 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*, vol. 34, no. 2, pp. 372–386, Feb. 2012.

[73] H. Lee, R. B. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *in International Conference on Machine Learning (ICML)*, Jun. 2009, pp. 609–616.

[74] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2009.

[75] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse

coding and vector quantization," *in International Conference on Machine Learning (ICML)*, Jul. 2011.

[76] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[77] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Partial Face Recognition: A Sparse Representation-based Approach," *in ICASSP*, Mar. 2016.

[78] K. Kavukcuoglu, P. Sermanet, Y. lan Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2010, pp. 1090–1098.

[79] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, Mar. 2011.

[80] D. Bertsimas and R. Weismantel, "Optimization over integers," *Athena Scientific*, 2005.

[81] J. Yang, Z. Wang, Z. Lin, X. Shu, and T. Huang, "Bilevel sparse coding for coupled feature spaces," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2012.

[82] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to

document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[83] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2014.

[84] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2007.

[85] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2006.

[86] Z. Wang, J. Yang, N. Nasrabadi, and T. Huang, "A max-margin perspective on sparse representation-based classification," in *in IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013.

[87] J. Huang, X. Huang, and D. Metaxas, "Simultaneous image transformation and sparse representation recovery," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2008.

[88] X. Shu, Y. Gao, and H. Lu, "Face recognition via robust face representation and compressive sensing," in *ISPACS*, Dec. 2010, pp. 1–4.

[89] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[90] S. Gao, Z. Zeng, K. Jia, T.-H. Chan, and J. Tang, "Patch set based representation for alignment-free image set classification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2015.

[91] S. Gao, K. Jia, L. Zhuang, and Y. Ma, "Neither global nor local: Regularized patch-based representation for single sample per person face recognition," *International Journal of Computer Vision*, vol. 111, no. 3, pp. 365–383, 2015.

[92] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *in IEEE International Conference on Computer Vision (ICCV)*, Sept. 2009.

[93] Y. LeCun, "Learning invariant feature hierarchies," *in European Conference on Computer Vision*, Oct. 2012.

[94] K. Sohn and H. Lee, "Learning invariant representations with local transformations," in *in International Conference on Machine Learning (ICML)*, Jul. 2012.

[95] Y. Chen, J. Mairal, and Z. Harchaoui, "Fast and robust archetypal analysis for representation learning," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[96] A. Cutler and L. Breiman, "Archetypal analysis," in *Technometrics*, vol. 36, no. 4, Nov. 1994, pp. 338–347.

[97] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Jun. 2006.

[98] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transaction on Information Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[99] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, Apr. 2011.

[100] T. Lin and H. T. Kung, "Stable and efficient representation learning with nonnegativity constraints," *in International Conference on Machine Learning (ICML)*, Jun. 2014.

[101] A. Coates and A. Y. Ng, "Selecting receptive fields in deep networks," in *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2011.

[102] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks," in *in International Conference on Machine Learning (ICML)*, Jun. 2013.

[103] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition,"

in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016.

[104] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[105] 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 International Conference on Machine Learning (ICML)*, Jul. 2015.

[106] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," *in International Conference on Machine Learning (ICML)*, Jun. 2010.

[107] Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi, "Unsupervised feature learning by deep sparse coding," *in SIAM International Conference on Data Mining*, Apr. 2014.

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

[109] K. He and J. Sun, "Convolutional neural networks at constrained time cost," *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[110] 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, Jul. 2012.

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

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

[113] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, "ICA with reconstruction cost for efficient overcomplete feature learning," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2011.

[114] H. Zou, "The adaptive lasso and its oracle properties," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, Dec. 2006.

[115] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, Dec. 2001.

[116] 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.

[117] V. C. Raykar and L. H. Zhao, "Nonparametric prior for adaptive sparsity," *Journal of Machine Learning Research*, vol. 9, pp. 629–636, May 2010.

[118] 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*, vol. 20, no. 7, pp. 1838–1857, Jul. 2011.

[119] C.-H. Z. Jian Huang, Shuangge Ma, "Adaptive Lasso for sparse high-dimensional regression models," *Statistica Sinica*, vol. 18, no. 4, pp. 1603–1618, Oct. 2008.

[120] P. O. Hoyer, "Non-negative sparse coding," *in IEEE Workshop on Neural Networks for Signal Processing*, Feb. 2002.

[121] J. Mairal, F. R. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, Mar. 2010.

[122] L. Zhuang, H. Gao, Z. Lin, Y. Ma, X. Zhang, and N. Yu, "Non-negative low rank and sparse graph for semi-supervised learning," *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2012.

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

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

BIBLIOGRAPHY

[125] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, Apr. 2009.

[126] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, May 2015.

[127] J. Mairal, "End-to-end kernel learning with supervised convolutional kernel networks," *CoRR*, vol. abs/1605.06265, May 2016. [Online]. Available: http://arxiv.org/abs/1605.06265

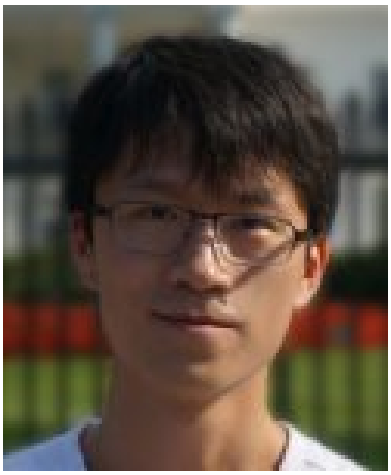
[128] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *CoRR*, vol. abs/1505.00387, May 2015.

[129] S. Zagoruyko and N. Komodakis, "Wide residual networks," *CoRR*, vol. abs/1605.07146, May 2016. [Online]. Available: http://arxiv.org/abs/1605.07146

[130] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," *in Conference and Workshop on Neural Information Processing Systems (NIPS)*, Dec. 2011.

[131] A. Patala, "Residual networks in torch (MNIST 100 layers)," https://deepmlblog.wordpress.com/2016/01/05/residual-networks-in-torch-mnist/, Jun. 2016.

Vita



Xiaoxia Sun was born in Shenyang, China in 1988. He received the B.Eng. degree in electronic engineering from Beihang University, Beijing, China, in 2011 and the M.Sc. degree in electrical and computer engineering from the Johns Hopkins University, Baltimore, MD, USA, in 2013.Since June 2012 he has been working as an Associate Member of Technical Staff at the U.S. Army Research Laboratory, Adelphi, MD, USA. He received M.S.E degree from the Johns Hopkins in 2012. He is currently with the Electrical and Computer Engineering Department, Johns Hopkins University. His research interests include the theory and applications of compressed sensing and sparse representations, deep learning and large-scale optimization problems.