SEMI-SUPERVISED AND DEEP LEARNING FOR HYPERSPECTRAL IMAGE ANALYSIS

A Dissertation

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in Electrical Engineering

By

Hao Wu

May 2017

SEMI-SUPERVISED AND DEEP LEARNING FOR HYPERSPECTRAL IMAGE ANALYSIS

Hao Wu

Approved:

Chair of the Committee
Saurabh Prasad, Assistant Professor,
Dept. of Electrical and Computer Engineering

Committee Members:

Zhu Han, Professor,
Dept. of Electrical and Computer Engineering

Thomas J. Hebert, Professor,
Dept. of Electrical and Computer Engineering

Miao Pan, Assistant Professor,
Dept. of Electrical and Computer Engineering

Demetrio Labate, Professor,
Dept. of Mathematics

Suresh K. Khator, Associate Dean,
Cullen College of Engineering

Badri Roysam, Department Chair,
Dept. of Electrical and Computer Engineering

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. Saurabh Prasad who guided me in choosing the right topic for my research and provided me valuable instructions throughout my PhD study. Without his persistent help, this dissertation and all my other achievements would not have been possible. I would also like to thank my committee members Dr. Han Zhu, Dr. Thomas Hebert, Dr. Miao Pan and Dr. Demetrio Labate for their support and valuable advices to improve this dissertation.

I would like to thank all of my labmates including Minshan Cui, Xiong Zhou, Yuhang Zhang, Jielian Guo, Tanu Priya, Lifeng Yan and Souvick Mukherjee for their help during my study and countless meaningful discussions on research.

Finally, I would like to thank my parents, Changxue Wu and Mingxiu Zhang, my grandmother Yinhong Chen, my sisters Ting Wu and Ying Wu for their endless love, support and encouragement in my whole life.

SEMI-SUPERVISED AND DEEP LEARNING FOR HYPERSPECTRAL IMAGE ANALYSIS

An Abstract

of a

Dissertation

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in Electrical Engineering

By

Hao Wu

May 2017

Abstract

Hyperspectral imaging is a technique which uses hyperspectral sensors to collect spectral information across the electromagnetic spectrum for each pixel in the image of a scene, with the purpose of identifying materials and detecting objects. The recorded hyperspectral data cover a wide range of wavelengths including visible and invisible light. The rich spectral information provides much more precise characteristics of materials, compared to natural color images. With such a wealth of spectral information, hyperspectral images have a variety of applications in remote sensing, such as target detection and land cover classification. In this dissertation, new algorithms and methods for semi-supervised learning and deep learning are proposed for hyperspectral image analysis. Specifically, a new semi-supervised dimensionality reduction algorithm named Semi-supervised Local Fisher Discriminant Analysis (SLFDA) is proposed to find a lower dimensional subspace for the high dimensional hyperspectral data, aiming to perform discriminant analysis on both labeled and unlabeled samples. Another semi-supervised learning system is also proposed that uses active learning to detect and identify unknown classes in a scene. We also present a deep learning method based on convolutional recurrent neural networks (CRNN) for hyperspectral data classification. Furthermore, a novel semi-supervised deep learning method that combines deep learning with semi-supervised learning is proposed for hyperspectral image classification. With extensive experiments on several real-world hyperspectral image datasets, we demonstrate that the proposed methods significantly outperform the state-of-the-art.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Remote Sensing and Hyperspectral Imagery

Remote Sensing [1–3], in contrast to on-site observation, is the technology of obtaining information about an object or area without making physical contact with the object. Remote sensing usually uses the satellite-based or aircraft-based sensor technologies to detect and classify objects on the surface of Earth based on the propagated signals (e.g., electromagnetic radiation). Generally, it can be split into active remote sensing (i.e., when a signal is emitted by a satellite or aircraft and its reflection by the object is detected by the sensor) and passive remote sensing (i.e., when the reflection of sunlight is detected by the sensor). Multispectral and hyperspectral imaging are a type of passive remote sensing, which produces an image where each pixel has spectral information (usually with hundreds of bands) that covers a wide range of wavelengths of light. Multispectral images are non-contiguous in their coverage of the spectrum, while hyperspectral images usually have dozens to hundreds of narrow contiguous bands. Thus hyperspectral images can provides much more precise characteristics of materials, compared to natural color images which have only three visible spectral bands (red, green and blue). With such a wealth of spectral information, hyperspectral images have a variety of applications in remote sensing, mineralogy, food science and astronomy, etc. This thesis will focus on suing hyperspectral images for remote sensing tasks, such as target detection, land cover classification and surveillance.

1.2 Machine Learning for Hyperspectral Image Analysis

Recently, with more and more high quality hyperspectral data being available, hyperspectral images are playing a key role for earth observation. At the same time, there has been a growing interest in development of machine leaning technologies for hyperspectral image analysis for remote sensing applications including target detection, anomaly detection and land cover classification.

The rich spectral information carried by hyperspectral images provides more precise measurement of characteristics of different types of objects, while the complicated distribution of the high dimensional data also makes it more difficult for effective analysis. For example, for classification problems, due to the high dimensionality of the data, it's hard to learn an accurate classifier, especially when we do not have enough training samples. Even when we have enough samples, the computation complexity is often high. High dimensionality not only significantly increases the time and memory requirements of algorithms, but also can degenerate their performance due to curse of dimensionality. Dimensionality reduction, projecting the original data to a lower dimensional subspace while preserving most of the intrinsic and discriminant information, is proven to be effective and efficient to handle high-dimensionality of the data and serves as a pre-processing step for tasks such as classification, regression and visualization. In general, there exists three types of dimensionality reduction methods, supervised, unsupervised, and semi-supervised. Supervised dimensionality reduction methods such as Fisher discriminant analysis (FDA) [4, 5] and local Fisher discriminant analysis (LFDA) [6] make use of the available labeled data to find low-dimensional embeddings that keep the discriminant information, while unsupervised methods such as principal component analysis (PCA) [7] and locality preserving projection (LPP)[8] use data without labels to find low-dimensional representations which try to keep the intrinsic structure of the data. Semi-supervised methods, such as semi-supervised discriminant analysis (SDA)[9] and semi-supervised local fisher discriminant analysis(SELF) [10], utilize both labeled and unlabeled data to find the subspace, which can be very useful and effective when we have limited labeled data and abundant unlabeled data, which is often the case in remote sensing applications.

Active learning [11] is a special case of semi-supervised machine learning in which a learning algorithm is able to interactively query label information for unlabeled data from the user (or some other information source). Active learning can be meaningful in remote sensing applications where unlabeled data are abundant but labeled data are very limited and annotation work is difficult, expensive, and time consuming. The goal of active learning is to achieve high classification performance by querying as few samples as possible from a large unlabeled data pool. The key aspect of active learning is to create an effective query strategy which is able to find the most informative samples and query a domain expert to provide additional information about these samples. A variety of query strategies have been proposed for active learning including uncertainty sampling, Query-By-Committee (QBC) and expected model change, etc [11, 12].

Unsupervised learning is a type of machine learning algorithm used to draw inferences from unlabeled data. The most common unsupervised learning task is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. Common parametric clustering algorithms include k-Means clustering, hierarchical clustering, and Gaussian mixture models (GMM), which require the number of clusters known in advance. On the other hand, non-parametric clustering methods such as the Dirichlet process mixture model (DPMM) make no assumptions about the underlying number of clusters and allows the number of clusters to grow as the the amount of data increases. DPMM [13–15] introduces a non-parametric prior over the number of clusters and the number of clusters are inferred automatically from the data. Due to this flexibility, it has been successfully applied in various clustering applications where the number of clusters is unknown a priori [16–19].

Deep learning, with the advances of computing power of computers and availability of large scale datasets, has become an important research topic within machine learning community. Deep neural networks can often be trained with a single end-to-end model and do not require traditional, task-specific feature extraction, which leads to their great success in a variety of applications such as computer vision [20–24], speech recognition [25–28], and natural language processing [29–31], etc. With the development of more advanced hyperspectral sensors, collecting large datasets has been easier than before and thus training deep neural networks on hyperspectral data has become possible. Recently, deep learning techniques, especially convolutional neural networks (CNN), have been introduced in the remote sensing community especially for hyperspectral data classification [32–37] and achieved state-of-the-art performance. Training a deep neural network for classification requires a large amount of labeled samples to learn a large number of parameters. However, for hyperspectral image classification in remote sensing applications, we are usually provided a large hyperspectral image with only a small amount of labeled samples available for training – collecting labeled data is expensive and time-consuming. On the other hand, we always have access to a large quantity of unlabeled data from a typical hyperspectral images. For such datasets, semi-supervised learning [38] techniques, which make use of both labeled and unlabeled data, hold promise.

1.3 Dissertation Contributions

The remainder of this dissertation is organized as follows. In Chapter 2, 3, 4, 5, we will go through the proposed algorithms and methods on semi-supervised learning and deep learning for hyperspectral image analysis with extensive experiments on real-world datasets. This dissertation focuses on semi-supervised learning and deep learning techniques for hyperspectral image analysis in remote sensing applications. The proposed semi-supervised learning techniques discussed in Chapter 2 and 3 solve the classification problems when we are given very limited labeled data but have access to abundant unlabeled data. The deep learning approaches discussed in Chapter 4 solves classification problems by training complex deep neural networks using large labeled datasets. Chapter **??** applies the deep learning methods discussed in Chapter 4 for hyperspectral and LiDAR sensor fusion. The semi-supervised deep learning method discussed in Chapter 5 solves classification problems by training complex deep neural networks using small amount of labeled data and large amount of unlabeled data. The main contributions of this dissertation are summarized as follows.

- In Chapter 2, a semi-supervised dimensionality reduction algorithm called Semi-Supervised Local Fisher Discriminant Analysis (SSLFDA) is proposed, aiming to perform discriminant analysis on both labeled and unlabeled samples. SPLFDA performs discriminative analysis on unlabeled data by utilizing the pseudo labels generated by the Dirichlet process mixture model (DPMM).
- In Chapter 3, an active learning system is developed. A new query strategy – local

information density (LID) is proposed, which provides robust classification performance with minimum manual labeling effort while simultaneously discovering unknown classes. The unknown classes are discovered as new clusters when we apply DPMM-based cluster analysis on the whole dataset including both labeled and unlabeled samples.

- In Chapter 4, several deep neural networks are analyzed and a deep convolutional recurrent neural network (CRNN) is designed for hyperspectral data classification which outperforms state-of-the-art. We also demonstrate how deep learning can be used for multi-sensor image classification within the remote sensing context.
- In Chapter 5, a new semi-supervised deep learning framework using pseudo labels is proposed, which involves pre-training a deep neural network using labeled and unlabeled data with pseudo labels followed by fine-tuning using labeled data with true labels. This semi-supervised deep learning framework provides significant improvement on classification performance compared to state-of-the-art methods.

Chapter 2

Semi-supervised Dimensionality Reduction

2.1 Introduction

The dimensionality of data is often very high in many modern applications such as pattern recognition, machine learning and computer vision. High-dimensional data not only significantly increase the time and memory requirements of algorithms, but also can degenerate their performance due to curse of dimensionality and redundant dimensions. For example, hyperspectral image provides spectral information of objects over a wide range of the electromagnetic spectrum, usually with hundreds of bands, which makes analysis of hyperspectral data a challenging problem. Dimensionality reduction, projecting the original data to a lower dimensional subspace while preserving most of the intrinsic and discriminant information, is proven to be effective and efficient to handle high-dimensional data and serves as a pre-processing step for tasks such as classification, regression and visualization.

In general, there exists two types of dimensionality reduction methods, supervised and unsupervised. Supervised dimensionality reduction methods such as Fisher discriminant analysis (FDA) [4, 5] make use of the available labeled data to find low-dimensional embeddings that keep the discriminant information, while unsupervised methods such as principal component analysis (PCA) [7] use data without labels to find low-dimensional representations which try to keep the intrinsic structure of the data.

FDA is a popular supervised dimensionality reduction method, which seeks an embedding transformation such that the between-class scatter is maximized and the within-class scatter is minimized. FDA is known to work well if samples in each class follow Gaussian distributions with a shared covariance structure. However, FDA tends to perform poorly if samples in some class follow a multimodal distribution, which means that some class can form several separate clusters. Within-class multimodality is often observed in a lot of practical machine learning applications. For example, in optical remote sensing, the distribution of vegetation classes could be multimodal due to different illumination conditions. To address this problem, local Fisher discriminant analysis (LFDA) [6] was proposed to preserve local structure of the data by localizing the evaluation of the within-class and between-class scatter. Furthermore, LFDA overcomes a critical limitation of FDA that the dimension of the FDA embedding space must be less than the number of total classes. While LFDA doesn't have such limitations in general.

In many practical applications, the number of labeled samples are very limited while the unlabeled samples are usually abundant, which is often the case in remote sensed hyperspectral image analysis where it's very expensive and time-consuming to acquire labeled pixels. When only a small number of labeled samples are available, all supervised dimensionality reduction methods (including FDA, LFDA, etc.) tend to have poor performance. The main reason lies in that supervised dimensionality reduction algorithms tend to find embedding spaces that are overfitted to the limited labeled samples. In such cases, it is helpful to make use of the unlabeled samples which are usually abundant and easy to acquire. Thus, by utilizing the labeled and unlabeled samples at the same time, semi-supervised dimensionality reduction methods can usually work better such as algorithms proposed in [9, 10, 39–42]. In [9], the authors proposed the semi-supervised discriminant analysis (SDA) which extends the FDA. SDA used labeled data to maximize the between-class scatter and unlabeled data to estimate the intrinsic geometric structure of the data by adding a regularizer term to the within-class scatter. SDA suffers from the same limitation as FDA when dealing with multimodally distributed data and it also ignores the relation between classes when using the unlabeled data. Similar to [9], [39] proposed a semi-supervised linear discriminant analysis (SSLDA) with an additional Tikhonov regularizer to the within-class scatter, which has the same limitations as [9]. Both [40] and [41] first calculated the labels of the unlabeled data via label propagation and then use the propagated labels to optimize the objective function of standard supervised dimensionality reduction algorithms. In [42], the authors proposed the semi-supervised discriminant analysis (SLDA) method which iterated the following two steps until convergence: (1) computes projection to optimize the criterion of LDA given the label matrix; (2) calculate the label matrix using the projected data. All methods involving calculating the labels of unlabeled samples may suffer from the inaccurate calculation of labels because of very small number of available labeled data. Thus the calculated labels may be quite different from the true labels and the resulting mapping will not be optimal for classification. [10] used LFDA to separate labeled samples and PCA to preserve the global structure of the unlabeled samples, which may not be optimal for classification since performing PCA on unlabeled samples may lose discriminate information.

In this paper, we propose a new semi-supervised dimensionality reduction method which we refer to as semi-supervised local Fisher discriminant analysis (SLFDA). Similar to [10], SLFDA separates (a small number of) labeled samples according to their labels by minimizing the local within-class scatter and maximizing the local between-class scatter. The difference lies in that [10] used PCA in order to preserve the global structure of the unlabeled data, but SLFDA tries to preserve the local cluster structure of the unlabeled samples based on the pseudo (cluster) labels by minimizing the local within-cluster scatter and maximizing the local between-cluster scatter, which can be regarded as an unsupervised LFDA (ULFDA) on the unlabeled data. In other words, we combine the supervised LFDA performed on labeled data and the unsupervised LFDA performed on unlabeled data to obtain our SLFDA.

The pseudo labels can be acquired through any clustering algorithm such as k-means, spectral clustering and Gaussian mixture models (GMM), Dirichlet process mixture models (DPMM), etc [43]. For parametric clustering methods like k-means, GMM and spectral clustering, the number of clusters needs to be specified as a parameter, which is often not an easy task. Thus in this work we choose to use DPMM for clustering due to its ability to infer the number of clusters from the data. DPMM [13–15] introduces a non-parametric prior over the number of clusters and allows the number of clusters to grow as the amount of data increases. It has been successfully applied in various clustering applications where the number of clusters is unknown a priori [16–19]. Markov chain Monte Carlo (MCMC) sampling methods [44, 45] are usually used for DPMMs to infer the posterior distributions of the desired latent variables. However, MCMC methods can be slow to converge and their convergence can be difficult to diagnose. One class of alternatives is provided by variational inference (VI) methods, which convert inference problems into optimization problems [46–48], i.e., VI methods construct a variational distribution to approximate the latent variables' posterior distribution by minimizing the Kullback-Leibler (KL) divergence between them. Hence, a variational inference approach is employed for the DPMM based clustering in our work.

There are a few recent work in utilizing pseudo labels for unsupervised feature selection: [49] used spectral clustering algorithms to learn the cluster labels of the unlabeled data which were then used for feature selection. [50] learned pseudo cluster labels via local learning regularized robust nonnegative matrix factorization. [51] incorporated learning discriminative features with generating pseudo labels. All of these existing methods utilized the concept of pseudo labels for unsupervised learning. However, to generate pseudo labels, they all required to specify the number of clusters a priori and used standard clustering algorithms such as k-means for the initialization of pseudo labels. Our proposed method circumvents the need to know the number of clusters by making use of the DPMM which infers the number of clusters from the data and allows more clusters to be found as data size increases.

SLFDA is a linear method which may have bad performance when the data manifold is highly nonlinear. Thus we also develop a kernel version of the proposed method for nonlinear dimensionality reduction, which is called KSLFDA.

The remainder of this chapter is organized as follows. Section 2.2 provides a brief introduction to the recently developed semi-supervised local Fisher discriminant analysis (SELF) [10] for dimensionality reduction. Section 2.3 provides a brief review of the DPMM and the variational inference algorithm. In section 2.4, we describe the proposed semi-supervised dimensionality reduction method SLFDA in detail. Section 2.5 presents a non-linear version of the proposed method based on the kernel trick. A description of three practical hyperspectral benchmarking datasets, and the experimental setup and results validating the proposed approach are detailed in 2.6. Finally, we summarize the key ideas and provide concluding remarks in Section 2.7.

2.2 Related Work

2.2.1 Notations

Let $\boldsymbol{x}_i \in \mathbb{R}^d$ be the i^{th} sample vector (e.g., pixel) with the corresponding class label $y_i \in \{1, ..., c\}$ and $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\} \in \mathbb{R}^{d\times N}$ be the data matrix. Among all the samples in $\boldsymbol{X} = [\boldsymbol{X}_l, \boldsymbol{X}_u]$, only a few samples $\boldsymbol{X}_l = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{n'}\}(1 \leq n' \leq N)$ are labeled and the rest $\boldsymbol{X}_u$ are unlabeled. Let $\boldsymbol{z}_i \in \mathbb{R}^r(1 \leq r \leq d)$ be the low dimensional representation of $\boldsymbol{x}_i$ via a transformation matrix $\boldsymbol{T} \in \mathbb{R}^{d\times r}$, which is calculated as

$$\boldsymbol{z}_i = \boldsymbol{T}^\top \boldsymbol{x}_i. \tag{2.1}$$

2.2.2 LFDA and RLFDA

LFDA can be formulated as an optimization problem [6, 10] expressed as

$$\boldsymbol{T}_{LFDA} = \underset{\boldsymbol{T}\in\mathbb{R}^{d\times r}}{\arg\max}\left[tr\left(\frac{\boldsymbol{T}^\top \boldsymbol{S}^{(lb)}\boldsymbol{T}}{\boldsymbol{T}^\top \boldsymbol{S}^{(lw)}\boldsymbol{T}}\right)\right], \tag{2.2}$$

where $\boldsymbol{S}^{(lb)}, \boldsymbol{S}^{(lw)} \in \mathbb{R}^{d\times d}$ are the local between-class scatter matrix and the local within-class scatter matrix. To construct $\boldsymbol{S}^{(lb)}$ and $\boldsymbol{S}^{(lw)}$, an affinity matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is used to quantify the affinity between samples, with A_{ij} defined as

$$A_{ij} = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{\sigma_i \sigma_j}\right), \tag{2.3}$$

where σ_i represents the local scaling around $\boldsymbol{x}_i$ defined by $\sigma_i = ||\boldsymbol{x}_i - \boldsymbol{x}_i{}^{(k)}||$ where $\boldsymbol{x}_i^{(k)}$ is the k^{th} nearest neighbor of $\boldsymbol{x}_i$ in the original feature space.

Then $\boldsymbol{S}^{(lb)}$ and $\boldsymbol{S}^{(lw)}$ are defined as

$$\boldsymbol{S}^{(lb)} = \sum_{i,j=1}^{n'} \frac{W_{ij}^{(lb)}}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \text{ and} \tag{2.4}$$

$$\boldsymbol{S}^{(lw)} = \sum_{i,j=1}^{n'} \frac{W_{ij}^{(lw)}}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top, \tag{2.5}$$

where n' is the number of labeled samples and $\boldsymbol{W}^{(lb)}, \boldsymbol{W}^{(lw)}$ are $n' \times n'$ weight matrices whose elements are defined as

$$W_{ij}^{(lb)} = \begin{cases} A_{ij}(1/n' - 1/n'_{y_i}), & \text{if } y_i = y_j \\ 1/n', & \text{if } y_i \neq y_j \end{cases} \quad \text{and} \tag{2.6}$$

$$W_{ij}^{(lw)} = \begin{cases} A_{ij}/n'_{y_i}, & \text{if } y_i = y_j \\ 0, & \text{if } y_i \neq y_j \end{cases}, \tag{2.7}$$

where n'_{y_i} denotes the number of samples in class y_i.

Note that when $A_{ij} = 1$ for all i, j (i.e., no locality), LFDA is reduced to FDA. Thus LFDA is regarded as a localized variant of FDA [6].

The solution to Eq. (2.2) is given by a generalized eigenvalue problem expressed as

$$\boldsymbol{S}^{(lb)}\boldsymbol{\varphi} = \lambda \boldsymbol{S}^{(lw)}\boldsymbol{\varphi}. \tag{2.8}$$

We assume that the generalized eigenvalues are sorted in decreasing order as

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d. \tag{2.9}$$

And $\{\boldsymbol{\varphi}_i\}_{i=1}^d$ are the corresponding generalized eigenvectors. Then the solution $\boldsymbol{T}^{(LFDA)}$ is analytically given as

$$\boldsymbol{T}_{LFDA} = (\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \cdots, \boldsymbol{\varphi}_r). \tag{2.10}$$

When we don't have enough labeled samples for training, the local within-class scatter matrix $\boldsymbol{S}^{(lw)}$ may be singular and overfitting tends to happen. A typical way to prevent the overfitting problem is to impose a regularizer to $\boldsymbol{S}^{(lw)}$, which results in a regularized local Fisher discriminant analysis (RLFDA) dimensionality reduction method. One of the most popular regularizers is the Tikhonov regularizer [52] and the corresponding optimization problem can be written as

$$\boldsymbol{T}_{RLFDA} = \underset{\boldsymbol{T}\in\mathbb{R}^{d\times r}}{\arg\max}\left[tr\left(\frac{\boldsymbol{T}^{\top}\boldsymbol{S}^{(lb)}\boldsymbol{T}}{\boldsymbol{T}^{\top}\left(\boldsymbol{S}^{(lw)}+\alpha\boldsymbol{I}\right)\boldsymbol{T}}\right)\right], \tag{2.11}$$

where the coefficient α controls the importance of the regularizer term and it usually takes a very small positive value.

2.2.3 SELF

SELF combines LFDA and PCA for semi-supervised dimensionality reduction in the way that the labeled data contributes to the projection via LFDA and PCA is used to preserve the global structure of the unlabeled data.

More specifically, the solution to SELF is given by the eigenvalue problem expressed as

$$\boldsymbol{S}^{(rlb)}\boldsymbol{\varphi} = \lambda\boldsymbol{S}^{(rlw)}\boldsymbol{\varphi}, \tag{2.12}$$

where $\boldsymbol{S}^{(rlb)}$ and $\boldsymbol{S}^{(rlw)}$ are the regularized local between-class scatter matrix and the regularized within-class scatter matrix defined as

$$\boldsymbol{S}^{(rlb)} = (1-\beta)\boldsymbol{S}^{(lb)} + \beta\boldsymbol{S}^{(t)} \text{ and} \tag{2.13}$$

$$\boldsymbol{S}^{(rlw)} = (1-\beta)\boldsymbol{S}^{(lw)} + \beta\boldsymbol{I}_d, \tag{2.14}$$

where $\beta \in [0,1]$ is a trade-off parameter which controls the importance of LFDA and PCA in SELF and $\boldsymbol{S}^{(t)}$ is the total scatter matrix defined as

$$\boldsymbol{S}^{(t)} = \sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^{\top}, \tag{2.15}$$

where $\boldsymbol{\mu}$ is the mean of all samples defined by

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i.$$

Unlike LFDA, $\boldsymbol{S}^{(rlw)}$ in SELF will not be singular (as long as $\beta > 0$) since it already has a regularier like term introduced by PCA.

2.3 DPMM and Variational Inference

DPMM is a type of Bayesian non-parametric models, in which the number of mixture components is not fixed in advance, but is determined by the model and data. The parameters of each component are generated by a Dirichlet Process (DP) [13], which can be seen as a distribution over the parameters of some other distributions. Each sample $\boldsymbol{x}_i$ is drew independently in turn from the chosen component given the parameters as follows:

$$\begin{aligned}
&G \sim DP(\alpha, G_0), \\
&\boldsymbol{\theta}_{z_i} \sim G, \text{ and} \\
&\boldsymbol{x}_i|\boldsymbol{\theta}_{z_i} \sim p(\boldsymbol{x}_i|\boldsymbol{\theta}_{z_i}).
\end{aligned}$$

The DP is parameterized by a base measure G_0 and a concentration parameter α. Both the base measure G_0 and the sample G from the DP can be seen as probability distributions over the component parameters $\boldsymbol{\theta}$. The concentration parameter $\alpha > 0$ determines the variance of the DP, i.e., it encodes how concentrated the samples from a DP will be around the base measure G_0. Intuitively, we can think of G as a randomly drawn distribution with mean G_0, and G will be more similar to G_0 with a larger α. z_i denotes the label of the mixture component with which the observation $\boldsymbol{x}_i$ is associated.

A popular representation of the DPMMs is the stick-breaking construction [53] where a draw G from a DP is represented as

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\theta}_k}, \tag{2.16}$$

where $\boldsymbol{\theta}_k$ are parameters of the k^{th} component, and $\delta_{\boldsymbol{\theta}_k}$ is an indicator function centered at $\boldsymbol{\theta}_k$ (zero elsewhere except for $\delta_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}_k) = 1$). $\pi_k \in [0, 1]$ is the mixing proportions (weights) of the k^{th} component, which are produced as

$$v_k \sim \mathcal{B}(1, \alpha);\ \pi_k = v_k \prod_{l=1}^{k-1}(1 - v_l), k = 1, 2, ..., \infty, \tag{2.17}$$

where $\mathcal{B}$ is the Beta distribution and it can be verified that $\sum_{k=1}^{\infty} \pi_k = 1$. Intuitively, the mixing proportions π_k are obtained by successively breaking a "stick" of unit length into an infinite number of pieces. The size of each successive piece is proportional to the length of the rest of the stick and the proportion is given by an independent draw from a $Beta(1, \alpha)$ distribution. To generate a sample $\boldsymbol{x}_i$, the component assignment variable z_i is chosen from a multinomial distribution parameterized by π_k. In this work, we use a DPMM with Gaussian mixtures which has the following likelihood functions as

$$p\left(\boldsymbol{x}|\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{R}_k\}\right) = \sum_{k=1}^{\infty} \pi_k \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{R}_k^{-1}\right), \tag{2.18}$$

where $\boldsymbol{\mu}_k$ and $\boldsymbol{R}_k$ are the mean vector and precision matrix of the k^{th} mixture component and we write $\boldsymbol{\theta}_k = \{\boldsymbol{\mu}_k, \boldsymbol{R}_k\}$ for simplicity. Correspondingly, the base distribution G_0 for the DP is chosen to be a conjugate prior for the likelihood, which is the Normal-Wishart (NW) distribution defined as

$$\begin{aligned} p(\boldsymbol{\theta}_k|G_0) = p(\boldsymbol{\mu}_k, \boldsymbol{R}_k|G_0) &= \mathcal{NW}(\boldsymbol{\mu}_k, \boldsymbol{R}_k|\boldsymbol{m}_0, r_0, \boldsymbol{B}_0, \nu_0) \\ &= \mathcal{N}(\boldsymbol{\mu}_k|\boldsymbol{m}_0, (r_0\boldsymbol{R}_k)^{-1})\mathcal{W}(\boldsymbol{R}_k|\boldsymbol{B}_0, \nu_0), \end{aligned} \tag{2.19}$$

where $\boldsymbol{m_0}$ is the mean vector and r_0 is the relative precision for the Gaussian prior on $\boldsymbol{\mu}_k$, $\boldsymbol{B}_0$ is the scale matrix and ν_0 is the degrees of freedom for the Wishart prior on $\boldsymbol{R}_k$, which is defined as

$$p(\boldsymbol{R}_k|\boldsymbol{B}_0, \nu_0) = \frac{1}{2^{\frac{\nu_0 d}{2}}|\boldsymbol{B}|^{\frac{\nu_0}{2}}\Gamma_d(\frac{\nu_0}{2})}|\boldsymbol{R}_k|^{\frac{\nu_0-d-1}{2}}e^{-\frac{1}{2}tr(\boldsymbol{B}^{-1}\boldsymbol{R}_k)}, \tag{2.20}$$

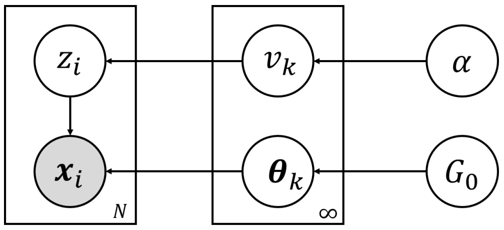


Figure 2.1: A graphical model representation of the DPMM, where the shaded and unshaded nodes indicate observed and latent variables respectively, and plates indicate repetition.

where $\Gamma_d(\cdot)$ is the d-dimensional gamma function. This Normal-Wishart distribution is a conjugate prior on $\boldsymbol{\theta}_k$, which means that the posterior will have the same form given that the likelihood is Gaussian.

In summary, DPMM generates observations according to the following generative process:

1. For $k = 1, 2, ..., \infty$:
   (a) Draw $v_k \sim \mathcal{B}(1, \alpha)$,
   (b) Compute $\pi_k = v_k \prod_{l=1}^{k-1} (1 - v_l)$.
2. For each component $k = 1, 2, ..., \infty$:
   (a) Draw precision $\boldsymbol{R}_k \sim \mathcal{W}(\boldsymbol{B}_0, \nu_0)$,
   (b) Draw mean $\boldsymbol{\mu}_k \sim \mathcal{N}(\boldsymbol{\mu}_0, (r_0 \boldsymbol{R}_k)^{-1})$.
3. For each observation $i = 1, 2, ..., N$:
   (a) Draw latent assignment $z_i \sim Mult(\boldsymbol{\pi})$,
   (b) Draw $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_{z_i}, \boldsymbol{R}_{z_i})$.

The $Mult(\boldsymbol{\pi})$ represents a multinomial distribution with parameter $\boldsymbol{\pi} = \{\pi_k\}_{k=1}^{\infty}$. The graphical representation for this generative model is depicted in Fig. 2.1.

The generative model for DPMM described above generates observations given model parameters. For inference, given data $\boldsymbol{X}$, our goal is to infer the model parameters, i.e., the cluster assignments $\boldsymbol{Z} = \{z_1, ..., z_N\}$ and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K\}$. Markov chain Monte Carlo (MCMC) sampling approaches such as Gibbs sampling [16, 17, 19, 44] can be used to infer the latent variables by obtaining samples from their posterior distribution $p(\boldsymbol{\Theta}, \boldsymbol{Z}|\boldsymbol{X})$. However, MCMC methods can be slow to converge and their convergence can be difficult to diagnose. To address this problem, variational inference (VI) methods [46, 48], which convert inference problems into optimization problems, were developed for inference of DPMMs in [47].

VI aims to approximate the desired posterior $p(\boldsymbol{V}, \boldsymbol{\Theta}, \boldsymbol{Z}|\boldsymbol{X})$ by some variational distribution $q(\boldsymbol{V}, \boldsymbol{\Theta}, \boldsymbol{Z})$ by minimizing the Kullback-Leibler (KL) divergence between them. The most common type of the variational distribution is the mean-field variational family, where latent variables and parameters are mutually independent and each governed by a distinct factor, which can be written as

$$\begin{aligned} q(\boldsymbol{V}, \boldsymbol{\Theta}, \boldsymbol{Z}) &= q(\boldsymbol{V})q(\boldsymbol{\Theta})q(\boldsymbol{Z}) \\ &= \left[\prod_t q(v_t)\right]\left[\prod_k q(\boldsymbol{\theta}_k)\right]\left[\prod_{i=1}^{N} q(z_i)\right]. \end{aligned} \tag{2.21}$$

In [47], a truncation level T is enforced for the variational distribution by setting $q(v_T = 1) = 1$, which leads to $\pi_t = 0$ for $t > T$ and $p(z_n > T) = 0$. In other words, the upper bound for the number of components is enforced to be T. Then we can rewrite the variational distribution in Eq. 2.21 as

$$q(\boldsymbol{V}, \boldsymbol{\Theta}, \boldsymbol{Z}) = \left[\prod_{t=1}^{T} q(v_t)\right]\left[\prod_{k=1}^{T} q(\boldsymbol{\theta}_k)\right]\left[\prod_{i=1}^{N} q(z_i)\right]. \tag{2.22}$$

The objective function, i.e., the KL divergence between the desired posterior and the variational distribution, can be expressed as

$$\begin{aligned}
&D_{KL}\left[q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})||p(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z}|\boldsymbol{X})\right]\\
&=\sum_{\boldsymbol{Z}}\int_{\boldsymbol{\Theta}}\int_{\boldsymbol{V}} q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})log\frac{q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})}{p(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z}|\boldsymbol{X})}\mathrm{d}\boldsymbol{V}\mathrm{d}\boldsymbol{\Theta}\\
&=-\sum_{\boldsymbol{Z}}\int_{\boldsymbol{\Theta}}\int_{\boldsymbol{V}} q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})log\frac{p(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})/p(\boldsymbol{X})}{q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})}\mathrm{d}\boldsymbol{V}\mathrm{d}\boldsymbol{\Theta}\\
&=-\sum_{\boldsymbol{Z}}\int_{\boldsymbol{\Theta}}\int_{\boldsymbol{V}} q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})log\frac{p(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})}{q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})}\mathrm{d}\boldsymbol{V}\mathrm{d}\boldsymbol{\Theta}+logP(\boldsymbol{X})\\
&=F+logP(\boldsymbol{X}),
\end{aligned}\tag{2.23}$$

where the first term F is called the free energy and the second term is constant given $\boldsymbol{X}$. Thus minimizing the KL divergence can be done by minimizing the free energy, which can be further simplified as

$$\begin{aligned}
F&=-\sum_{\boldsymbol{Z}}\int_{\boldsymbol{\Theta}}\int_{\boldsymbol{V}} q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})log\frac{p(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})}{q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})}\mathrm{d}\boldsymbol{V}\mathrm{d}\boldsymbol{\Theta}\\
&=\mathbb{E}\left[log\frac{q(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})}{p(\boldsymbol{V},\boldsymbol{\Theta},\boldsymbol{Z})}\right]_{q(\boldsymbol{Z},\boldsymbol{V},\boldsymbol{\Theta})}\\
&=\sum_{k=1}^{T}\mathbb{E}\left[log\frac{q(\boldsymbol{\theta}_k)}{p(\boldsymbol{\theta}_k)}\right]_{q(\boldsymbol{\theta}_k)}+\sum_{k=1}^{T}\mathbb{E}\left[log\frac{q(v_k)}{p(v_k)}\right]_{q(v_k)}+\sum_{i=1}^{N}\mathbb{E}\left[log\frac{q(z_i)}{p(\boldsymbol{x}_i|z_i,\boldsymbol{\Theta})p(z_i|\boldsymbol{V})}\right]_{q(z_i,\boldsymbol{V},\boldsymbol{\Theta})}.
\end{aligned}\tag{2.24}$$

The minimization of the free energy with respect to the variational distribution can be solved using the coordinate ascent variational inference (CAVI) algorithm [47, 54], which iteratively optimize each factor in Eq. (2.21) while holding others fixed until the free energy converges. For more detail about this parameter inference process, readers can refer to [47, 54, 55]. Using the properties of conjugate priors, the update rules for each factor can be easily calculated which is given as follows:

(a) For $t = 1, ..., T$:

$$q(v_t)=\mathcal{B}(a_{t1},a_{t2}),\tag{2.25}$$

which is a Beta distribution with parameters a_{t1} and a_{t2} defined as

$$a_{t1} = 1 + \sum_{i=1}^{N} q(z_i = t), \text{ and} \tag{2.26}$$

$$a_{t2} = \alpha + \sum_{i=1}^{N} \sum_{j=t+1}^{T} q(z_i = j). \tag{2.27}$$

(b) For $k = 1, ..., T$:

$$q(\boldsymbol{\theta}_k) = \mathcal{NW}(r_k, \boldsymbol{m}_k, \nu_c, \boldsymbol{B}_c), \tag{2.28}$$

where

$$r_k = r_0 + N_k, \tag{2.29}$$

$$\boldsymbol{m}_k = \frac{N_k \overline{\boldsymbol{x}}_k + r_0 \boldsymbol{m}_0}{r_k}, \tag{2.30}$$

$$\nu_k = \nu_0 + N_k, \text{ and} \tag{2.31}$$

$$\boldsymbol{B}_k = \boldsymbol{B}_0 + N_k \boldsymbol{S}_k + \frac{N_k r_0}{r_k} (\overline{\boldsymbol{x}}_k - \boldsymbol{m}_0)(\overline{\boldsymbol{x}}_k - \boldsymbol{m}_0)^\top, \tag{2.32}$$

and where we define

$$N_k = \sum_{i=1}^{N} q(z_i = k),$$
$$\overline{\boldsymbol{x}}_k = \frac{1}{N_k} \sum_{i=1}^{N} q(z_i = k) \boldsymbol{x}_i,$$
$$\boldsymbol{S}_k = \frac{1}{N_k} \sum_{i=1}^{N} q(z_i = k)(\boldsymbol{x}_i - \overline{\boldsymbol{x}}_k)(\boldsymbol{x}_i - \overline{\boldsymbol{x}}_k)^\top,$$

(c) For $i = 1, ..., N$:

$$q(z_i = k) = \frac{exp(S_{n,k})}{\sum_{k'=1}^{T} exp(S_{n,k'})}, \tag{2.33}$$

where

$$S_{n,k} = \mathbb{E}\left[log\ p(\boldsymbol{x}_i | \boldsymbol{\theta}_k)\right]_{q(\boldsymbol{\theta}_k)} + \mathbb{E}\left[log\ p(z_i = k | \boldsymbol{V})\right]_{q(\boldsymbol{V})}. \tag{2.34}$$

By iterating these updates, CAVI finds a local minimum of the free energy and the clustering label z_i for each sample $\boldsymbol{x}_i$ is given by $z_i = argmax_k q(z_i = k)$.

2.4 The Proposed Dimensionality Reduction Method

Similar to SELF, the labeled samples $\boldsymbol{X}_l$ are separated based on the LFDA criterion in SLFDA. Since we only have very limited number of labeled samples, we need to make use of the unlabeled samples $\boldsymbol{X}_u$ to prevent overfitting. The way that SLFDA use unlabeled samples is to obtain the pseudo labels of them, generated by a clustering algorithm, and then use the pseudo labels in LFDA to preserve the local structure of clusters.

2.4.1 Pseudo Labels Generation from DPMM Based Clustering

Pseudo labels can be very informative in dimensionality reduction when we don't have real class labels. We assume that each cluster is mainly dominated by a major class and a class can occupy more than one cluster. By separating different clusters in the embedding space, different classes can also be well separated. Thus the quality of clustering result is very important in our framework. Without prior knowledge of number of clusters, the ability to discover the potential number of clusters is important. Some clustering algorithms like k-means and GMMs may set the number of clusters to be the number of classes in the labeled data set. However, this is not correct for many practical data where some classes may have multimdal distribution which result in multiple clusters. For example, in optical remote sensed hyperspectral images, the distribution of some class can be very complicated and multimodal due to spatial variation and various illumination conditions. In such cases, simply setting the number of clusters to be the number of classes is not the preferred way. Thus we choose to use DPMM for clustering due to its robust performance and ability to automatically infer the number of clusters, as described in section 2.3. DPMMs are expected to discover more number of clusters than the number of classes, which turns out to be true in our experiments.

2.4.2 ULFDA Based on Pseudo Labels

The cluster labels $\boldsymbol{Z} = \{z_1, z_2, ..., z_N\}$ for each data point, generated by the DPMM, can be used as pseudo labels for supervised dimensionality reduction. We propose an unsupervised LFDA (ULFDA) which uses LFDA on the unlabeled data with pseudo labels. We define an unsupervised local between-cluster scatter matrix $\boldsymbol{S}^{(ulb)}$ and an unsupervised local within-cluster scatter matrix $\boldsymbol{S}^{(ulw)}$ as

$$\boldsymbol{S}^{(ulb)} = \sum_{i,j=1}^{N} \frac{W_{ij}^{(ulb)}}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \text{ and} \tag{2.35}$$

$$\boldsymbol{S}^{(ulw)} = \sum_{i,j=1}^{N} \frac{W_{ij}^{(ulw)}}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top, \tag{2.36}$$

where $\boldsymbol{W}^{(ulb)}, \boldsymbol{W}^{(ulw)}$ are $N \times N$ weight matrices with

$$W_{ij}^{(ulb)} = \begin{cases} A_{ij}(1/N - 1/N_{z_i}), & \text{if } z_i = z_j \\ 1/N, & \text{if } z_i \neq z_j \end{cases} \quad \text{and} \tag{2.37}$$

$$W_{ij}^{(ulw)} = \begin{cases} A_{ij}/N_{z_i}, & \text{if } z_i = z_j \\ 0, & \text{if } z_i \neq z_j \end{cases}, \tag{2.38}$$

where N_{z_i} denotes the number of samples in cluster z_i and the affinity matrix A is defined the same way as in LFDA.

The optimization problem for ULFDA can be formulated as

$$\boldsymbol{T}_{ULFDA} = \underset{\boldsymbol{T}\in\mathbb{R}^{d\times r}}{\arg\max} \left[tr\left(\frac{\boldsymbol{T}^\top \boldsymbol{S}^{(ulb)} \boldsymbol{T}}{\boldsymbol{T}^\top \boldsymbol{S}^{(ulw)} \boldsymbol{T}} \right) \right]. \tag{2.39}$$

And the solution to ULFDA is given by an eigenvalue problem expressed as

$$\boldsymbol{S}^{(ulb)}\boldsymbol{\varphi} = \lambda \boldsymbol{S}^{(ulw)}\boldsymbol{\varphi}. \tag{2.40}$$

Since we have a lot of unlabeled data to use in ULFDA, $\boldsymbol{S}^{(ulw)}$ will not have the singularity problem. By preserving the local cluster structure encoded in the pseudo labels, ULFDA can preserve more discriminative information than other unsupervised dimensionality reduction methods such as PCA.

2.4.3 SLFDA

Given a small number of labeled samples and abundant unlabeled samples, SLFDA seeks to separate the labeled samples based on their labels and preserve the local cluster structure of the unlabeled samples based on their pseudo labels. Similar to SELF, we define a semi-supervised between-class scatter matrix and a semi-supervised within-class scatter matrix as

$$\boldsymbol{S}^{(slb)} = (1-\beta)\boldsymbol{S}^{(lb)} + \beta\boldsymbol{S}^{(ulb)} \text{ and} \tag{2.41}$$

$$\boldsymbol{S}^{(slw)} = (1-\beta)\boldsymbol{S}^{(lw)} + \beta\boldsymbol{S}^{(ulw)}, \tag{2.42}$$

where $\boldsymbol{S}^{(lb)}$ and $\boldsymbol{S}^{(lw)}$ are defined in Eq. (2.4) and (2.5), and $\boldsymbol{S}^{(ulb)}$ and $\boldsymbol{S}^{(ulw)}$ are defined in Eq. (2.35) and (2.36). $\beta \in [0,1]$ is a trade-off parameter which controls the importance of labeled and unlabeled samples. In our implementation, a constant $\beta = 0.5$ is used to balance the effects of labeled ans unlabeled samples.

The optimization problem for SLFDA is expressed as

$$\boldsymbol{T}_{SLFDA} = \underset{\boldsymbol{T}\in\mathbb{R}^{d\times r}}{\arg\max}\left[tr\left(\frac{\boldsymbol{T}^\top\boldsymbol{S}^{(slb)}\boldsymbol{T}}{\boldsymbol{T}^\top\boldsymbol{S}^{(slw)}\boldsymbol{T}}\right)\right]. \tag{2.43}$$

In other words, SLFDA seeks a transformation matrix $\boldsymbol{T}$ such that the semi-supervised local between-class scatter in the embedding space is maximized and the semi-supervised local within-class scatter in the embedding space is minimized. In addition, the semi-supervised local within-class matrix will not be singular since we have abundant unlabeled data. Finally, the solution to SLFDA is given by the eigenvalue problem expressed as

$$\boldsymbol{S}^{(slb)}\boldsymbol{\varphi} = \lambda\boldsymbol{S}^{(slw)}\boldsymbol{\varphi}. \tag{2.44}$$

An outline of the SLFDA algorithm is shown in Algorithm 1.

Algorithm 1 *SLFDA*

```
Input: Labeled samples $\boldsymbol{X}_l = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n'}$, unlabeled samples $\boldsymbol{X}_u = \{\boldsymbol{x}_i\}_{i=n'+1}^{N}$, dimensionality of the subspace $r$, hyperparameters of DPMM $\{\alpha, \boldsymbol{m}_0, r_0, \boldsymbol{B}_0, \nu_0\}$.
Output: Transformation matrix $\boldsymbol{T}_{SLFDA}$.
1. Cluster the whole data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ to generate pseudo class labels $\boldsymbol{Z} = \{z_i\}_{i=1}^{N}$.
2. Calculate $\boldsymbol{S}^{(lb)}$ and $\boldsymbol{S}^{(lw)}$ using $\boldsymbol{X}_l$ according to Eq. (2.4) and (2.5).
3. Calculate $\boldsymbol{S}^{(ulb)}$ and $\boldsymbol{S}^{(ulw)}$ using $\boldsymbol{X}$ and $\boldsymbol{Z}$ according to Eq. (2.35) and (2.36).
4. Calculate $\boldsymbol{S}^{(slb)}$ and $\boldsymbol{S}^{(slw)}$ according to Eq. (2.41) and (2.42).
5. Solve the generalized eigenvalue problem in Eq. (2.44) and select $r$ eigenvectors associated with the $r$ largest eigenvalues to form $\boldsymbol{T}_{SLFDA}$.
```

2.5 Kernelization for Nonlinear Dimensionality Reduction

So far, we have focused on linear dimensionality reduction methods, which may fail to discover the intrinsic geometry when the data manifold is highly nonlinear. Using the standard kernel trick [56, 57], we can obtain nonlinear variants of many linear dimensionality reduction methods, such as Kernel PCA (KPCA), kernel LFDA (KLFDA) [6], and kernel SELF (KSELF) [10]. The derivation of kernel ULFDA (KULFDA) is the same as the KLFDA except that we use pseudo labels instead of true labels. In this section, we will show how to perform the proposed SLFDA in the reproducing kernel Hilbert space (RKHS), which gives us kernel SLFDA (KSLFDA).

For LFDA[6], which only uses the labeled data, let's first define two graph Laplacian matrices as

$$\boldsymbol{L}^{(lb)} = \boldsymbol{D}^{(lb)} - \boldsymbol{W}^{(lb)} \text{ and} \tag{2.45}$$

$$\boldsymbol{L}^{(lw)} = \boldsymbol{D}^{(lw)} - \boldsymbol{W}^{(lw)}, \tag{2.46}$$

where $\boldsymbol{D}^{(lb)}$ and $\boldsymbol{D}^{(lw)}$ are diagonal matrices with the i-th elements defined by: $D_{ii}^{(lb)} = \sum_{j=1}^{N} W_{ij}^{(lb)}$ and $D_{ii}^{(lw)} = \sum_{j=1}^{N} W_{ij}^{(lw)}$, and $W_{ij}^{(lb)}$ and $W_{ij}^{(lw)}$ are defined in Eq. (2.37) and (2.38). Then the local between-class scatter matrix and the local within-class scatter matrix in LFDA can be rewritten using the graph Laplacian matrices defined as

$$\boldsymbol{S}^{(lb)} = \boldsymbol{X}_l \boldsymbol{L}^{(lb)} \boldsymbol{X}_l^\top \text{ and} \tag{2.47}$$

$$\boldsymbol{S}^{(lw)} = \boldsymbol{X}_l \boldsymbol{L}^{(lw)} \boldsymbol{X}_l^\top . \tag{2.48}$$

Similarly, for ULFDA, the unsupervised local between-class scatter matrix and the unsupervised local within-class scatter matrix can be expresses as

$$\boldsymbol{S}^{(ulb)} = \boldsymbol{X} \boldsymbol{L}^{(ulb)} \boldsymbol{X}^\top \text{ and} \tag{2.49}$$

$$\boldsymbol{S}^{(ulw)} = \boldsymbol{X} \boldsymbol{L}^{(ulw)} \boldsymbol{X}^\top , \tag{2.50}$$

where the $N \times N$ graph Laplacian matrices are defined by

$$\boldsymbol{L}^{(ulb)} = \boldsymbol{D}^{(ulb)} - \boldsymbol{W}^{(ulb)} \text{ and} \tag{2.51}$$

$$\boldsymbol{L}^{(ulw)} = \boldsymbol{D}^{(ulw)} - \boldsymbol{W}^{(ulw)} . \tag{2.52}$$

For later use, we make the graph Laplacian matrices of LFDA have the same size as that in ULFDA (i.e., $N \times N$) by zero-padding as

$$\overline{\boldsymbol{L}}^{(lb)} = \begin{bmatrix} \boldsymbol{L}^{(lb)} & \mathbf{0}_{N_l \times (N-N_l)} \\ \mathbf{0}_{(N-N_l) \times N_l} & \mathbf{0}_{(N-N_l) \times (N-N_l)} \end{bmatrix} \text{ and} \tag{2.53}$$

$$\overline{\boldsymbol{L}}^{(lw)} = \begin{bmatrix} \boldsymbol{L}^{(lw)} & \mathbf{0}_{N_l \times (N-N_l)} \\ \mathbf{0}_{(N-N_l) \times N_l} & \mathbf{0}_{(N-N_l) \times (N-N_l)} \end{bmatrix} , \tag{2.54}$$

where $\mathbf{0}$ is a matrix with all elements equal to 0. Then Eq. (2.47) and (2.48) can be reformulated as

$$\boldsymbol{S}^{(lb)} = \boldsymbol{X} \overline{\boldsymbol{L}}^{(lb)} \boldsymbol{X}^\top \text{ and} \tag{2.55}$$

$$\boldsymbol{S}^{(lw)} = \boldsymbol{X} \overline{\boldsymbol{L}}^{(lw)} \boldsymbol{X}^\top . \tag{2.56}$$

For SLFDA, from Eq. (2.41) and (2.42), we can define the graph Laplacian matrices are defined by

$$\boldsymbol{L}^{(slb)} = \beta\overline{\boldsymbol{L}}^{(lb)} + (1-\beta)\boldsymbol{L}^{(ulb)} \text{ and} \tag{2.57}$$

$$\boldsymbol{L}^{(slw)} = \beta\overline{\boldsymbol{L}}^{(lw)} + (1-\beta)\boldsymbol{L}^{(ulw)}. \tag{2.58}$$

Then the semi-supervised local between-class scatter matrix and semi-supervised local within-class scatter matrix can be formulated as

$$\boldsymbol{S}^{(slb)} = \boldsymbol{X}\boldsymbol{L}^{(slb)}\boldsymbol{X}^\top \text{ and} \tag{2.59}$$

$$\boldsymbol{S}^{(slw)} = \boldsymbol{X}\boldsymbol{L}^{(slw)}\boldsymbol{X}^\top. \tag{2.60}$$

Based on this, the generalized eigenvalue problem of SLFDA in Eq. (2.44) can be expressed as

$$\boldsymbol{X}\boldsymbol{L}^{(slb)}\boldsymbol{X}^\top\boldsymbol{\varphi} = \lambda\boldsymbol{X}\boldsymbol{L}^{(slw)}\boldsymbol{X}^\top\boldsymbol{\varphi}. \tag{2.61}$$

When $d \leq N$, any vector $\boldsymbol{\varphi} \in \mathbb{R}^d$ can be expresses using some vector $\boldsymbol{\alpha} \in \mathbb{R}^N$ as $\boldsymbol{\varphi} = \boldsymbol{X}\boldsymbol{\alpha}$. Then multiplying Eq. (2.61) by $\boldsymbol{X}^\top$ from the left-hand side yields

$$\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{L}^{(slb)}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\alpha} = \lambda\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{L}^{(slw)}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\alpha}. \tag{2.62}$$

Let's define a $N \times N$ matrix $\boldsymbol{K}$ with the (i,j)-th element being $K_{ij} = \boldsymbol{x}_i^\top\boldsymbol{x}_j$. Then Eq. (2.62) becomes

$$\boldsymbol{K}\boldsymbol{L}^{(slb)}\boldsymbol{K}\boldsymbol{\alpha} = \lambda\boldsymbol{K}\boldsymbol{L}^{(slw)}\boldsymbol{K}\boldsymbol{\alpha}. \tag{2.63}$$

Now consider a nonlinear mapping $\phi(\boldsymbol{x})$ from the original space $\mathbb{R}^d$ to a reproducing kernel Hilbert space $\mathcal{H}$. Let $k(\boldsymbol{x}, \boldsymbol{x}')$ be the kernel function of $\mathcal{H}$. A typical choice would be the Gaussian kernel defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = exp(-||\boldsymbol{x} - \boldsymbol{x}'||^2/2\sigma^2). \tag{2.64}$$

For other choices please refer to [57]. Based on the reproducing property, $\boldsymbol{K}$ is now a kernel matrix in $\mathcal{H}$ with the (i,j)-th element defined as

$$K_{ij} = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle = k(\boldsymbol{x}, \boldsymbol{x}'), \tag{2.65}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathcal{H}$.

Let $\{\boldsymbol{\alpha}_i\}_{i=1}^N$ be the generalized eigenvectors associated with the generalized eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ of the generalized eigenvalue problem in Eq. 2.63. Then the $N \times r$ transformation matrix is analytically given as

$$\boldsymbol{T}_{KSLFDA} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdot\cdot\cdot, \boldsymbol{\alpha}_r). \tag{2.66}$$

Finally, a data point $\boldsymbol{x}$ can be embedded into the r dimensional subspace by

$$\boldsymbol{x} \rightarrow \boldsymbol{z} = \boldsymbol{T}_{KSLFDA}^{\top} \boldsymbol{K}(:, \boldsymbol{x}), \tag{2.67}$$

where $K(:, \boldsymbol{x}) = [k(\boldsymbol{x}_1, \boldsymbol{x}), \cdot, \cdot, \cdot, k(\boldsymbol{x}_N, \boldsymbol{x})]^{\top}$. An outline of the KSLFDA algorithm is shown in Algorithm 2.

Algorithm 2 *KSLFDA*

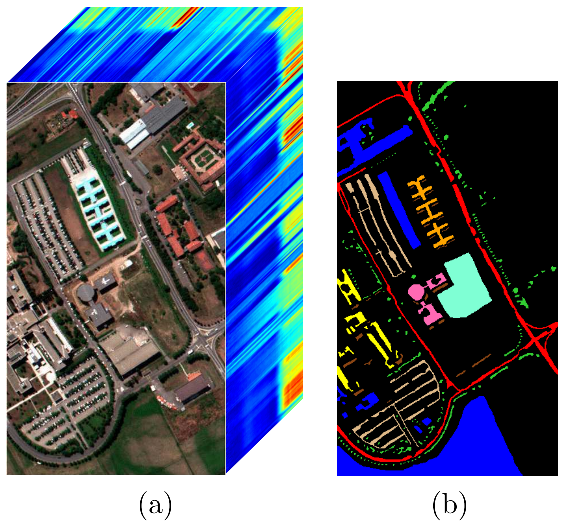
```
Input: Labeled samples $X_l = \{(x_i, y_i)\}_{i=1}^{n'}$, unlabeled samples $X_u = \{x_i\}_{i=n'+1}^{N}$, dimensionality of the subspace $r$, hyperparameters of DPMM $\{\alpha, m_0, r_0, B_0, \nu_0\}$, Gaussian kernel width $\sigma$.
Output: Transformation matrix $T_{KSLFDA}$.
1. Cluster the whole data $X = \{x_i\}_{i=1}^{N}$ to generate pseudo class labels $Z = \{z_i\}_{i=1}^{N}$.
2. Calculate $L^{(lb)}$ and $L^{(lw)}$ using $X_l$ according to Eq. (2.45) and (2.46).
3. Calculate $L^{(ulb)}$ and $L^{(ulw)}$ using $X$ and $Z$ according to Eq. (2.51) and (2.52).
4. Calculate $L^{(slb)}$ and $L^{(slw)}$ according to Eq. (2.57) and (2.58).
5. Solve the generalized eigenvalue problem in Eq. (2.63) and select $r$ eigenvectors associated with the $r$ largest eigenvalues to form $T_{KSLFDA}$.
```

2.6 Experiments

2.6.1 Datasets

We validate the proposed approaches with three hyperspectral imagery data sets: University of Pavia, University of Houston, and Galveston Wetland. The first two are standard benchmarking data sets collected over urban areas. The third hyperspectral imagery data was collected by our lab to study wetland species composition, which is included to show that the proposed methods also work for very different applications.

The first hyperspectral image dataset, the University of Pavia (UP), was collected using the Reflective Optics System Imaging Spectrometer (ROSIS-3) sensor [58]. This dataset has 103 spectral bands collected within the 430 $nm - 860$ nm wavelength range. The image has a spatial size of 610×340 pixels at a spatial resolution of 1.3 m per pixel. There are 9 land cover classes of interests considered in this dataset. A true color image and the ground truth map are shown in Fig. 2.2.



⬛	🟥	🟦	🟨	🟩
Unlabelled Area	Asphalt	Meadows	Gravel	Trees
🟧	🟩	🟪	⬜	🟫
Metal Sheets	Soil	Bitumen	Bricks	Shadows

Figure 2.2: (a) True color image and (b) ground truth of the University of Pavia hyperspectral imagery data.

The University of Houston (UH) hyperspectral image was acquired by the NSF-funded National Center for Airborne Laser Mapping (NCALM) over the University of Houston campus and the neighboring urban area using the ITRES-CASI (Compact Airborne Spectrographic Imager) 1500 hyperspectral imager. The hyperspectral image has 15 classes and contains 144 spectral bands over the $364nm-1046nm$ wavelength range. It has a dimension of 1905×349 pixels with a spatial resolution of $2.5m$. Fig. 2.3 shows the true color image of this data set with ground truth for 15 classes of interest.

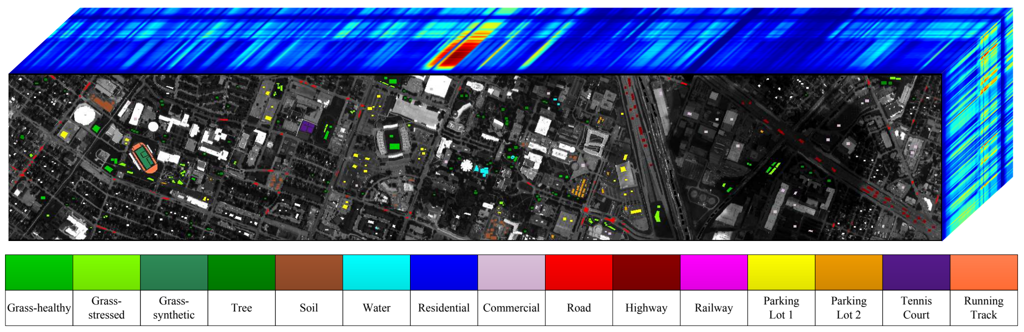


Figure 2.3: True color image of UH hyperspectral data with ground truth for 15 classes.

The third hyperspectral dataset, Galveston Wetland dataset, used in this work was acquired by us in Galveston, Texas in October, 2014. It was acquired at ground-level (side-looking views) over the wetlands using a Headwall Photonics hyperspectral imager which provides measurements in 163 spectral bands with a spatial dimension of $\times$ pixels. The image uniformly spanned the visible and near-infrared spectrum from $400nm - 1000nm$. The objects of interests are primarily vegetation species common in such wetlands.

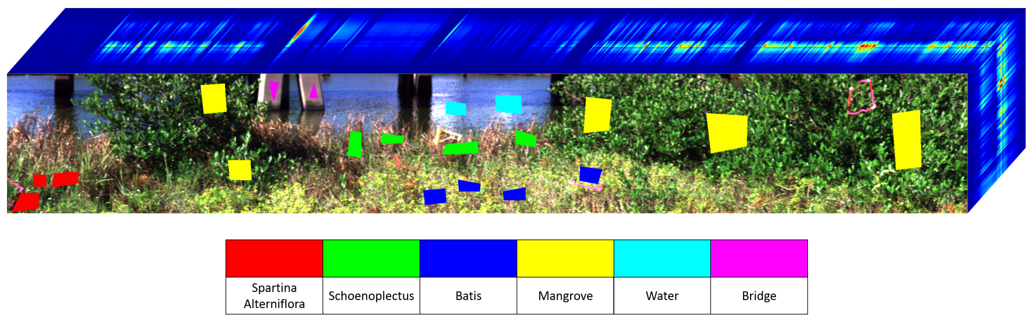


Figure 2.4: True color image of the Galveston wetland data set with ground truth for 6 classes.

2.6.2 Experimental Setup and Results

The efficacy of the proposed algorithms (ULFDA, SLFDA and their kernel variants) and the state-of-the-art methods are evaluated by the classification performance on their low dimensional embeddings using the three hyperspectral data sets described previously. In each experiment, the training data set includes 5, 10 or 20 labeled samples per class (which is a small number for hyperspectral image classification) and 200 unlabeled samples per class, and the test set includes 100 samples per class. For supervised dimensionality reduction methods, LFDA and RLFDA and their kernel variants, only labeled samples are used to compute the transformation matrix. For unsupervised methods, PCA and ULFDA and their kernel variants, all data (both labeled and unlabeled samples) are used to find the transformation matrix, but the labels of the labeled samples are not used. The transformation matrix for each algorithm is used to calculate the projections of the training and test data. The projected features of the labeled samples are used to train a nearest-neighbor classifier, which is then used to classify the test samples. For semi-supervised algorithms, SELF and SLFDA and their variants, both labeled and unlabeled samples are used to find the transformation. The dimension of the subspace for each algorithm is chosen by cross validation using the training data. For kernel based methods, the optimal value for the parameter of the Gaussian kernel function is also set by cross validation. Each experiment is repeated 10 times with randomly subsampled training and test sets, and the average accuracy is reported.

The parameter values used for each dataset are summarized next. For the University of Pavia dataset, the values for various parameters for each methods are set as follows: The regularization parameter α used in RLFDA and KRLFDA is 10^{-5}. For all semi-supervised methods, and their kernel versions, the trade-off parameter β is set to 0.5 to balance the importance of labeled and unlabeled samples. The dimension of the subspace for all dimensionality reduction methods is set to 10. For the variational inference in ULFDA and SLFDA, and their kernel versions, the truncation level T is set to be a number which is at least two times the number of classes in the dataset. In our experiments, we use $T = 20$ for this dataset. The value of the kernel parameter σ for different methods is set as: 0.5(KPCA), 0.5(KLFDA), 0.5(KRLFDA), 0.1(KULFDA), 5(KSELF), 0.6(KSLFDA).

For the University of Houston dataset, the values for various parameters for each methods are set as follows: The regularization parameter α used in RLFDA and KRLFDA is 10^{-5}. For all semi-supervised methods, and their kernel versions, the trade-off parameter β is set to 0.5 to balance the importance of labeled and unlabeled samples. The dimension of the subspace for all dimensionality reduction methods is set to 15. The truncation level T is set to 40 for this dataset. The value of the kernel parameter σ for different methods is set as: 0.5(KPCA), 0.5(KLFDA), 3(KRLFDA), 5(KULFDA), 1(KSELF), 8(KSLFDA).

For the Galveston Wetland dataset, the values for various parameters for each methods are set as follows: The regularization parameter α used in RLFDA and KRLFDA is 10^{-5}. For all semi-supervised methods, and their kernel versions, the trade-off parameter β is set to 0.5 to balance the importance of labeled and unlabeled samples. The dimension of the subspace for all dimensionality reduction methods is set to 10. The truncation level T is set to 20 for this dataset. The value of the kernel parameter σ for different methods is set as: 1(KPCA), 0.5(KLFDA), 0.5(KRLFDA), 0.8(KULFDA), 5(KSELF), 0.4(KSLFDA).

We show the classification performances for linear and kernel methods separately in Table 2.1, 2.2, 2.3, 2.4, 2.5, and 2.6, from which we have the following observations:

1. SLFDA has a better performance than any of other linear dimensionality methods and the performance of KSLFDA also surpasses other kernel methods on two benchmark

hyperspectral data sets and one new data set.

2. LFDA has the worst performance in all cases due to the small amount of labeled samples, which makes the local within-class scatter matrix $\boldsymbol{S}^{(lw)}$ singular or ill-conditioned, and overfitting may happen. With a Tikhonov regularizer added to $\boldsymbol{S}^{(lw)}$, RULFDA becomes less overfitted and achieves a significant improvement in performance compared to LFDA.

3. The benefits of pseudo labels are huge and significant, which is demonstrated by comparing the performances of PCA and ULFDA. PCA produces a subspace where the global structure of the data is preserved, which may not be optimal for classification. With pseudo labels, ULFDA can preserve more discriminative information between difference classes because each cluster generated by DPMM can be very pure. In other words, each cluster is mainly dominated by only one class, so different classes are separated by separating different clusters. Thus the subspace produced by ULFDA can be very beneficial to classification.

4. For semi-supervised algorithms, the proposed SLFDA/KSLFDA have much better performances than SELF/KSELF, which again shows the benefits of pseudo labels because the difference between SLFDA and SELF lie in the way of handling unlabeled data: the former makes use of pseudo labels to preserve local cluster structure and the latter use PCA to preserve global structure. We can also notice that, for most cases, the accuracy of ULFDA/KULFDA is even higher that that of SELF/KSELF, which means that even without labeled data, ULFDA can still generate very discriminative features provided by the pseudo labels.

In ULFDA, SLFDA and their kernel versions, DPMM is employed for clustering and the resulting number of clusters is usually larger than the number of classes in the dataset because of multimodality of some classes. In our experiments, for the University of Pavia

Table 2.1: Classification performances of linear methods on University of Pavia dataset: mean accuracy (%) with standard deviation (%).

University of Pavia	5 labeled/class	10 labeled/class	20 labeled/class
PCA	71.6(1.9)	74.2(0.9)	77.2(1.1)
LFDA	43.5(3.3)	33.8(5.3)	72.6(2.5)
RLFDA	69.4(3.6)	73.7(2.1)	81.0(1.5)
SELF	73.7(1.7)	76.8(1.1)	79.5(1.6)
ULFDA(proposed)	77.1(1.1)	80.1(1.0)	84.0(1.1)
SLFDA(proposed)	78.9(1.4)	81.5(1.0)	85.4(1.1)

Table 2.2: Classification performances of kernel methods on University of Pavia data: mean accuracy (%) with standard deviation (%).

University of Pavia	5 labeled/class	10 labeled/class	20 labeled/class
KPCA	69.0(1.5)	70.3(1.1)	71.3(1.0)
KLFDA	61.8(5.2)	71.7(4.6)	73.1(3.6)
KRLFDA	69.5(4.0)	76.9(2.4)	81.6(1.4)
KSELF	73.2(1.8)	76.3(1.1)	78.5(1.3)
KULFDA(proposed)	83.9(4.6)	84.6(2.8)	88.1(2.7)
KSLFDA(proposed)	84.9(1.8)	85.1(1.6)	88.2(1.8)

Table 2.3: Classification performances of linear methods on University of Houston data: mean accuracy (%) with standard deviation (%).

University of Houston	5 labeled/class	10 labeled/class	20 labeled/class
PCA	65.4(1.6)	72.1(1.2)	78.3(1.1)
LFDA	36.3(10.2)	22.4(3.8)	73.8(1.2)
RLFDA	64.7(2.7)	63.3(2.8)	74.5(1.3)
SELF	68.6(1.7)	75.8(1.1)	83.1(1.1)
ULFDA(proposed)	68.1(1.8)	75.5(1.1)	81.8(0.7)
SLFDA(proposed)	69.3(2.3)	76.2(1.3)	86.9(1.4)

Table 2.4: Classification performances of kernel methods on University of Houston data: mean accuracy (%) with standard deviation (%).

University of Houston	5 labeled/class	10 labeled/class	20 labeled/class
KPCA	60.2(1.7)	66.0(1.3)	71.1(1.1)
KLFDA	65.0(2.7)	72.7(2.1)	75.4(1.9)
KRLFDA	70.3(2.2)	76.2(2.1)	79.7(2.2)
KSELF	68.7(1.5)	76.1(0.9)	83.3(1.3)
KULFDA(proposed)	71.1(1.1)	77.7(0.8)	83.9(0.9)
KSLFDA(proposed)	71.7(1.6)	79.5(1.2)	85.9(0.8)

dataset which has 9 classes, DPMM finds 11.38 (1.77) clusters. For the University of Houston dataset which has 15 classes, DPMM finds 20.88 (1.96) clusters. For the Galveston

Table 2.5: Classification performances of linear methods on Galveston Wetland data: mean accuracy (%) with standard deviation (%).

Galveston	5 labeled/class	10 labeled/class	20 labeled/class
PCA	73.0(4.1)	78.0(1.8)	81.1(0.9)
LFDA	36.2(13.6)	34.5(6.7)	30.1(5.2)
RLFDA	78.9(3.4)	81.6(2.5)	80.3(2.4)
SELF	77.6(3.0)	81.8(1.7)	84.7(1.0)
ULFDA(proposed)	78.2(2.6)	82.5(1.8)	83.9(1.1)
SLFDA(proposed)	81.8(3.2)	85.2(1.7)	86.7(1.6)

Table 2.6: Classification performances of kernel methods on Galveston Wetland data: mean accuracy (%) with standard deviation (%).

Galveston	5 labeled/class	10 labeled/class	20 labeled/class
KPCA	70.1(4.9)	74.6(1.7)	77.9(0.9)
KLFDA	62.7(2.2)	65.9(7.9)	79.8(3.7)
KRLFDA	70.8(3.7)	79.8(3.5)	84.9(2.3)
KSELF	77.3(2.9)	81.3(1.9)	84.1(1.0)
KULFDA(proposed)	81.4(5.6)	87.3(2.8)	87.4(2.2)
KSLFDA(proposed)	86.9(2.5)	87.8(2.2)	87.6(2.3)

Wetland dataset which has 6 classes, DPMM finds 12.75 (1.83) clusters. To show the advantage of DPMM, we also implement a k-means based ULFDA/SLFDA and compare their classification performance in Table 2.7. It's clear to see that DPMM based unsupervised (ULFDA and KULFDA) and semi-supervised (SLFDA and KSLFDA) methods significantly outperform the k-means based methods.

Table 2.7: Comparison of k-means based and DPMM based unsupervised and semi-supervised dimensionality reduction methods on the Galveston Wetland dataset.

Galveston	k-means	DPMM
ULFDA	76.6(3.7)	78.2(2.6)
SLFDA	76.7(3.2)	81.8(3.2)
KULFDA	69.1(4.7)	81.4(5.6)
KSLFDA	80.2(2.9)	86.9(2.5)

2.7 Conclusions

Discriminant analysis is widely adopted in dimensionality reduction methods to extract useful and discriminative features for classification. However, due to lack of enough labeled samples, supervised dimensionality reduction methods tend to overfit and fail to find the optimal lower dimensional embedding space, and traditional unsupervised methods are unable to perform discriminant analysis. In this chapter, a novel semi-supervised dimensionality reduction method is introduced which can extract discriminative information in unlabeled data by making use of pseudo labels learned from DPMM based clustering. Experimental results have shown that the proposed method significantly improves the classification performance compared to the state-of-the-art methods on three practical hyperspectral imagery data sets. To the best of our knowledge, this is the first work which introduces a semi-supervised dimensionality reduction with the help of pseudo labels. The proposed method has profound benefits for a variety of applications when we don't have enough labeled samples for training but the unlabeled samples are quite abundant.

Chapter 3

Active Learning with Unknown Classes

3.1 Introduction

Active learning [11] is well-motivated in many machine learning problems such as remote sensing [12, 59, 60], image retrieval [61], speech recognition [62] and natural language processing [63, 64], where unlabeled data are abundant but labeled data is very limited and annotation work is difficult, expensive and time consuming. The main goal of an active learning algorithm is to achieve better classification performance by inducting as few samples as possible from an unlabeled data pool that are labeled by an annotator and added to the training pool.

A key aspect of active learning is the construction of effective query strategy which is designed to find the most informative samples and pose queries. A variety of query strategies have been created for active learning including uncertainty sampling, Query-By-Committee (QBC) and expected model change, etc [11, 12]. While there has been substantial work on active learning for classification, active learning with unknown class discovery has received considerably less attention. Traditional active learning systems assume that we have labeled data for every class of interest, even if the number of training samples initially available per class is small. However, we may encounter situations where we do not have labeled data for all classes, i.e., nothing is known about possibly new classes in the unlabeled data pool. This is common, especially in remote sensing applications, where there may be unknown or new classes in unexplored geospatial areas. In such scenarios, traditional active learning will not work well with regards to the detection of the unknown classes. Thus, we aim to design an active learning system which is capable of detecting unknown classes as fast as possible and queries the most informative samples from them.

The Dirichlet process mixture model (DPMM) [13, 45, 65, 66], a non-parametric Bayesian model, is well-suited to data with complex mixture structures. A widely used DPMM is the infinite Gaussian mixture model (IGMM) [15] which overcomes the requirement of the number of mixture components in traditional Gaussian mixture modeling (GMM) by assuming that data comes from a Gaussian mixture model with an infinite number of mixture components. Due to the flexibility of DPMM, it has been used in many applications [16, 18, 67, 68] for clustering and density estimation, etc. Thus, under DPMM, when a new class emerges, we will have one new cluster assigned to it or a few clusters due to the possible multimodal distribution of that class. We note that with remote sensing images, due to spatial variability, new clusters do not have to be new classes — they may also come from a known class from a different spatial area than the labeled data. In [69, 70] the new class detection problem has been addressed and solved by utilizing a DPMM, but detecting the new classes is just the first step to classification. In [69, 70] the new class detection problem has been addressed and solved by utilizing a DPMM. Detecting new classes is the first step to classification, so we carry it on to build an active learning framework with the ability of unknown class detection.

In this work we integrate the new class detection problem into an active learning system for efficient classification. Although some work has been done on this problem based on

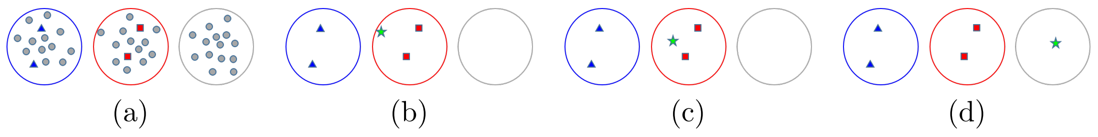


Figure 3.1: An illustration comparing different methods for active learning with an unknown class.

DPMM in [71] which aims to discover rare classes in a pool of unlabeled images. The query strategy in [71] was efficient only for rare class discovery and assumed that every class (both known and unknown) occupies exactly one cluster. In our proposed framework, the unknown classes do not have to be rare. Additionally, our framework does not make an assumption about the number of clusters occupied by the known or unknown classes. This makes our method suitable for remote sensing, where classes may have multi-modal distributions (e.g., due to spatial variability within such images classes can occupy more than one cluster).

The proposed query strategy — local information density (LID), is built on a combination of conventional uncertainty based active query strategy and a local density generated by clustering, which makes it more suited for classification. Moreover, when new clusters are detected, we give priority to query data from new clusters which can contain data from either new classes or different spatial areas of known classes. That is, we wish to discover new classes and explore new clusters of existing classes as quickly as possible, since both cases lead us to query data considered to be informative for the underlying active learning and image classification task. Fig. 3.1 gives an illustrative example demonstrating the effectiveness of our proposed approach in dealing with unknown classes, compared to traditional active learning approaches. Uncertainty sampling methods [11, 63] tend to query the most uncertain samples for the classifier based on the current training set. The queried samples are mainly located around the boundary between known classes, as illustrated in Fig. 3.1 (b). In such a scenario, the unknown class will correspond to "certain samples", because it's easy to classify them as the "square" class based on the current training set. Thus the uncertainty method cannot effectively detect the unknown class. Information density (ID) based method, as proposed in [63], considers uncertainty and density at the same time, aiming to query uncertain samples that are representative of the unlabeled pool, i.e., have large global density. It helps to prevent the system from querying outliers which usually have high uncertainty but are not helpful to classification. A simulated results for ID is shown in Fig. 3.1 (c), which failed to detect the unknown class because there is no guarantee that samples of unknown classes have high information density, which is in fact data dependent. For example, in a remote sensing scenario where the known classes are mainly vegetation classes and an unknown class is spectrally different, such as an urban class, the unknown class is more likely to have low information density because it is located far away from the known ones. Thus, the value of information density will not be large because the density term in the definition of information density, as described in section 3.2, has a relatively low value. Thus in this case, ID based methods cannot effectively detect the unknown class. However, the proposed LID method takes into account both uncertainty and a local density. It queries uncertain samples which have a high local density which is calculated within clusters generated by DPMM. When unknown classes are detected as new clusters by DPMM, some samples of unknown classes will have a high local density within the new clusters. Thus, by quickly identifying the new class as an emerging cluster, LID based method can direct active queries from that emerging cluster, as illustrated in Fig. 3.1 (d).

Hyperspectral images provide spectral information of objects over a wide range of the electromagnetic spectrum, usually with hundreds of bands, which yields precise characteristics of materials in the scene, compared to natural color images and multispectral images (often less than 10 bands). However, the rich spectral information also comes with a big challenge, owing to the high dimensionality especially when using statistical model for processing or analyzing such data. Dimensionality reduction is hence commonly undertaken for feature extraction of hyperspectral images. Popular methods include unsupervised algorithms such as principal component analysis (PCA), Isomap [72], locally linear embedding (LLE) [73, 74] and locality preserving projection (LPP) [75] and supervised algorithms such as Fisher linear discriminant analysis (LDA) and local Fisher discriminant analysis [6] and local tangent space alignment (LTSA) [74, 76]. There are also semi-supervised dimension reduction methods such as Semi-supervised local Fisher discriminant analysis (SELF) [10] but they have not been developed for or studied in the context of unknown class discovery. In our framework, we employ SELF as a preprocessing, with the goal of maximizing separation between known classes in a lower dimensional subspace, while also trying to preserve the local structure of unlabeled samples which may contain data of unknown classes — i.e., we want to find a subspace where we can discriminate data from known classes without confusing data from unknown classes.

The remainder of this chapter is organized as follows. Section 3.2 provides a brief literature review about related work. The proposed framework is described in detail in section 3.3. The experimental datasets, setup and results validating the proposed approach are detailed in 3.4. Section 3.5 summarizes the key ideas and experimental results in this work and provides concluding remarks.

3.2 Related Work

3.2.1 Active Learning

In general, an active learning system starts with a small labeled dataset $\mathcal{L}$. It iteratively selects the most informative samples from the unlabeled dataset $\mathcal{U}$, queries their labels (which is done by huaman annotator in real world application) and adds them to $\mathcal{L}$, aiming to improve the classification performance with the least number of queried samples. Three main scenarios for active learning problem have been considered in the literature — (i) pool-based sampling, (ii) stream-based selective sampling, and (iii) membership query synthesis. Since pool-based scenario is the most widely used used in remote sensing, we only consider this scenario where samples are queried from a large unlabeled data pool and labeled by a human annotator. The key task in active learning involves evaluating the informativeness of each unlabeled sample based on an appropriate query strategy $\phi(\cdot)$.

For posterior probability based active learning methods, one of most common query strategies for evaluating informativeness is uncertainty sampling [77] which includes least confidence (LC), breaking ties (BT), entropy, etc. The least confidence strategy queries the instance for which the current model has the least confidence in its most likely labeling, which is defined as

$$\phi_{LC}(\boldsymbol{x}) = 1 - P(\hat{y}|\boldsymbol{x}), \tag{3.1}$$

where $\hat{y}$ is the most probable class label for $\boldsymbol{x}$, i.e., $\hat{y} = argmax_y P(y|\boldsymbol{x})$.

Breaking ties queries the instance with the smallest difference between posteriors for its two most likely labelings, which is defined as

$$\phi_{BT}(\boldsymbol{x}) = P(\hat{y}_1|\boldsymbol{x}) - P(\hat{y}_2|\boldsymbol{x}), \tag{3.2}$$

where $\hat{y}_1$ and $\hat{y}_2$ are the first and second most probable class labels for $\boldsymbol{x}$ under the current model.

Entropy based uncertainty sampling queries the instance which has the largest entropy:

$$\phi_E(\boldsymbol{x}) = -\sum_{j=1}^{N_c} P(y_j|\boldsymbol{x}) log P(y_j|\boldsymbol{x}), \tag{3.3}$$

where N_c is the total number of possible class labels.

Query-By-Committee (QBC) [78], another active learning framework, measures informativeness based on the degree of disagreement between a committee of classification models. Another framework is the expected model change, which queries the instance that will result in the biggest change to the current model if added to the labeled set. An example of this framework is the expected gradient length (EGL) [79].

However, the conventional active learning frameworks such as those described above have a drawback that they are prone to querying outliers which typically have high uncertainty and large disagreement between a committee. For uncertainty sampling, outliers and the most uncertain samples lie on the classification boundary and they are not "representative" of other samples in the distribution, thus knowing their labels is unlikely to improve the classification performance on the data as a whole. Representativeness measures the ability of a sample to express the distribution of the whole data. Thus outliers with high informativeness are not representative of the unlabeled samples — as a result, they are not beneficial for classification. Similarly, QBC and EGL will spend time querying possible outliers simply because they are controversial or they are expected to impart the most significant change to the model. This is commonly seen during the initial query steps of active learning, when we do not have enough labeled samples, especially for classifiers built from generative models.

To address this, information density (ID) as a query strategy was proposed in [63],which aims to combine informativeness and representativeness, and is defined as

$$\phi_{ID}(\boldsymbol{x}) = \phi_E(\boldsymbol{x}) \left(\frac{1}{|\mathcal{U}| - 1} \sum_{\boldsymbol{x}^{(u)} \in \mathcal{U} \setminus \boldsymbol{x}} sim(\boldsymbol{x}, \boldsymbol{x}^{(u)}) \right)^{\beta}, \tag{3.4}$$

which implies that the informativeness of an unlabeled sample $\boldsymbol{x}$ is weighted by its average similarity to all the other samples in $\mathcal{U}$ with a parameter β that controls the relative importance of the density term. $|\mathcal{U}|$ in (3.4) denotes the number of samples in the candidate dataset $\mathcal{U}$. ϕ_E in (3.4) serves as the "base" informativeness measure which can be an uncertainty criterion or QBC, etc. Here, we choose entropy to be the "base" measure. Two commonly used similarity measures are the exponential Euclidean distance defined as

$$sim(\boldsymbol{x}, \boldsymbol{x}^{(u)}) = exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}^{(u)}||}{\sigma^2}\right),$$

and the cosine similarity defined as

$$sim(\boldsymbol{x}, \boldsymbol{x}^{(u)}) = \frac{\boldsymbol{x} \cdot \boldsymbol{x}^{(u)}}{||\boldsymbol{x}|| \times ||\boldsymbol{x}^{(u)}||}.$$

In our experiments, we find that the exponential Euclidean distance performs slightly better than the cosine distance. Thus the exponential Euclidean distance is employed in this work. However, ID has a drawback — samples with high information density may not be sufficiently representative for every class, even though they are representative for the entire unlabeled dataset. Thus such a query scheme may focus on querying data from only a few classes which have much higher density than the other classes, making it less competitive for classification and new class discovery.

3.2.2 Active Learning with New Class Discovery

New class (novelty) detection can be described as the identification of new or "unknown" data that a machine learning system was not aware of during training. The ability to detect new classes can have a significant impact in remote sensing applications, where the unlabeled data may contain information about objects that were not present in the labeled data.

Since DPMMs are capable of fitting data with an unknown number of mixtures, it is possible to differentiate between known and unknown classes by learning the clustering structure of the labeled and unlabeled data. If unlabeled data contains new classes, they will be assigned to new clusters that are different from those of the known classes. This has been done in [69] and [70]. Since our final goal is classification, we need to acquire labeled data of new classes after detecting them. Thus in this work, we embed the new class discovery problem in an active learning framework, aiming to improve classification performance with the least possible query effort for both known and unknown classes.

Recent work [71] for this problem based on DPMM is aimed at discovering rare classes in a pool of unlabeled images. The query strategy in [71] was constructed on the clustering result, which makes it efficient in class discovery. However, it is assumed that the unknown classes are rare and every class (including known and unknown) occupies one cluster, which are not valid assumptions in many remote sensing applications. First, although rare unknown classes can be commonly encountered in various applications, with remotely sensed image analysis over wide geographic areas, we can expect scenarios where unknown classes can also be prevalent (not rare). Second, for remotely sensed hyperspectral images, the properties of some classes may have large spatial variability, which makes their distribution possess a complex form (e.g., a multi-modal distribution). Thus, the initial training set of a known class may only contain samples from a specific region with low spatial variability while the unlabeled set may contain candidate samples from other unexplored regions with a different distribution than the samples in the training set. In this case, the samples in the unlabeled pool can form a new cluster by DPMM which is different from the cluster occupied by the samples of the same class in the training set. It is expected that these unlabeled samples are also useful for classification. The method we propose not only detects new classes but also unexplored areas of known classes which contains useful information for classification not currently available in the training dataset. It is important to note that not all new classes may be quickly and effectively detected. Note that there may be scenarios where some new classes may not be quickly detected (for e.g., if their spectra is very similar to some known class, they may merge into the same cluster).

3.3 The proposed framework

The flowchart of the proposed active learning system with a query strategy based on local information density is shown in Fig. 3.2. In the proposed framework, SELF is used for semi-supervised dimensionality reduction of the hyperspectral imagery. We expect SELF to be particularly appropriate for our problem, since the unsupervised (LPP) component in SELF will ensure that dominant information pertinent to missing classes is not lost in an otherwise supervised dimensionality reduction approach (with the intuition that not accounting for missing classes in a completely supervised feature reduction approach will result in sub-optimal transformations wherein information about missing classes may be potentially lost). Following this, the unlabeled data are clustered via DPMM and local density for each candidate is calculated based on the clustering result. At the same time, entropy for each candidate is computed from the posterior provided by the classifier. Queries are then made based on local information density which is computed from local density and entropy. Finally, the queried samples are added to the training set and removed from the candidate set.

In the following discussion, clustering is applied to data $\boldsymbol{X}$ in a lower dimensional subspace obtained from SELF. Compared to a static projection (such as LDA) as is commonly utilized in traditional classification, we have a dynamic projection which is refined after each step of active learning. As more data from both known and unknown classes are labeled and added to the training set, we will reach a better subspace given by SELF.

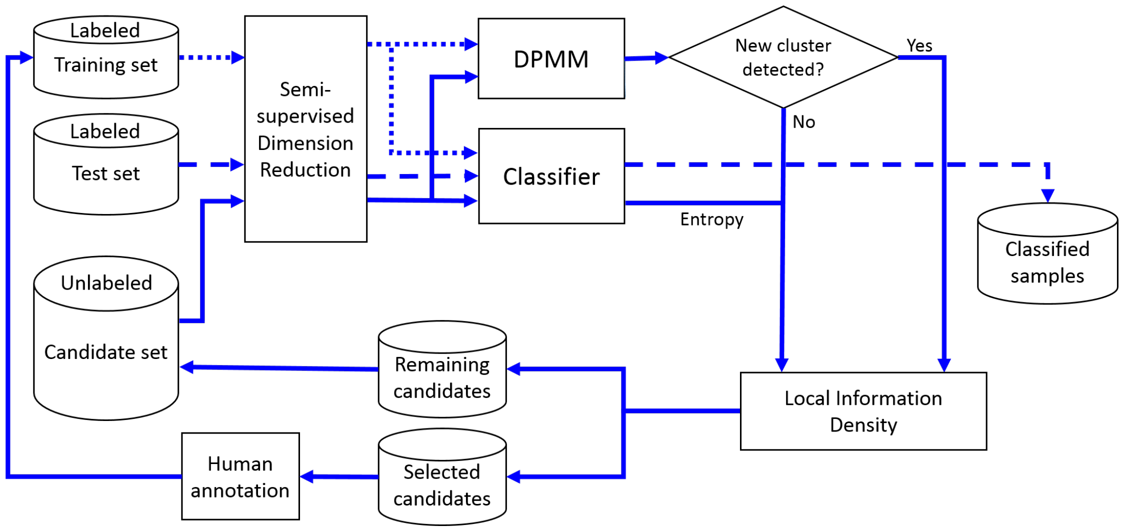


Figure 3.2: Flowchart of the proposed active learning framework.

3.3.1 Clustering based on DPMM

Gaussian mixture models (GMMs) can successfully capture the complex multi-modal statistical structure of various data such as remote sensing [18, 67] and audio data [80], etc. However, GMM assumes the number of Gaussian components to be known and uses expectation maximization (EM) algorithm for parameter inference. IGMM doesn't have this limitation, by assuming the number of Gaussian components to be infinity. Given the data, the number of components can be inferred automatically by Markov Chain Monte Carlo (MCMC) [44] sampling methods. IGMM, as defined in [15], is a special case of a Dirichlet process mixture, where the mixture components are assumed to be Gaussian distributed. In our framework, DPMM is used to cluster data in both the labeled set $\mathcal{L}$ and unlabeled set $\mathcal{U}$, as shown in Fig. 3.2. Readers can refer to Section 2.3 of Chapter 2 for more details about DPMM based clustering.

3.3.2 Query Strategy based on Local Information Density

The information density described in Section 3.2 has a drawback that samples with high information density may not be representative of every class even though they are representative of the entire unlabeled dataset. Hence, it may focus on querying data from only a few classes which have much higher density than the other classes, making it less competitive. In this work, a local information density is proposed here to address this problem, which aims to query samples that are both informative and representative of every class in $\mathcal{U}$. In the information density strategy, the density of $\boldsymbol{x}$ is calculated by its average similarity to all the other samples in the unlabeled data pool $\mathcal{U}$. In the proposed local information density strategy, the density of $\boldsymbol{x}$ is calculated by its average similarity to its neighboring samples and the neighborhood is defined by clusters generated by DPMM. Since both labeled samples in $\mathcal{L}$ and unlabeled samples in $\mathcal{U}$ are clustered together, some clusters generated by DPMM contain both labeled and unlabeled samples and the others only contain unlabeled samples. Those containing only unlabeled samples are considered to be new clusters. These samples in new clusters are either from unknown classes or known classes in unexplored regions of an image due to spatial variability.

Given the cluster assignments of all the unlabeled candidates in $\mathcal{U}$, the local density for an unlabeled candidate $\boldsymbol{x}$ with cluster label c can be computed as

$$\phi_{LD}(\boldsymbol{x}) = \frac{1}{|\mathcal{U}_c| - 1} \sum_{\boldsymbol{x}^{(u)} \in \mathcal{U}_c \setminus \boldsymbol{x}} sim(\boldsymbol{x}, \boldsymbol{x}^{(u)}), \tag{3.5}$$

where $\mathcal{U}_c$ represents all the unlabeled samples with the same cluster assignment c and $|\mathcal{U}_c|$ denotes the size of cluster c.

When new clusters emerge, in many applications, it is highly desirable to prioritize new clusters, to enable fast discovery. Hence, local densities are computed only for the samples in new clusters, while the local densities for all the other samples are set to be zero. If no new cluster is found, local densities will be calculated for every unlabeled sample in $\mathcal{U}$. Thus our local information density (LID) is formulated as

$$\phi_{LID}(\boldsymbol{x}) = \begin{cases} \phi_{LD}(\boldsymbol{x})\mathrm{I}_{new}(\boldsymbol{x}), & \text{if new cluster exists} \\ \phi_E(\boldsymbol{x})\left(\phi_{LD}(\boldsymbol{x})\right)^{\beta}, & \text{otherwise} \end{cases} \tag{3.6}$$

where the indicator function $\mathrm{I}_{new}(\boldsymbol{x})$ equals to 1 only when $\boldsymbol{x}$ comes from a new cluster and 0 otherwise. As in information density, we choose entropy $\phi_E(\cdot)$ to be the base measure. The parameter β which controls the relative importance of the local density term can simply be set to be 1 to make the two terms equally important. In our implementation, the value of β is set in an adaptive way such that the relative importance of the local density term $\phi_{LD}(\boldsymbol{x})$ gets lower as more and more samples are labeled. Thus we use a simple scheme to achieve this using

$$\beta = \frac{|\mathcal{U}|}{|\mathcal{U}| + |\mathcal{L}|}. \tag{3.7}$$

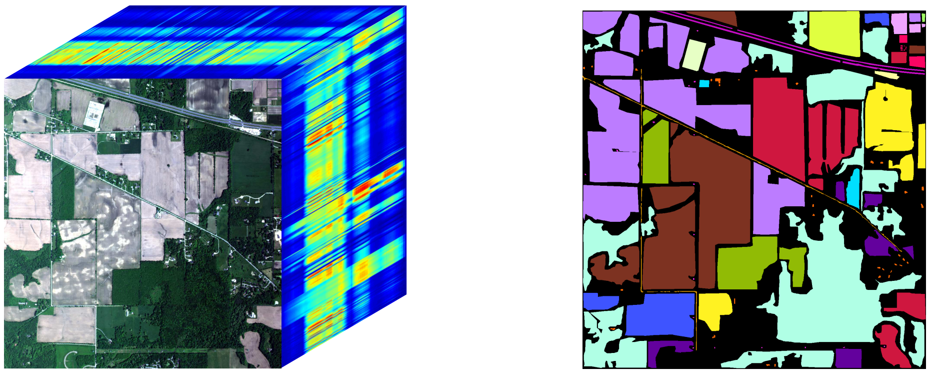
The information density (ID) based query strategy defined in Eq. (3.4) that combines the entropy with a global density term which is calculated by averaging the similarities between $\boldsymbol{x}$ and all the other samples in the unlabeled pool $\mathcal{U}$. It aims to query globally representative samples without emphasizing new classes. For our proposed LID based strategy queries, when new clusters are detected, it queries samples from the new clusters. When no new clusters exists, it queries uncertain samples with large local density computed in local clusters, which are locally representative and more beneficial for classification.

3.4 Experiments

3.4.1 Datasets

The first dataset used in this work, the University of Houston (UH) dataset which has been described in Section 2.6.

The second dataset, the Indian Pines hyperspectral image, was acquired using ProSpecTIR instrument in May 2010 over an agriculture area in Indiana, USA. The image has 1342×1287 spatial dimension with 2m spatial resolution. It consists of 180 spectral bands over the $400nm - 2500nm$ wavelength range. The 19 classes contain agriculture fields with different residue cover. Fig. 3.3 shows the true color image of the dataset with the corresponding ground truth, and Fig. 3.4 shows the mean spectral signatures (reflectance) of these 19 classes.



(a) (b)

⬛	🟦	🟫	🟩	🟪	🟥	🟨	🟪	🟥	🟨
Unlabelled Area	Corn-high	Corn-middle	Corn-low	Soybean-high	Soybean-middle	Soybean-low	Residues	Wheat	Hay
🟦	🟪	🟩	⬜	🟪	🟧	🟩	🟪	🟧	🟨
Grass/ Pasture	Grass	Wood-uniform	Wood-rugged	Highway	Local Road	Power Station	Power Towers	Houses/ Buildings	Urban Areas

(c)

Figure 3.3: The Indian Pines dataset: (a) True color image and (b) ground-truth with class names in (c).

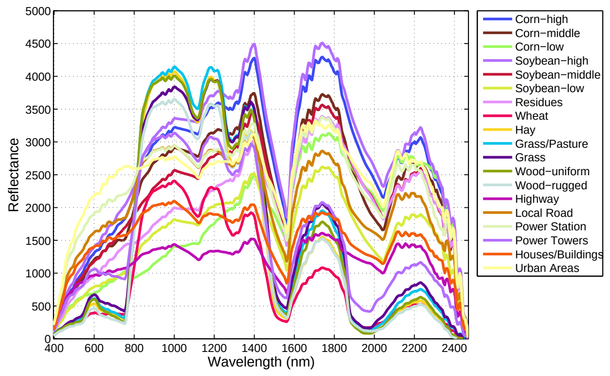


Figure 3.4: Mean spectral signatures of the 19 classes in Indian Pines dataset.

3.4.2 Experimental Setup

For dimension reduction, we use SELF to reduce the dimensionality of the two datasets to 15 and a constant $\gamma = 0.5$ in SELF is used to balance the effects of LFDA and LPP. To choose the number of features to retain after SELF, we use a cross validation method (leave one out) on the training data and we find 15 to be an optimal value for our datasets. For the active learning setup, 20 samples per class are randomly selected from the dataset to be the initial training set and 150 samples per class are randomly selected to be the candidate dataset. The initial number of clusters in DPMM is set to be the number of known classes and we take the sample mean and sample covariance of the training samples as the parameter initialization of each Gaussian component, which is a reasonable choice for 20 samples per cluster when the dimensionality of the data is 15. We run active learning for 80 iterations with a batch size $B = 5$ (number of samples queried at each iteration). For validation, 200 samples per class are chosen to be the test dataset on which the classifier is evaluated for each query step of active learning. For each query step of active learning, labeled and unlabeled samples are clustered via DPMM using 100 iterations of Gibbs sampling per run.

To simulate scenarios where unknown classes occur in the unlabeled candidate pool, randomly selected classes are removed from the initial training set. Typically, it is expected that for most practical applications, at most only a few classes are possibly unknown. Hence, in our experiments, for the UH dataset, we investigate three cases where the number of randomly removed classes ranges from 0 to 2 and for Indian Pines dataset, we remove 0 to 3 classes. When no class is removed from the initial training set, it reduces to a traditional active learning problem. A Bayes classifier is utilized for the classification part where the kernel density estimation (KDE) algorithm is used to estimate the probability density functions (pdf) of each class by placing a Gaussian kernel around each training sample. After each active learning query, KDE updates the pdf by adding one more Gaussian kernel around the new labeled sample. Note that our framework can be applied to any classifier (e.g., support vector machine or logistic regression and their variants). However, since we require posterior probabilities to calculate entropy, in this work, we use KDE due to its simplicity and computational efficiency.

Five active learning query strategies are implemented and compared in our experiments — random sampling (RS), breaking ties (BT), entropy-based uncertainty sampling (Entropy), information density (ID) and the proposed local information density (LID). Random sampling is implemented such that unlabeled samples are randomly selected from the candidate set $\mathcal{U}$ at each query step, labeled and added to the training set $\mathcal{L}$.

3.4.3 Results

3.4.3.1 Results for University of Houston Dataset

When all the classes in the candidate set are known, the problem reduces to a traditional active learning problem. We evaluate the performance of each query strategy by constructing learning curves that plot the overall accuracies across all classes as a function of the number of queries made, which is shown in Fig. 3.5 where each curve is averaged across 10 experiments with different randomly selected initial training set, candidate set and test set. The proposed LID based query strategy achieves the best performance for this traditional active learning setting in the sense that it starts with a higher rate of classification improvement. Also note that the overall accuracy for the uncertainty based strategies (BT and Entropy) and information density drops during the first few query steps, which illustrates the point that representative samples are more important for classification during the initial and crucial stage of active learning where we do not have enough labeled samples.

When we have one unknown class in the candidate set, 30 experiments are implemented

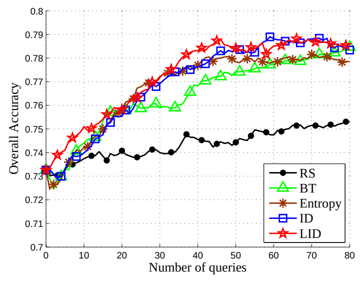


Figure 3.5: Learning curves of overall accuracy for all query strategies without unknown classes, UH dataset.

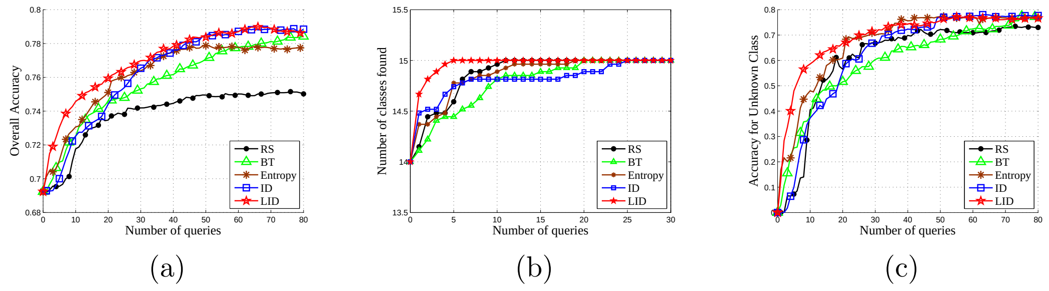
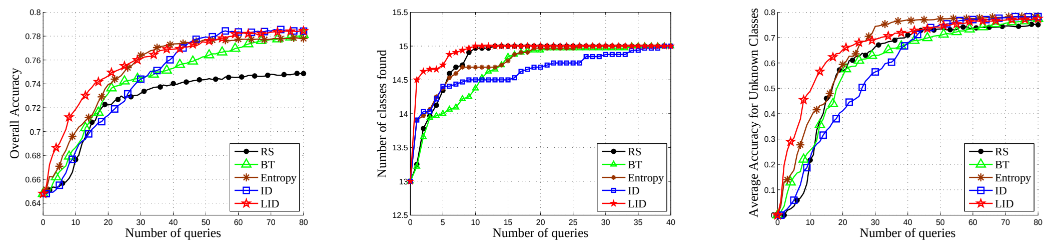


Figure 3.6: Learning curves with 1 unknown classes: (a) Overall accuracy. (b) Class discovery (number of classes found). (c) Accuracy for the unknown class, UH dataset.

and each of them has one randomly selected class removed from the starting training set $\mathcal{L}$. For the case of two unknown classes, 40 experiments are implemented with each of them having two randomly selected classes removed from the starting training set $\mathcal{L}$. For both cases, we plot the learning curves of the overall accuracies, class discovery (number of classes found) and the average accuracies for the corresponding unknown class in Fig. 3.6 and Fig. 3.7, which are computed by averaging the results across all the experiments. Again, in Fig. 3.6 (a) and Fig. 3.7 (a), we observe that LID achieves higher classification improvement at the beginning and reaches a higher accuracy upon convergence. More importantly, the proposed method discovers the new class much faster and significantly outperforms the other methods in classifying the new class, as shown in Fig. 3.6 (b), (c) and Fig. 3.7 (b) , (c). In more detail, in the first 20 steps of active query, LID based method achieves an improvement of 10% $\sim$ 40% on the accuracy of the new class than the other baseline methods and finally



(a) (b) (c)

Figure 3.7: Learning curves with 2 unknown classes: (a) Overall accuracy. (b) Class discovery (number of classes found). (c) Average accuracy for the unknown classes, UH dataset.

they all converge to the same level.

As discussed in Section 3.2 and 3.3, both ID and LID choose entropy as the base measure in their query strategies, LID achieves improvements over entropy but ID degrades the performance of entropy. The reason for this should be that ID only queries uncertain samples with high global density, which may not be representative for each class. On the other hand, LID queries uncertain samples with high local density, which help the KDE based Bayes classifier better estimate the distributions of each class and achieves better classification results.

Another phenomenon we can observe from these figures is that uncertainty and information density based query strategies (BT, Entropy, ID) are performing better than random sampling in overall classification for all cases in Fig. 3.5 to 3.7. However, as shown in Fig. 3.6 (b) and Fig. 3.7 (b), they are not always better than random sampling in detecting unknown classes due to the fact that uncertainty and information density based query strategies may treat the unlabeled samples of unknown classes as less informative than some samples of known classes while random sampling treat every unlabeled sample equally. Thus random sampling can occasionally pick up the unknown classes faster than uncertainty and information density based query strategies. However, the proposed LID based approach still outperforms all other baseline methods in detecting and classifying the unknown classes.

Since the performance of proposed query strategy depends on the clustering quality of the DPMM, we show the clustering quality using the normalized mutual information (NMI) [81], a standard method of measuring clustering quality, across all the active learning steps in Fig. 3.8. As illustrated, the clustering performance becomes better as more samples are queried because a more accurate parameter initialization and a better subspace from SELF results, when the training sample size grows.

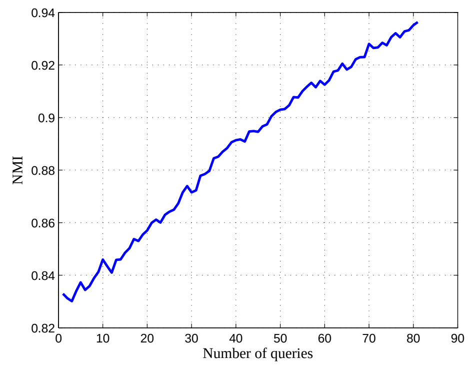


Figure 3.8: Learning curves of the normalized mutual information for the cluster results given by DPMM using UH data set with one unknown class.

3.4.3.2 Results for Indian Pines Dataset

We perform similar experiments for the Indian Pines hyperspectral dataset and obtained similar results shown in Fig. 3.9 to 3.12 where the number of unknown classes increases from 0 to 3. In all cases, the proposed LID based query strategy achieved the best overall classification accuracy due to the reason that the locally representative samples it selected are more useful for classification when we do not have enough training samples, no matter whether new classes exist or not in the unlabeled candidate pool. When new classes emerge, the proposed method will first focus on querying samples which are assigned to new clusters by the Dirichlet process mixture model and are more likely to be new classes. Thus it significantly outperforms the other methods in discovering and classifying the unknown classes. In particular, in the first 20 steps of active query, LID based method achieves an improvement of about $10\% - 30\%$ in the accuracy of the missing classes, compared to the other methods. We should keep in mind that the new clusters found by the DPMM do not have to be new classes especially in remote sensing applications. New clusters usually emerge at spatial locations which are different from the labeled data, so they can either be new classes or correspond to unrepresented areas of known classes due to spatial variability of remote sensing images, which results in multimodal distributions of certain classes.

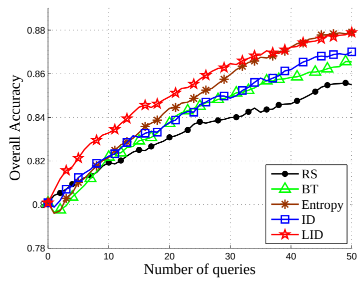
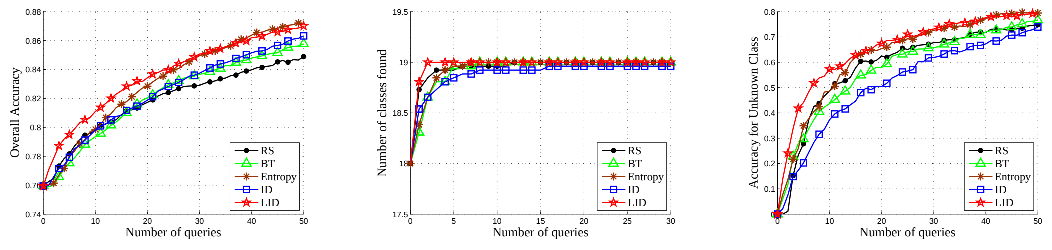


Figure 3.9: Learning curves of overall accuracy for all query strategies without unknown classes, Indian Pines dataset.

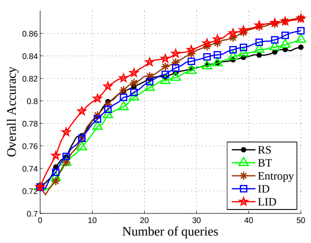


(a) (b) (c)

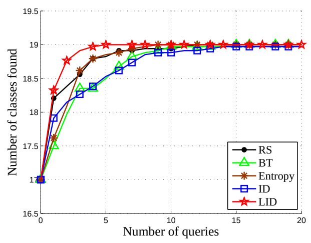
Figure 3.10: Learning curves with 1 unknown classes: (a) Overall accuracy. (b) Class discovery (number of classes found). (c) Accuracy for the unknown class, Indian Pines dataset.

3.4.3.3 Visualizing the Detection of New Classes

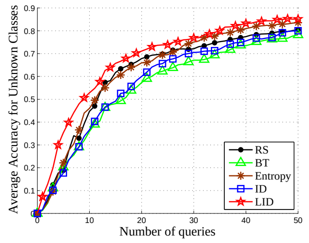
To demonstrate the clustering performance on an image, we crop a region from the UH which contains three classes as shown in Fig. 3.13 (a): class 2 ("Grass-stressed"), class



(a)

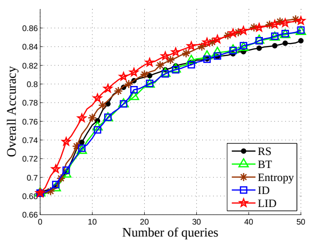


(b)

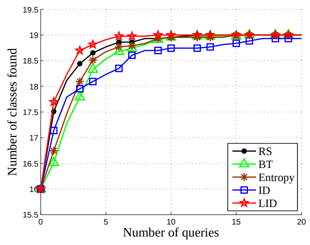


(c)

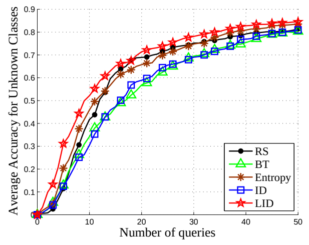
Figure 3.11: Learning curves with 2 unknown classes: (a) Overall accuracy. (b) Class discovery (number of classes found). (c) Average accuracy for the unknown classes, Indian Pines dataset.



(a)



(b)



(c)

Figure 3.12: Learning curves with 3 unknown classes: (a) Overall accuracy. (b) Class discovery (number of classes found). (c) Average accuracy for the unknown classes, Indian Pines dataset.

3 ("Grass-synthetic") and class 15 ("Running Track"). We remove the three classes one at a time (resulting in different "unknown" classes each time). When class 15 is removed, clustering result (including known and unknown classes) and detected new class are shown in Fig. 3.13 (b) and (e). Similarly, results when class 3 is removed are shown in Fig. 3.13 (c) and (f); results when class 2 is removed are shown in Fig. 3.13 (d) and (g). As we can see in all cases, different classes are separated into different clusters (Fig. 3.13 (b) to (d)) and the unknown class is detected as a new cluster successfully (Fig. 3.13 (b) to (d)). Note that if there are isolated small clusters comprising of very few pixels, they are likely to belong to outliers, and we ignore them.

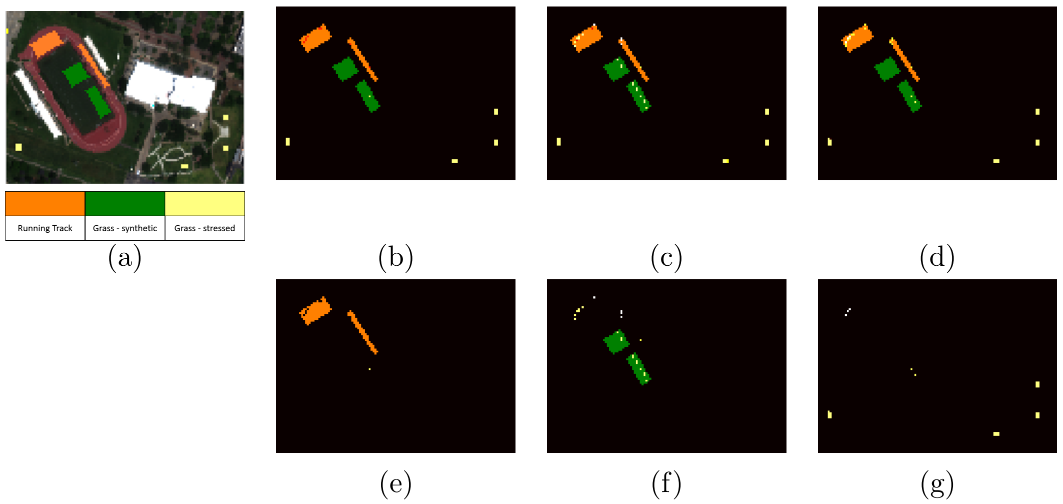


Figure 3.13: Visualizing the detection of new classes.

3.5 Conclusions

The new active learning paradigm introduced in this work simultaneously improves classification performance and discovers unknown (missing) classes with the least effort of labeling/annotating — missing or unknown classes are commonly encountered in machine learning tasks, particularly image analysis of remotely sensed data where it is of great value to find and label new classes and unexplored regions of existing classes (that may exhibit variability in the spectral reflectance characteristics). The framework achieves this goal by employing local information density as the query strategy where the local density is obtained via DPMM based clustering. The experimental results have shown that the proposed method provides significantly better performance than the other commonly used active learning frameworks both in classification accuracy and detection speed of unknown classes. To the best of our knowledge, this is the first work that demonstrates a successful active learning paradigm that seeks out discovery of unseen classes. This has profound benefits for a variety of applications, including image analysis of big geospatial data cubes, where it is often not possible to have every class on the ground represented in the initial training library. Such an approach is also expected to be particularly advantageous to geospatial images where class variability (e.g., due to significant illumination differences such as classes under shadows) can lead to multi-modal distributions which are not effectively accounted for in traditional labeled training libraries — the proposed framework can assist with enhancing the library by ensuring that sources of variability that express themselves as new clusters are systematically accounted for during the creation of a training library.

Chapter 4

Deep Learning for Remote Sensed Image Classification

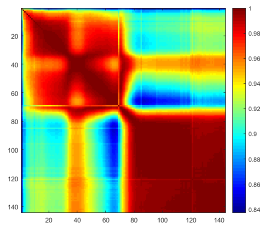
4.1 Introduction

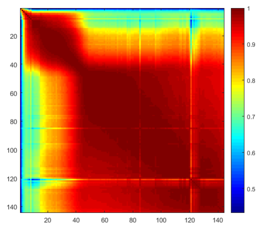
In the last decade, with the advances of computing power of computers and the availability of large scale dataset, deep learning [82] techniques such as deep belief networks (DBN), deep convolutional neural networks (CNN) and deep recurrent neural networks (RNN), have gained great success in a variety of machine learning tasks such as computer vision, speech recognition, and natural language processing etc. For example, since first proposed in [83], CNN has been widely used for various computer vision tasks such as large-scale detection and classification of object categories [20–24], and speech recognition [84]. RNNs, another important branch of deep neural networks family, were mainly designed for sequence modeling. The long short term memory (LSTM) [85, 86] network is a special type of RNN, which is able to capture very long term dependencies embedded in sequence data. Both the regular RNN and the LSTM networks have been successfully used for time series data analysis such as speech recognition [25–28], machine translation [29–31], etc.

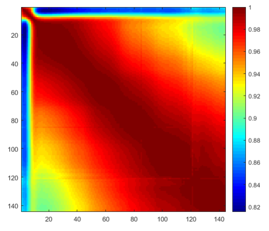
Hyperspectral data, which captures spectral information of objects over a wide range of the electromagnetic spectrum, has been played a key role in remote sensed data analysis [87]. With rich spectral information, hyperspectral image data has been used for target detection

[88, 89], land cover classification [90–92], space surveillance [93], environmental science [94]. Modern advanced hyperspectral sensors are able to collect hyperspetral images of very high resolution, from which we can a large number of labeled pixels for training a deep neural networks. Recently, deep learning techniques have been introduced into remote sensing community especially for hyperspectral data classification [32–37] and achieved achieved state-of-the-art performance. [32] proposed to use stacked autoencoder to extract deep features from hyperspectral data. [33] used one dimensional CNN to extract spectral features for hyperspectral data. [34, 36] employed two dimensional CNN to extract features for hyperspectral data. [35] trained deep convolutional networks in an unsupervised greedy layerwise fashion [95] to learn the network filters.

However, all the existing work such as [32, 35, 36] that employed deep learning techniques for hyperspectral data classification all treat the hyperspectral data samples as high dimensional input vectors to the network and try to extract deep features from it. In this work, we try to analyze hyperspectral data in a sequential point of view, i.e., each hyperspectral sample is seen as a data sequence, which can be modeled by recurrent neural networks. As the sequence model, RNN assumes that the current output of a sequence depends not only on the current input but also the previous outputs. In other words, RNNs are suited for sequences where there are dependencies between different time steps. Since hyperspectral data is densely sampled from the full spectrum of some material, it is expected to have dependencies between different spectral bands. First, it is easy to observe that for any material, the adjacent spectral bands tend to have very close values, which means that adjacent spectral bands are highly dependent on each other. In addition, some materials also demonstrate long-term dependency between non-adjacent spectral bands. We show these dependencies as correlation coefficients between each pair of spectral bands for some classes from two hyperspectral datasets in Fig. 4.1 and 4.2. The block diagonal structure in each figure indicates short-term spectral dependencies. Some classes clearly demonstrate long-term spectral dependencies such as Fig. 4.1 (d), Fig. 4.2 (b), (c) and (d).


(a) Class 1


(b) Class 5


(c) Class 8

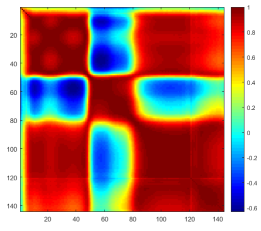
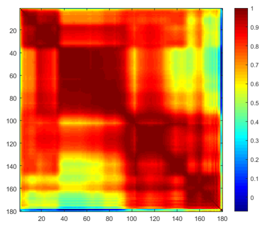
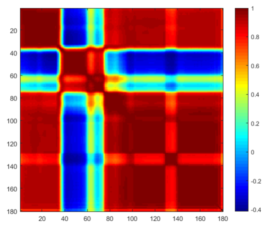

(d) Class 14

Figure 4.1: Correlation coefficient matrix between all spectral bands for 4 classed in the "University of Houston" dataset.

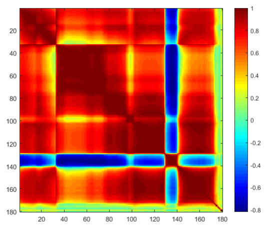
In this work, our goal is to investigate RNNs for hyperspectral data classification. Although CNNs are also able to capture some dependencies in the input sequence data, it only model local dependencies since the convolutional filters often have short length. Thus recurrent networks is a better method for modeling sequence dependency. In addition, we also propose to use a convolutional neural network (CRNN) [96, 97] which is composed of a few convolutional layers followed by a few recurrent layers. The motivation to use convolutional layers (and pooling layers) before recurrent layers is that convolutional layers can first extract middle-level, locally invariant features from the input sequence. Pooling layers make the sequence shorter by subsampling, which will accelerate the backpropagation through recurrent layer since the computation complexity of recurrent layers grows linearly with respect to the length of the input sequence. Thus recurrent layers can then



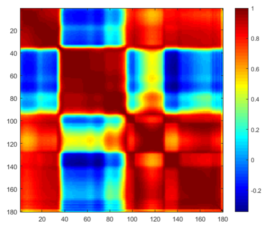
(a) Class 1



(b) Class 8



(c) Class 9



(d) Class 15

Figure 4.2: Correlation coefficient matrix between all spectral bands for 4 classed in the "Indian Pines" dataset.

extract the spectral dependencies more efficiently from the middle-level features provided by convolutional layers. Another benefit of RNN is that, unlike CNN, it does not require the input sequence to have the a fixed length. Although each pixel in a hyperspectral image has the same length of spectrum in our problem, the method can still be extended to handle problems where input sequences may have variable lengths.

In remote sensing applications, sensor fusion [98, 99] refers to the process of combining data from multiple sensors to produce a dataset that contains more detailed information than each of the individual sources. A sensor by itself provides a unique perspective and is designed for a specific purpose. Thus combining complementary data modalities from multiple sensors can generate more accurate interpretation for the features of the objects in the scene.

Hyperspectral imagery (HSI), collected by hyperspectral sensors, contains spectral information across the electromagnetic spectrum for each pixel in the image of a scene. With a wealth of spectral information, HSI has been playing a key role in remote sensed data analysis [87] such as land cover classification [92, 100, 101] and anomaly detection [102], etc. Light detection and ranging (LiDAR) data, another widely used remote sensing modality, provides geometrical information on the 3D structure of the surface of the ground, which can also be used for land cover classification such as urban area and tree classification [103–105]. Due to the complementary information provided by HSI and LiDAR data, combining them in a sensor fusion system can produce more discriminative features for land cover classification. For instance, in an urban classification scenario, the materials used in parking lots and roofs of buildings may have similar hyperspectral measurements which makes them difficult to discriminate using just HSI. However, just by taking elevation of the surface of the observed scene into account, they can be discriminated easily. From the LiDAR point cloud, we can further generate a LiDAR pseudo waveform (LPW) [106] for each pixel on the hyperspectral image, which results in a LPW data cube with the same spatial dimensions as the HSI data cube. In this work, the LPW data is produced using a dense point cloud from a discrete return LiDAR system. This was done via binning the elevation information into fine slices, and estimating the average intensity in each bin, within each voxel. The voxel was the same spatial resolution as the HSI. This LPW data provide not only the elevation information but also the geometry of the objects on the ground. Thus by combining the registered HSI and LPW data, we are able to create more accurate land cover classification.

In general, there are three common ways of sensor fusion. Data (pixel) fusion combines raw pixel values from multiple source images into a single image which will then be used for analysis. Decision fusion fuses the results of multiple algorithms on different sources to yield a final decision. Feature fusion extracts different features from multiple data sources to yield combined feature maps for subsequent analysis such as classification. In this chapter, we will investigate all these three sensor fusion methods using deep learning techniques. Especially we will focus on the deep feature fusion method which combined the features extracted by deep neural networks for HSI and LPW data. However, unlike the traditional feature fusion method, the feature extraction for different data sources are not independent in our proposed method because the networks for different sources are trained simultaneously and they benefit from each other.

The remainder of this Chapter is organized as follows: Section 4.2 talks about different deep neural networks for hyperspectral data classification: we provide a basic overview for CNNs and how they have been used for hyperspectral data classification, introduce RNNs for hyperspectral data classification, and describe how to combine CNN with RNN to get the proposed method CRNN for hyperspectral data classification. A spatial constraint based on decision fusion is described in section 4.3 for spectral-spatial classification. Three sensor fusion methods including the proposed deep sensor fusion method are described in Section 4.4. The detail of the experiments conducted for validating the proposed methods is given in section 4.5 which includes descriptions of the datasets, experimental setup and results. Section 4.6 summarizes the key ideas and experimental results in this work and provides concluding remarks.

4.2 Convolutional and Recurrent Neural Networks for Hyperspectral Data Classification

In this section, we will go through different deep neural networks for hyperspectral data classification. In section 4.2.1, we briefly explain the basics of CNN and how they have been used for hyperspectral data classification. In section 4.2.2, we introduce RNN for hyperspectral data classification. In section 4.2.3, we talk about how to construct a CRNN model by combining CNN with RNN and use it for hyperspectral data classification.

4.2.1 CNN

As in [33], 1-D CNNs have been successfully used for hyperspectral image pixel-level classification. Moreover, [34, 36] made use of 2-D CNNs for classification by taking a neighborhood window of size $w \times w$ (where w usually takes values about 20) around each labeled pixel and treat the whole window as a training sample, which is basically image-level classification instead of pixel-level classification. And this 2-D CNN framework will need a large number of labeled pixels to generate enough 2-D neighborhood regions to train a deep CNN. However, we usually don't have enough labeled pixels on a remotely sensed hyperspectral image to generate so many neighborhood regions for training a deep network. Another concern is that having a fixed size neighborhood cannot guarantee each neighborhood region contains only one class of object. Since different classes tend to have different spatial sizes, so the optimal neighborhood size should be class specific. Thus we think 1-D CNN is more practical for remotely sensed hyperspectral image classification and we will only focus on 1-D CNNs in the following discussion.

A graphical illustration of the 1-D CNN we used in this work is shown in Fig. 4.3 where a hyperspectral vector is fed to the input layer and then propagated through several successive convolutional and pooling layers for feature extraction. Each convolutional layer has multiple 1-D convolutional filters (kernels). The size of the kernel is a hyperparameter and data dependent. The pooling layers are used for subsampling to reduce the dimensionality of the network, which can help reduce computation and control overfitting.

Let's denote an input hyperspectral vector as $\boldsymbol{x} = (x_1, x_2, ..., x_T) \in \mathbb{R}^T$ where T is the length of the input vector. In the first convolutional layer, a set of d filters $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_d\}$

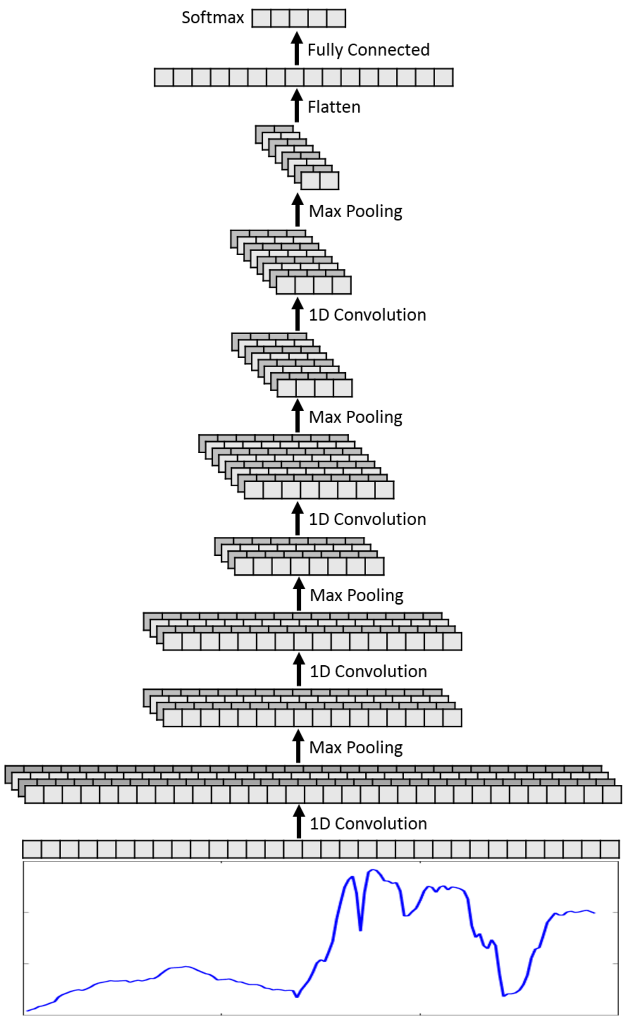


Figure 4.3: Architecture of the CNN for hyperspectral data classification.

of receptive field size r, are applied to the input vector to get the feature map using

$$\boldsymbol{F} = (\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_T) = g(\boldsymbol{x} \star \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_d\}), \tag{4.1}$$

where $\boldsymbol{f}_t \in \mathbb{R}^d$ and g is a nonlinear activation function such as tanh or a rectified linear unit (ReLU). ReLU is defined as $f(x) = max(0, x)$ which has become the most used activation function in CNNs, and we also choose to use it in this work.

The max pooling layer subsample every depth slice of the input by *max* operation. The most common form is a pooling layer with filters of length 2 applied with a stride of 2. After applying the pooling layer, the depth (dimension) remains unchanged but the length is reduced by half.

Finally the extracted high-level features will be flattened to be a fixed-dimensional vector which is then fully connected to the output layer for classification where we have a softmax activation function to compute the predictive probabilities for all the categories. This is done by

$$p(y = k|\boldsymbol{x}) = \frac{exp(\boldsymbol{w}_k^\top \boldsymbol{x} + b_k)}{\sum_{k'=1}^{K} exp(\boldsymbol{w}_{k'}^\top \boldsymbol{x} + b_{k'})}, \tag{4.2}$$

where $\boldsymbol{w}_k$'s and b_k's are the weight and bias vectors, and there are K categories.

Training a neural network is to find the best parameters (weights of the network) to minimize the loss function, which in a classification task measures the compatibility between a prediction (e.g., the class scores in classification) and the ground truth label. The loss takes the form of an average over the losses for every training example: $L = \frac{1}{N}\sum_{i=1}^{N} L_i$ where N is the number of samples and L_i is the loss for sample i. For output layer with softmax activation, the cross-entropy loss (also known as negative log likelihood) is most widely used and is defined as

$$L_i = -log(p(y_i|\boldsymbol{x}_i)). \tag{4.3}$$

The network is trained with stochastic gradient descent (SGD) and gradients are calculated by the back-propagation algorithm. A mini-batch strategy is utilized in our implementation to reduce loss fluctuation, so the gradients are calculated with respect to mini-batches. The algorithm will be running iteratively until the loss converges when the change of training and validation loss is below some threshold.

4.2.2 RNN

Recurrent neural networks which consists of successive recurrent layers are sequential models which map a sequence to another sequence. RNN have a strong capability of capturing contextual information within a sequence. The contextual cues are stable and useful for classifying hyperspectral data. What's more, RNN is able to operate on sequences of arbitrary lengths, although this advantage is not utilized in this work. But it's worth noting that our work can be extended to handle the problem where the input sequences have variable lengths.

The structure of a basic RNN with one recurrent layer is illustrated in Fig. 4.4 where we have a sequence of vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$ as input, $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T\}$ is a sequence of hidden sates, and $\{\boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_T\}$ is a sequence of outputs.

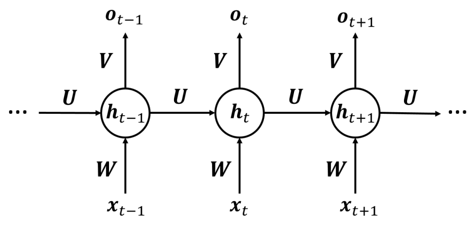


Figure 4.4: Structure of a basic recurrent layer.

A recurrent layer has a recursive function f which takes as input one input vector $\boldsymbol{x}_t$ and the previous hidden state $\boldsymbol{h}_{t-1}$, and returns the new hidden state as

$$\boldsymbol{h}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = tanh(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1}) \tag{4.4}$$

and the outputs are calculates as

$$\boldsymbol{o}_t = softmax(\boldsymbol{V}\boldsymbol{h}_t), \tag{4.5}$$

where $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{V}$ are the weight matrices which are shared across all steps, and activation function *tanh* is the hyperbolic tangent function. This recursive function however is known to suffer from the problem of vanishing gradient [107] for long input sequences such as the speech signal or text document, which makes it difficult to learn to long-term dependencies. To overcome this problem, the Long Short Term Memory (LSTM) RNN [85, 86] was introduced which uses a more complicated function that learns to control the flow of information, allowing the recurrent layer to capture long-term dependencies more easily. The structure of a basic LSTM unit in an RNN is illustrated in Fig. 4.5.

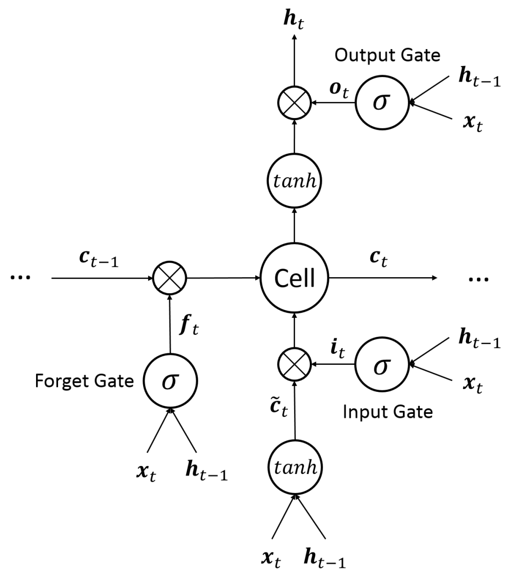


Figure 4.5: Structure of a basic recurrent layer.

The LSTM unit consists of four sub-units: input gate, output gate, forget gate and new memory, which are computed as follows:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1}), \tag{4.6}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1}), \tag{4.7}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1}), \text{ and} \tag{4.8}$$

$$\tilde{\boldsymbol{c}}_t = tanh(\boldsymbol{W}^{(c)}\boldsymbol{x}_t + \boldsymbol{U}^{(c)}\boldsymbol{h}_{t-1}), \tag{4.9}$$

where activation function σ and $tanh$ are logistic sigmoid and hyperbolic tangent functions respectively. Based on these, the LSTM unit then computes the memory cell and output as

$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \text{ and} \tag{4.10}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t), \tag{4.11}$$

where the point-wise multiplication of two vectors is denoted with $\odot$.

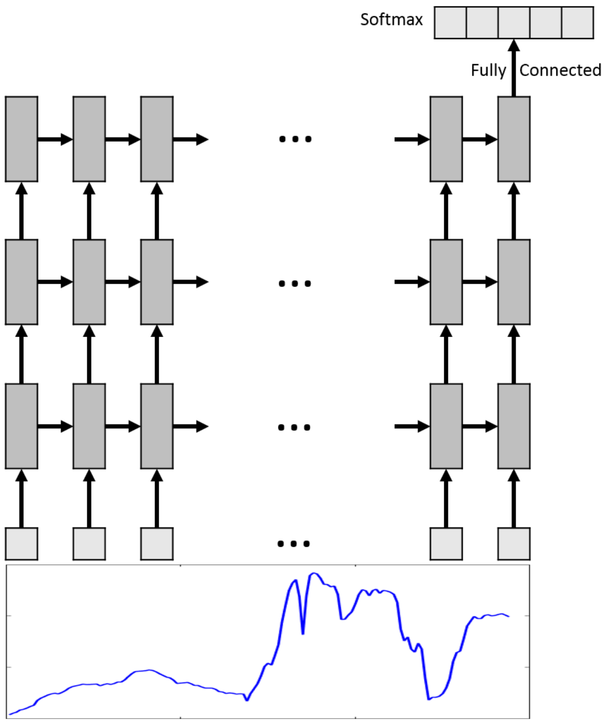


Figure 4.6: Architecture of the RNN for hyperspectral data classification.

A graphical illustration of the RNN framework (regular RNN or LSTM RNN) we used in this work is shown in Fig. 4.6 where the input hyperspectral data $\boldsymbol{x} = (x_1, x_2, ..., x_T)$ is viewed as a sequence of 1-D vectors, which is propagated through several recurrent layers to extract deep features. The hidden states of the last recurrent layer are a sequence of high-level features. As indicated in Fig. 4.6, we only take the last hidden state of the last recurrent layer as the input to the classification layer since the last hidden state should already contain all the useful contextual information in previous time steps. Like in CNN, the softmax activation function is applied at the output layer and the loss function is formulated as the cross-entropy. Training is done by mini-batch gradient descent. But gradients of the loss function are calculated by the back-Propagation through time (BPTT) algorithm [108].

4.2.3 CRNN

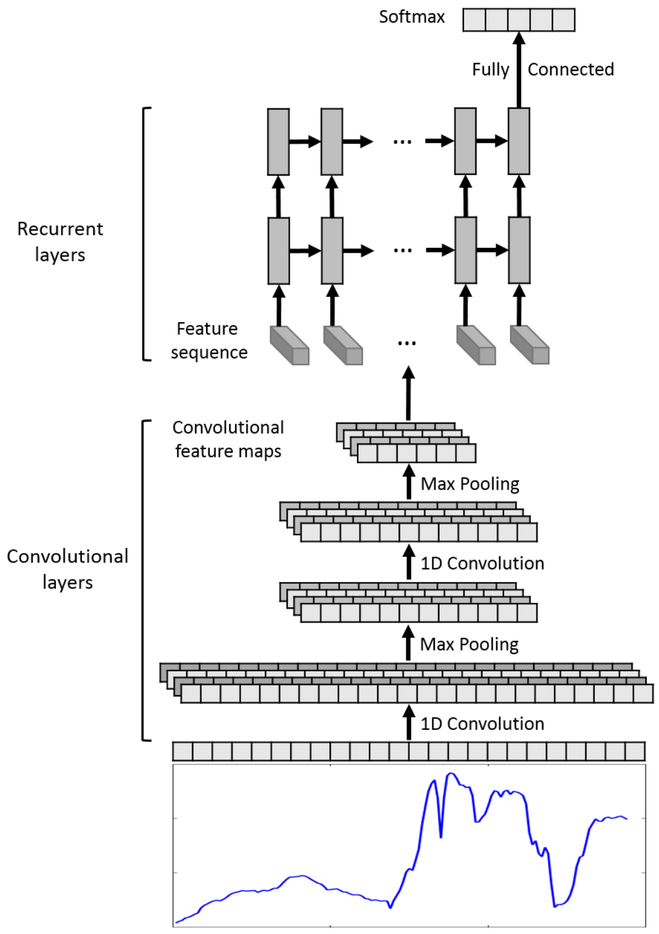


Figure 4.7: Architecture of the CRNN for hyperspectral data classification.

The CRNN is a hybrid of convolutional and recurrent neural networks. It's composed of several convolutional (and pooling) layers followed by a few recurrent layers, as indicated in the graphical illustration of CRNN used in this work in Fig. 5.1. CRNN has the advantages of both convolutional and recurrent networks. First, the convolutional layers are able to efficiently extract middle-level, abstract and locally invariant features from the input sequence. The pooling layers help reduce computation and control overfitting. Second, the recurrent layers extract contextual information from the feature sequence generated by the previous convolutional layers. Contextual information captures the dependencies between different bands in the hyperspectral sequence, which is more stable and useful for classification. Third, recurrent layers can handle variable-length input sequences, though we are not making using of this benefit in this work.

For the recurrent layers, we can either use regular recurrent function or more complicated ones like LSTM which can capture very long dependencies more efficiently. However, since the length of input hyperspectral sequences are not very long (usually below 200) and the pooling layers additionally reduce the length a lot (usually below 50), we won't have long-term dependency and vanishing gradient problems in most cases. Thus regular RNN should work as well as LSTM networks, which has been verified in our experiments.

Finally, as in RNN, the last hidden state of the last recurrent layer will be fully connected to the classification layer where a softmax activation function is applied. For training, as in CNN and RNN, the loss function is chosen as cross-entropy and mini-batch gradient descent is used to find the best parameters of the network. The gradients in the convolutional layers are calculated by the back-propagation algorithm and gradients in the recurrent layers are calculated by the back-propagation through time (BPTT) algorithm [108].

4.3 Spatial Constraint by Decision Fusion

The neural networks described in Section 4.2 all extract features from spectral information for each pixel. In order to further improve the classification performance, we integrate the spatial constraint by linear opinion pools (LOP) [106, 109] based on the fact that, in pixel-based image classification, spatially neighboring pixels tend to have similar categories. The spatial constraint encourages piecewise smooth segmentation of images by smoothing out noisy predictions due to noisy data or outliers.

Based on the predictive probabilities computed by the output layer (softmax) of the neural networks, LOP-based decision fusion calculate the posterior probability for each pixel $\boldsymbol{x}_i$ as a weighted sum of the posterior probabilities of the spatial neighbors of that pixel defined as

$$p(y_i|\boldsymbol{x}_i) = \sum_{j \in \mathcal{N}_i} \omega_j p(y_j|\boldsymbol{x}_j), \tag{4.12}$$

where $\mathcal{N}_i$ are the spatial neighbors of pixel $\boldsymbol{x}_i$ and ω_j's are weights for each neighbor that satisfy $\sum_{j \in \mathcal{N}_i} \omega_j = 1$. For simplicity, we use uniform weights for neighbors in our implementation.

4.4 Deep Sensor Fusion of Hyperspectral Imagery and LiDAR Data for Land Cover Classification

The complementary information provided by the HSI and LPW data sources is illustrated in Fig. 4.8 which shows the HSI and LPW signatures for a few pixels of different classes. For example, roads and roofs of commercial buildings, trees and grasses have similar hyperspectral signatures, but their LPW signatures are quite different from each other. On the other hand, the grasses and roads have similar LPW signatures but very different hyperspectral measurements. Another important property of LiDAR data is that they are not sensitive to the atmosphere's condition since LiDAR sensors are active sensors. However, HSI data can be badly affected by the atmosphere's condition. For example, as shown in Fig. 4.9, the same object shows different hyperspectral signatures in the sun and in the cloud shadow, but the LPW data stay very consistent. Therefore, it's meaningful to combine the different information provided by the HSI and LPW data for land cover classification in a sensor fusion system.

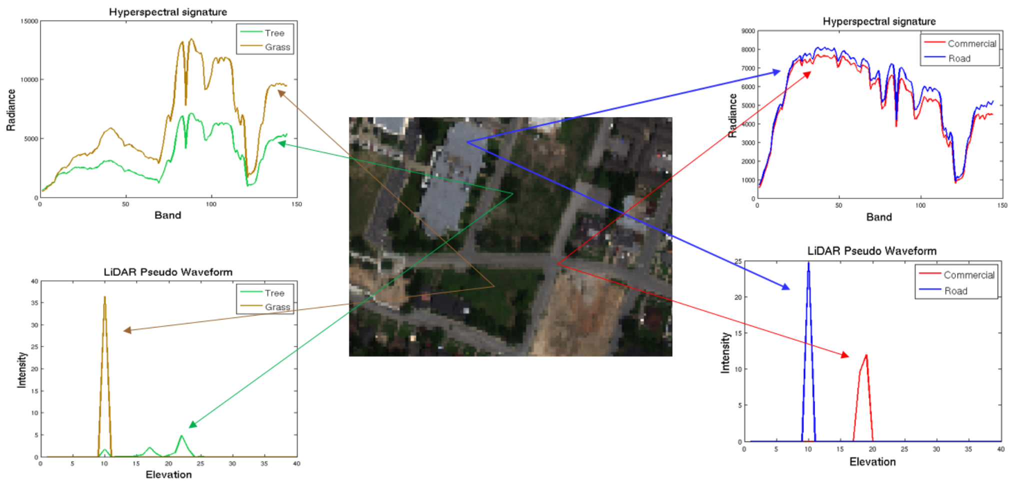


Figure 4.8: Examples of HSI and LPW signatures.

As mentioned in Section **??**, there are three common ways of sensor fusion for land cover classification: data fusion, decision fusion and feature fusion. The proposed deep feature fusion network has two base networks corresponding to the two data sources. A CRNN is used for the HSI source in the same way as [110] since it is able to extract robust contextual features for classification. As shown in Fig. 4.8, LPW data for urban areas usually demonstrate strong local structures, i.e., the pseudo waveform usually has only one peak for objects with smooth surface and may have a few peaks for other objects such as trees. Thus we choose to use a CNN to extract features for the LPW data source. Readers can refer to Chapter 4 for more details about the CNN [33, 110] and CRNN [110].

4.4.1 Data Fusion

Data fusion [111] concatenates different data sources to form a longer input vector. Let's denote hyperspectral data as $\boldsymbol{x}_H \in \mathbb{R}^{d1}$, and LiDAR pseudo-waveform data as $\boldsymbol{x}_L \in \mathbb{R}^{d2}$, then data fusion concatenate them to form a new input vector as $\boldsymbol{x}_C = [\boldsymbol{x}_H, \boldsymbol{x}_L] \in \mathbb{R}^{(d1+d2)}$. Then we can design a deep neural network (such as an CNN or CRNN) to classify the concatenated data $\boldsymbol{x}$. The flowchart for this sensor fusion system is illustrated in Fig. 4.10.

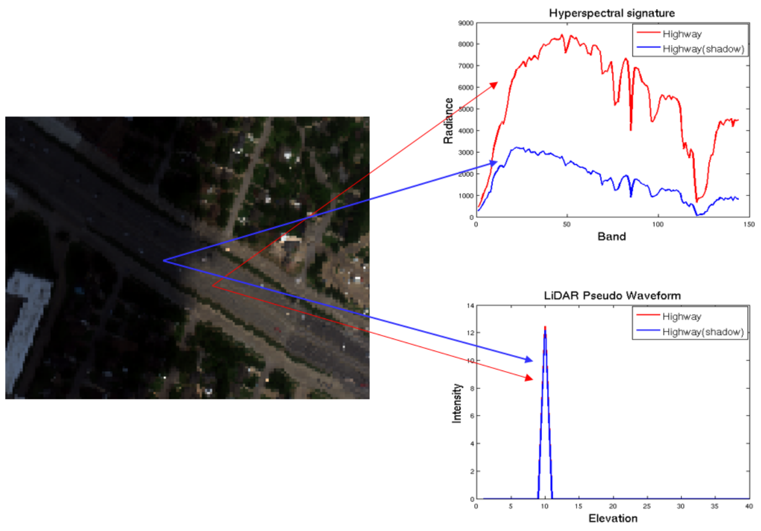


Figure 4.9: HSI and LPW signatures for pixels in the cloud shadow and in the sun.

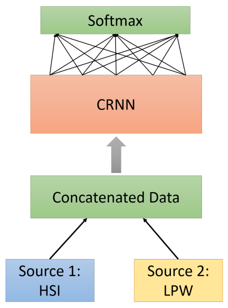


Figure 4.10: Sensor fusion based on data fusion and deep neural networks.

4.4.2 Decision Fusion

Decision fusion [112] averages the predictions from two base neural networks trained on the two data sources independently. In our case, a CRNN is trained for the HSI data and a CNN is trained for the LPW data, as shown in the flowchart in Fig. 4.11. When predicting the label for a new input data $\{\boldsymbol{x}_H, \boldsymbol{x}_L\}$, we first compute the predictive probabilities for both networks: $P_H(y = k|\boldsymbol{x}_H)$ and $P_L(y = k|\boldsymbol{x}_L)$. The final predictive probability will be calculated using decision fusion techniques such as Linear Opinion Pooling (LOP):

$$P(y = k|\boldsymbol{x}_H, \boldsymbol{x}_L) = \omega_1 P_H(y = k|\boldsymbol{x}_H) + \omega_2 P_L(y = k|\boldsymbol{x}_L), \tag{4.13}$$

where the positive weights ω_1 and ω_2 satisfy $\omega_1 + \omega_2 = 1$.

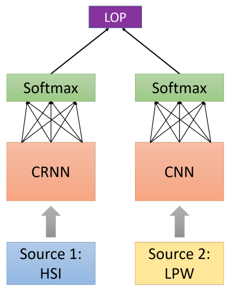


Figure 4.11: Sensor fusion based on decision fusion and deep neural networks.

4.4.3 Deep Feature Fusion

In a deep feature fusion system [111, 113], we have two base networks designed for two data sources: a CRNN for HSI and a CNN for LPW. The deep features extracted by the base networks are concatenated before being fed to the output layer for classification. The cross entropy loss function for the whole network is calculated based on this concatenated features. In this way, the CRNN for HSI and the CNN for LPW can be trained simultaneously using gradient descent and they can benefit from each other, i.e., the former helps to train the latter and meanwhile the latter also helps to train the former. The flowchart for this sensor fusion system is illustrated in Fig. 4.11.

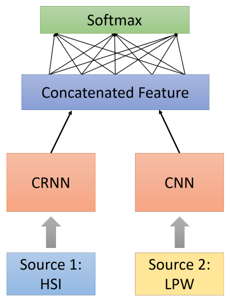


Figure 4.12: Sensor fusion based on deep feature fusion.

Compared with the previous two sensor fusion systems, this method have a few advantages. Data fusion systems use a single network to model the concatenated data. However, since HSI and LiDAR have very different properties, they should be modeled using different network structures. As discussed previously, CRNN is a good model for HSI and CNN is a proper model for LPW. Decision fusion could be a better method than data fusion since it treat different sources separately and their final predictions are averaged. However, the two base networks are trained independently. The deep feature fusion system, as an end-to-end classification model, train the two base networks simultaneously and they benefit from each other.

4.5 Experiments

This section is devoted to illustrate the capabilities of the presented deep neural networks (CNN, RNN and CRNN) for two real hyperspectral image datasets in remote sensing. A traditional method based on support vector machines (SVM) with radial basis function (RBF) kernel is also implemented for comparison.

4.5.1 Datasets

Two modern high resolution hyperspectral images are used in our study. One covers an urban area over the University of Houston at Houston, Texas, and the other covers a mixed vegetation cite at the Indian Pines area in Indiana.

The first dataset used in this work, the University of Houston (UH) hyperspectral image, has been described in Section 2.6. Here we also show the mean spectral signatures (radiance) for each class in Fig. 4.13. The spectral signatures tells that different classes have different shapes and local structures. Thus, when we treat the hyperspectral pixels as sequences, RNNs can be used to extract discriminative contextual information from them, which is very useful for classification.

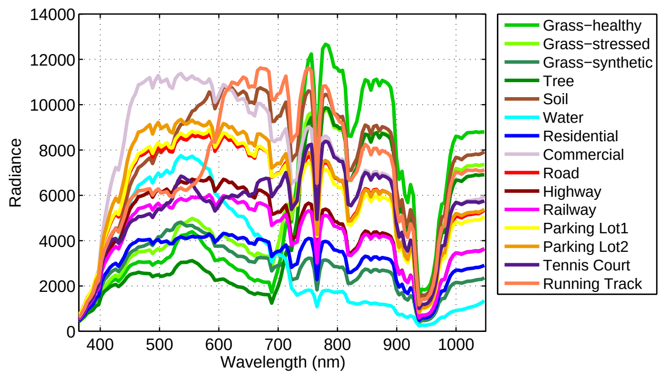


Figure 4.13: Mean spectral signatures of the 15 classes from the UH dataset.

The second dataset, the Indian Pines hyperspectral image, has been described in Section 3.4.

4.5.2 Experimental Setup

For both datasets, we randomly split the labeled samples into training and test sets. During training, 10% of the training samples are used as a validation set to learn the hyperparameters of the neural networks (i.e., layer size, number of layers, learning rate and mini-batch size) using a grid search strategy.

In our experiments, the CNN has 4 convolutional (with pooling layers), RNN and LSTM have 3 recurrent layers, CRNN and CLSTM have 2 convolutional layers (with pooling layers) and 2 recurrent layers. We also implemented the 2-D CNN in the same way as [34, 36] to extract spatial features from hyperspectral images. In particular, PCA was first employed on the whole image to reduce the dimensionality down to 3, and then we take a neighborhood region of size 11×11 around each labeled pixel to form 2-D images which will be fed to the input of the 2-D CNN which has 3 convolutional layers.

The configurations of all networks used in our experiments are summarized in Tables 4.1 and 4.2, where convolutional layers are denoted as "conv$\langle$receptive field size$\rangle$-$\langle$number of filters⟩" and recurrent layers are denoted as "recur-⟨feature dimension⟩".

Table 4.1: Summary of network configurations for University of Houston dataset.

CNN	2-D CNN	RNN/LSTM	CRNN/CLSTM
input-144			
conv6-32	conv3×3-32	recur-128	conv6-32
maxpool	conv3×3-32	recur-256	maxpool
conv6-32	maxpool	recur-512	conv6-32
maxpool	conv3×3-64		maxpool
conv3-64	maxpool		recur-256
maxpool			recur-512
conv3-64			
maxpool			
fully connected-15			

Table 4.2: Summary of network configurations for University of Houston dataset.

CNN	2-D CNN	RNN/LSTM	CRNN/CLSTM
input-180			
conv10-32	conv3×3-64	recur-128	conv10-32
maxpool	conv3×3-64	recur-256	maxpool
conv10-32	maxpool	recur-512	conv10-32
maxpool	conv3×3-96		maxpool
conv5-64	maxpool		recur-256
maxpool			recur-512
conv5-64			
maxpool			
fully connected-19			

We implemented the neural networks using the TensorFlow [114] and Keras [115] framework. Experiments are carried out on a workstation with a 3.0 GHz Intel(R) Core i7-5960X CPU, and an NVIDIA(R) GeForce Titan X GPU. The training process starts with the weights of all networks randomly initialized and the initial learning rate is set to 10^{-4}. For mini-batch stochastic gradient descent, we use a batch size of 128 in all experiments. During training, learning rate will be decreased by half every 500 epochs until loss reaches convergence.

4.5.3 Results

Since the datasets are highly unbalanced, so we run our experiments with the same number of training samples for each class. For University of Houston dataset which has 15 classes, we run experiments with 50 samples per class (750 in total), 100 samples per class (1500 in total) and 200 samples per class (3000 in total). For Indian Pines dataset which has 19 classes, we run experiments with 100 samples per class (1900 in total), 200 samples per class (3800 in total) and 300 samples per class (5700 in total). The classification performances for different models (SVM, CNN, 2-D CNN, RNN, LSTM, CRNN, CLSTM) and the corresponding methods with LOP spatial constraint on University of Houston dataset are presented in Table 5.3, and results on Indian Pines dataset are presented in Table 4.4.

Table 4.3: Classification results obtained by different approaches with different number of training samples on University of Houston dataset

training set size	750	1500	3000
RBF-SVM	89.42(±1.91)	92.86(±1.13)	95.43(±0.77)
CNN	90.91(±0.69)	93.42(±0.71)	96.25(±0.46)
2-D CNN	88.85(±1.52)	94.26(±0.61)	97.14(±0.62)
RNN	78.67(±1.93)	82.84(±1.72)	92.16(±0.66)
LSTM	86.55(±0.89)	91.87(±0.63)	94.05(±0.54)
CRNN	93.42(±0.46)	95.33(±0.41)	97.64(±0.30)
CLSTM	90.52(±0.77)	94.53(±0.47)	97.55(±0.44)
CNN-LOP	93.36(±0.81)	95.02(±1.03)	97.87(±0.50)
RNN-LOP	88.11(±1.39)	93.52(±1.43)	96.40(±0.88)
LSTM-LOP	91.86(±1.50)	94.67(±0.62)	97.11(±0.44)
CRNN-LOP	95.17(±0.11)	97.08(±0.36)	98.61(±0.37)
CLSTM-LOP	93.22(±1.16)	96.28(±0.82)	98.21(±0.45)

Analyzing the classification performances on both datasets, we have the following conclusions:

1. CNN and CRNN/CLSTM achieved better classification results than traditional method RBF-SVM in all scenarios, while the performances of RNN/LSTM are still worse than RBF-SVM.

Table 4.4: Classification results obtained by different approaches with different number of training samples on Indian Pines dataset

training set size	1900	3800	5700
RBF-SVM	92.82(±1.07)	94.36(±0.93)	95.13(±0.64)
CNN	93.11(±0.95)	94.53(±0.39)	95.84(±0.31)
2-D CNN	88.78(±0.85)	92.04(±0.58)	92.82(±0.65)
RNN	84.83(±1.62)	89.74(±0.98)	91.86(±0.77)
LSTM	85.04(±1.15)	89.83(±0.74)	92.15(±0.51)
CRNN	94.43(±1.01)	96.24(±0.60)	96.83(±0.47)
CLSTM	92.72(±1.08)	95.13(±0.57)	96.16(±0.55)
CNN-LOP	94.99(±0.85)	96.78(±0.76)	97.26(±0.46)
RNN-LOP	91.39(±1.11)	94.44(±0.53)	95.27(±0.52)
LSTM-LOP	91.62(±0.36)	94.92(±0.38)	95.32(±0.41)
CRNN-LOP	96.61(±0.75)	96.98(±0.29)	98.08(±0.44)
CLSTM-LOP	95.17(±0.68)	96.52(±0.81)	97.01(±0.44)

2. The 2-D spatial CNN performs better than SVM on University of Houston dataset but worse on the Indians Pines dataset. The reason is that University of Houston dataset contains urban objects which have much more spatial features than the vegetation categories in Indian Pines dataset.

3. As expected, the performances of CRNN/CLSTM are better than CNN because CRNN/CLSTM have advantages of both convolutional networks and recurrent networks.

4. The fact that the performances of CRNN/CLSTM are significantly better than RNN/LSTM tells us that the middle-level features extracted by the convolutional layers in CRNN/CLSTM help the following recurrent layers better captures the contextual information.

5. LSTM network has better performances than the regular RNN especially for University of Houston dataset because LSTM networks are capable of capturing the long-term dependencies in the input sequence and thus avoid the gradient vanishing problem.

6. CLSTM performs no better than CRNN in all cases, meaning that CRNN does not have the long-term dependency and gradient vanishing problem because the length of the

sequence is already much reduced by the two pooling layers before the recurrent layers. The reason why CLSTM is even worse than CRNN when the training set is small is that the CLSTM has much more parameters than CRNN so it tend to overfit the training data and performs worse on test data.

7. LOP-based spatial constraint further improved the performances of all the 1-D models.

Table 4.5: Number of parameters and training time for University of Houston dataset.

Network	# parameters	training epochs	run time (min.)
CNN	33615	5000	15
2-D CNN	37295	500	3.6
RNN	516623	5000	158
LSTM	2043407	2000	330
CRNN	481807	500	4.7
CLSTM	1884943	500	14

We also show he training and validation loss, training and validation accuracy of when training the neural networks on the University of Houston dataset with 1500 samples in Fig. 4.14. For every neural network, the training loss all converged to a level close to 0 and training accuracy converged to 100%. At the same time, the validation loss all converged to a low level and the validation accuracy all reached a high number. It's also worth noting that CRNN and CLSTM converge faster than than CNN and RNN. The corresponding number of parameters and training time (in minutes) for each network are summarized in Table 4.5. We can see that RNN and LSTM needs much more training time than the others because the computation complexity for RNN/LSTM grows linearly with respect to the length of the input sequence and most of the computations need to be done sequentially.

To better understand the classification power of each neural network, we take the high dimensional features extracted by all models trained on the University of Houston dataset with 1500 training samples, and use t-SNE [116] algorithm to reduce the dimensionality to 2. The results are visualized in Fig. 4.15 where different colors stands for 15 classes in

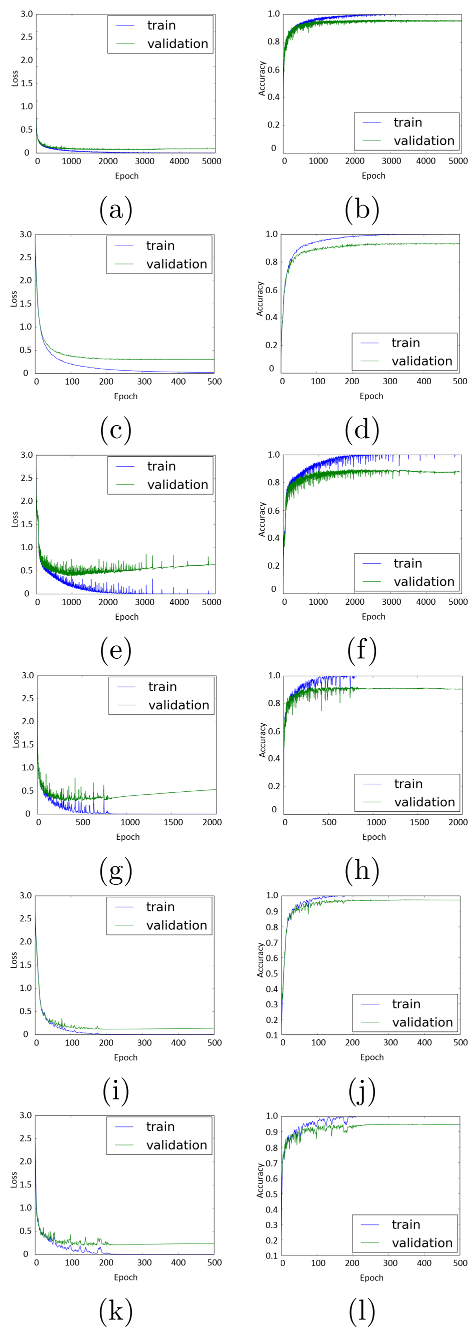
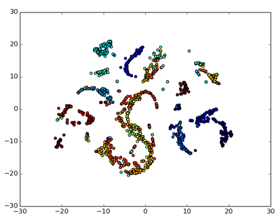
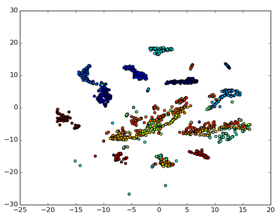


Figure 4.14: Loss and accuracy for training and validation set: (a) and (b) for CNN; (c) and (d) for 2-D CNN; (e) and (f) for RNN; (g) and (h) for LSTM; (i) and (j) for CRNN; (k) and (l) for CLSTM.

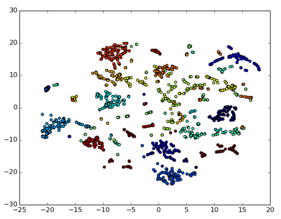
the University of Houston dataset. Similarly, the feature visualizations for the Indian Pines dataset are depicted in Fig. 4.16. Features extracted by all models are more discriminative than the original hyperspectral data. Features extracted by the CRNN for different classes



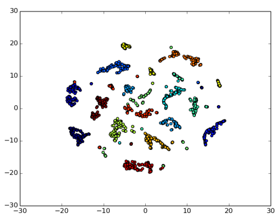
(a)



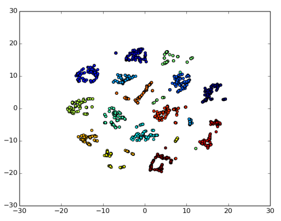
(b)



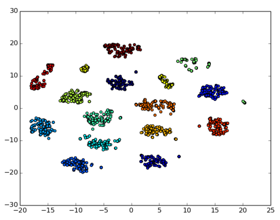
(c)



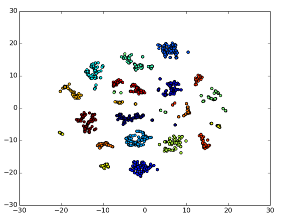
(d)



(e)



(f)



(g)

Figure 4.15: University of Houston dataset: two dimensional embedding extracted by t-SNE for (a) input data, and features extracted by (b) CNN, (c) 2-D CNN, (d) RNN, (e) LSTM, (f) CRNN, (g) CLSTM.

are better separated than the other models, which means that features extracted by CRNN are the most discriminative. The reason is that the first few convolutional (and pooling) layers in CRNN extract middle level features where the spectral variation has been decreased, which makes it easier for the following recurrent layers to learn the contextual information. The efficacy of the convolutional layers can be seen from Fig. 4.17 which shows an example of

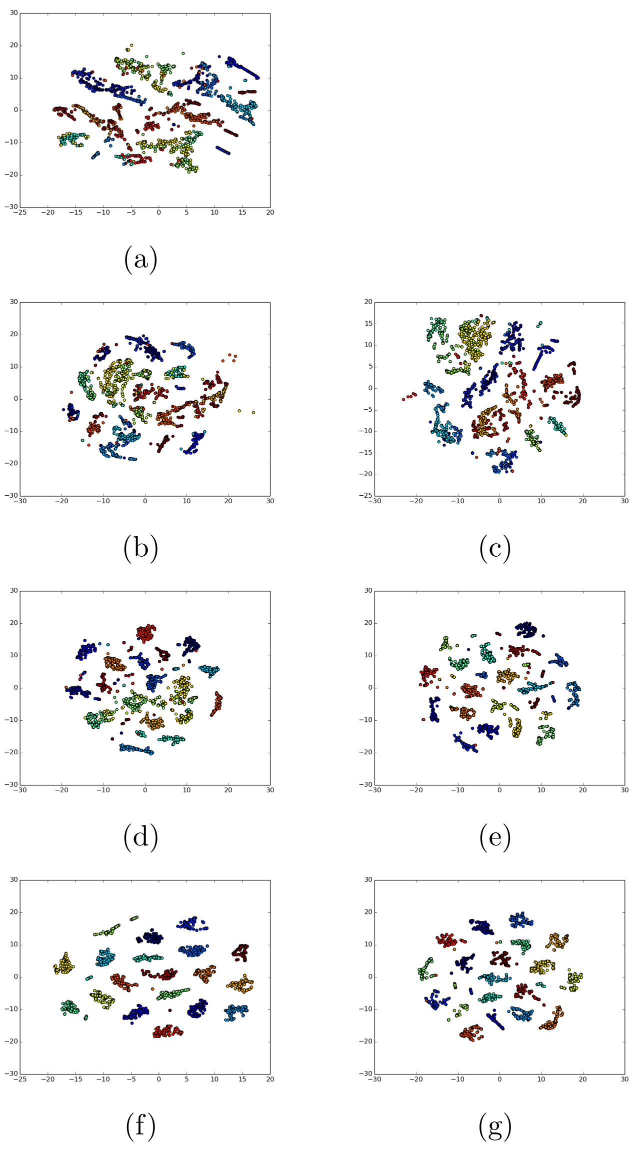
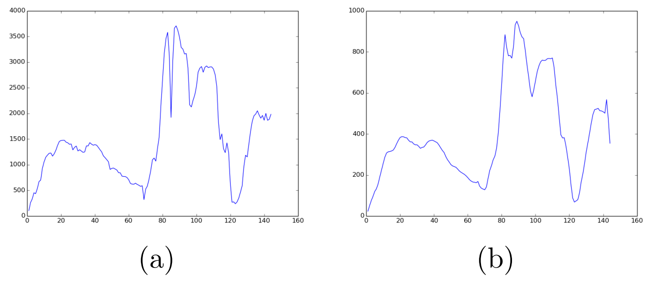


Figure 4.16: Indian Pines dataset: two dimensional embedding extracted by t-SNE for (a) input data, and features extracted by (b) CNN, (c) 2-D CNN, (d) RNN, (e) LSTM, (f) CRNN, (g) CLSTM.

features extracted by the convolutional layers. Compared to the input signal, the extracted features are more smooth since convolutional layers are able to remove local variations. Furthermore, important information such as locations of peaks and slopes was successfully captured by the convolutional layer.

Finally, we show the classification maps for different models used in this work in Fig.


(a) (b)

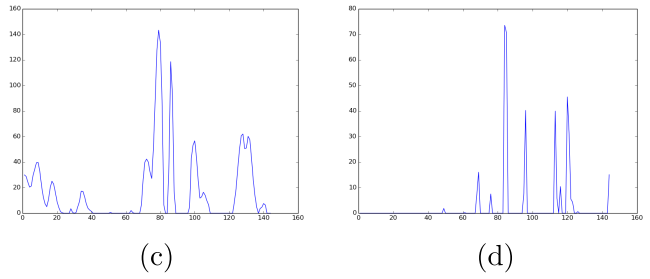
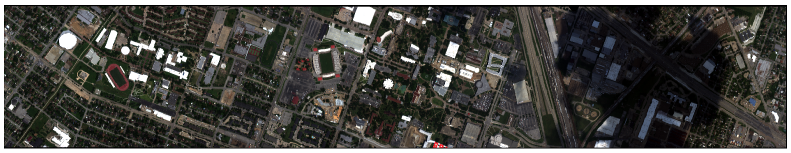
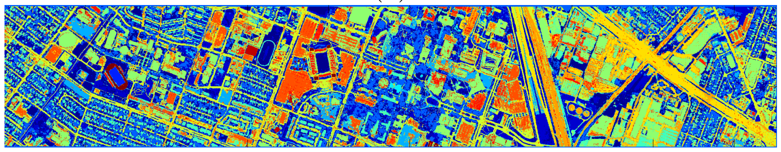

(c) (d)

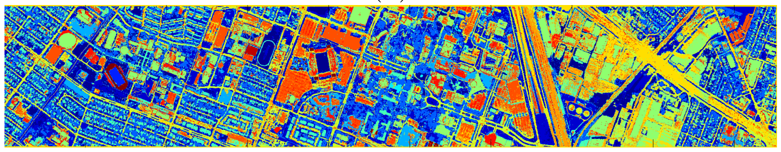
Figure 4.17: An input hyperspectral pixel in (a) and features extracted by the convolutional layer in (b), (c) and (d).

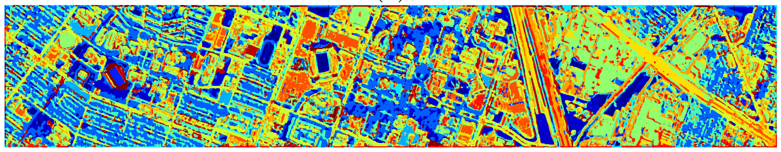
4.18 and 4.19. It is easy to see that the classification map of 2D spatial CNN is more smooth than the other 1-D models in Fig. 4.18, but a lot of important details (like edges between different classes) got smoothed out as well, which resulted in incorrect prediction around the edges. These incorrect classifications are caused by the fact that the 2-D CNN extracts features from a large sliding window that's centered on each pixel. However, the sliding windows around edges contain objects from multiple classes, which makes it difficult for the model to make the correct predictions. The other 1-D models don't have such problem even their maps look noisy in some regions. When adding LOP as spatial constraint to the 1-D models, the maps for 1-D models all become smoother and less noisy, while the important edges still got preserved.

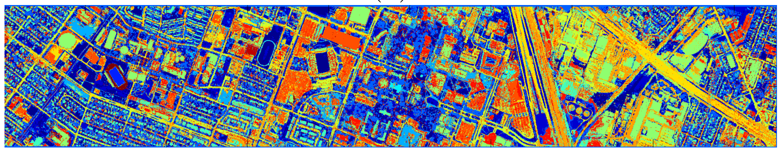
For the sensor fusion experiments, we randomly selected 750 samples (i.e., 50 per class) or 1500 sample (i.e., 100 per class) for training the neural networks and test on another 1500 samples (i.e., 100 per class). The classification results for single source and different sensor fusion systems are presented in Table 4.6 where each experiment is repeated 10 times

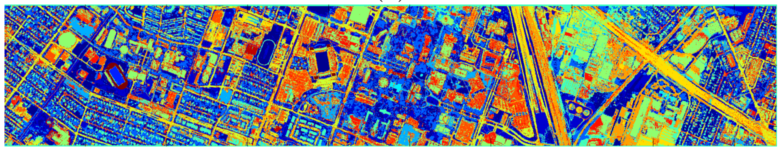

(a)

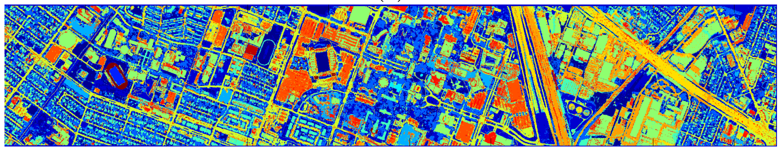

(b)


(c)


(d)


(e)


(f)


(g)

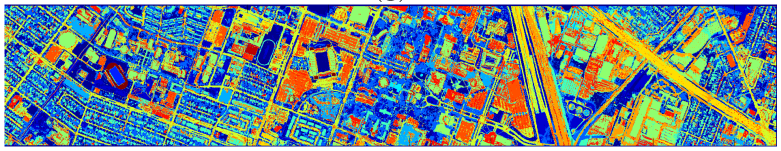

(h)

Figure 4.18: University of Houston dataset: (a) RGB image. (b)–(f) Classification maps for different models: (b) SVM, (c) CNN, (d) 2-D CNN, (e) RNN, (f) LSTM, (g) CRNN and (h) CLSTM.

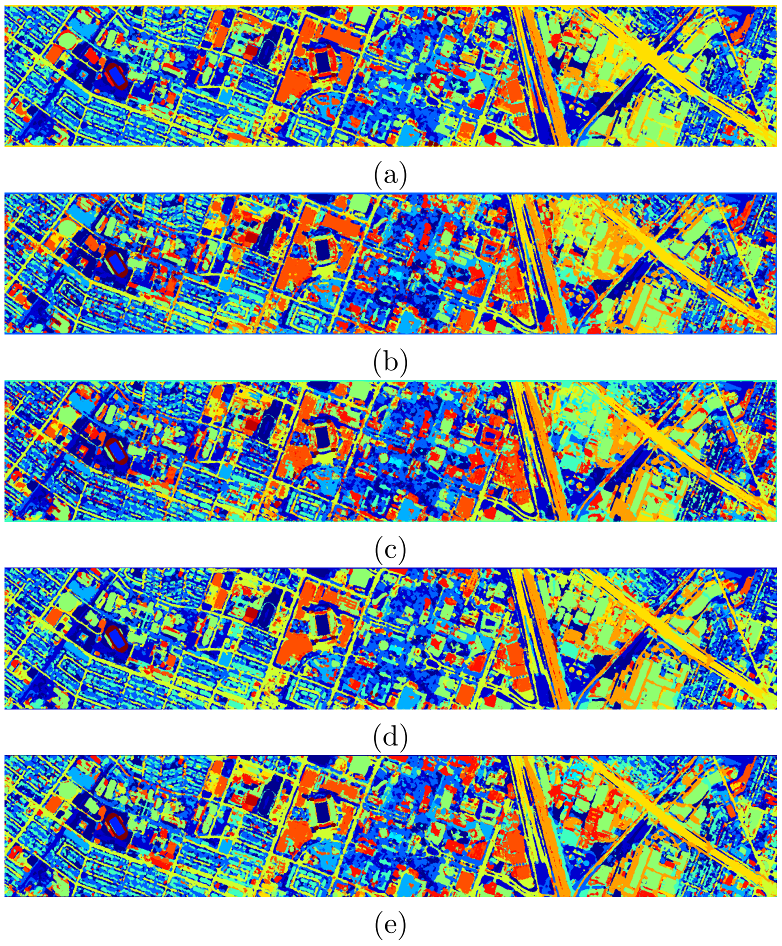


Figure 4.19: University of Houston dataset: Classification maps for different models when using LOP as spatial constraint: (a) CNN, (b) RNN, (c) LSTM, (d) CRNN and (e) CLSTM.

with randomly selected training and test sets, and the average accuracy with the standard deviation is reported.

Table 4.6 shows that the proposed deep feature fusion achieved the best performances among all methods. It's also interesting to note that, for single source classification, HSI is much better than LPW because HSI alone provides more discriminative information than the LPW. In addition, data fusion and decision fusion show no significant improvement compared to the HSI only classification method. Moreover, if we apply these methods to the whole image, we can get the classification maps. Fig. 4.20 shows the classification maps

Table 4.6: Classification performances for single source and different sensor fusion methods.

Training set size	750	1500
HSI only	92.75(1.39)	95.26(0.77)
LPW only	54.03(1.89)	56.99(1.82)
Data fusion	93.14(1.36)	95.47(0.35)
Decision fusion	93.01(0.92)	95.79(0.54)
Deep feature fusion	**94.87(1.24)**	**96.89(0.67)**

for a cropped region (with a spatial dimension of 300×300) from the UH dataset, which demonstrates the benefit of sensor fusion. For example, the HSI only method incorrectly classified the roof of one building as parking lot because they have similar spectral measurements. LPW only method can correctly recognize the building roof but is not doing well for other classes such as roads and grass. The three sensor fusion methods generate better classification maps than the single source methods. Especially, the proposed deep feature fusion method produced the best classification map (e.g., the predictions on the building roof and parking lot are less noisy and more accurate than the other two sensor fusion methods).

4.6 Conclusions

In this Chapter, we proposed to use RNN to extract the contextual information in hyperspectral data by modeling the dependencies between different spectral bands. In particular, we employed a CRNN model, which consists of several convolutional layers and a few recurrent layers, for hyperspectral data classification. The first convolutional layers are utilized to extract middle-level and locally invariant features which are then fed to a

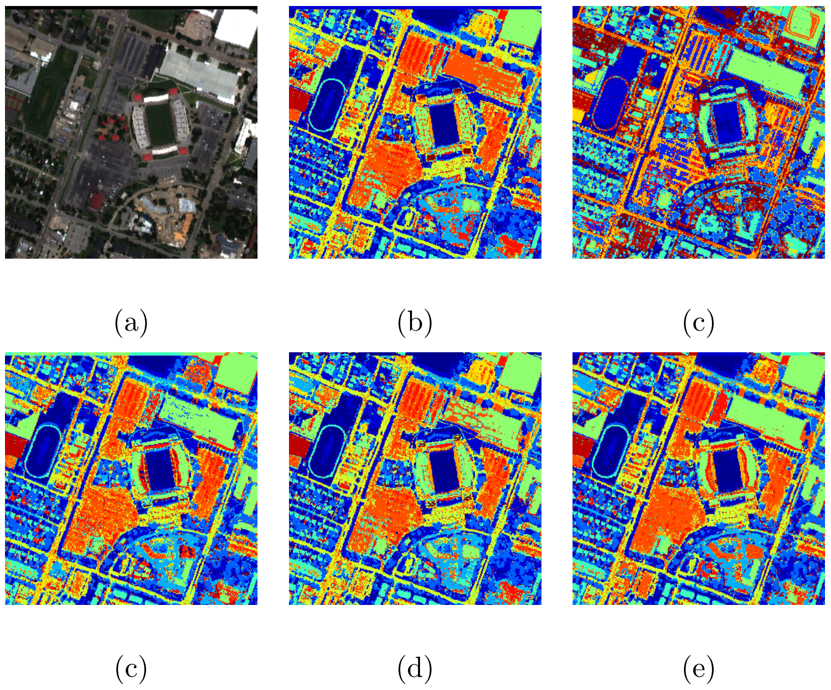


Figure 4.20: Classification maps by different systems: (a) RGB image, (b) HSI only, (c) LPW only, (d) Data fusion, (e) Decision fusion, (f) Deep feature fusion.

few recurrent layers to additionally extract the contextual information between different spectral bands. By combining convolutional and recurrent layers, our CRNN model is able to extract more discriminative feature representations for classification, and it outperformed other state-of-the-art methods on real hyperspectral image datasets.

We also investigated different sensor fusion methods using deep learning techniques and experimental results have shown that the proposed deep feature fusion method outperforms the other sensor fusion methods including data fusion and decision fusion. The proposed method has profound benefits for modern remote sensing applications where we have multiple data sources and enough training samples to train deep neural networks.

Chapter 5

Semi-supervised Deep Learning using Pseudo Labels

5.1 Introduction

In recent years, with the development of computing hardware like GPUs and the availability of large scale datasets, deep learning [82] techniques have gained a lot of success in a variety of machine learning tasks such as computer vision [20, 21, 23, 24], speech recognition [25–28], and natural language processing [29–31] etc.

Hyperspectral data, with rich spectral information, have been playing a key role in remote sensed data analysis [87] such as land cover classification [92, 100, 101] and anomaly detection [102]. Recently, deep learning techniques was introduced into the remote sensing community especially for hyperspectral image analysis and achieved state-of-the-art performance. For example, stacked autoencoders were used to extract deep features from hyperspectral data in [32]. Convolutional neural networks (CNN) were utilized for hyperspectral image classification in [33]. In [117], a recurrent neural network (RNN) was used for land cover change detection on multi-spectral images. By treating the hyperspectral data as spectral sequences, convolutional recurrent neural networks (CRNN) [110] were used to extract the contextual information (i.e., dependencies between different spectral bands) for hyperspectral image classification and achieved better performances than CNN based classification methods. The CRNN is composed of a few convolutional layers (and pooling layers) followed by a few recurrent layers. The convolutional layers first extract middle-level, locally invariant features from the input hyperspectral sequence. Pooling layers make the feature sequence shorter by subsampling, which will accelerate the computation and help avoid overfitting. The following recurrent layers can then extract contextual information from the middle-level feature sequences generated by the previous convolutional layers. Thus, in this work, we choose to use CRNN as the basic model for hyperspectral image classification.

Training a deep neural network for classification requires a large amount of labeled samples to learn a large number of parameters. However, for hyperspectral image classification in real remote sensing applications, we are usually provided a large hyperspectral image with only a small amount of labeled samples available for training because collecting them are expensive and time-consuming. On the other hand, we always have access to a large quantity of unlabeled data from the given hyperspectral images. Thus, semi-supervised learning [38] techniques, which make use of both labeled and unlabeled data, have been popular in classification of remotely sensed hyperspectral images. For instance, transductive Support Vector Machines (TSVM) were used in [118, 119] for semi-supervised classification of hyperspectral data. Laplacian Support Vector Machines (LapSVM) [120] were used for semi-supervised classification by regularizing the standard SVM with the graph Laplacian using unlabeled data. A spatio-spectral LapSVM (SS-LapSVM) was proposed in [121] which also took the spatial information of hyperspectral images into consideration. Deep neural networks can also be used in semi-supervised learning. This can be done by pre-training a deep network such as a stacked denoising autoencoder [122] using unlabeled data in an unsupervised way, followed by supervised fine-tuning using labeled data [123, 124]. Ladder networks [125] was a recently proposed method for semi-supervised learning which simultaneously minimize a supervised and an unsupervised cost during training.

Traditionally, we need a large set of labeled data to train supervised deep neural networks. In our proposed semi-supervised learning framework, the large set of unlabeled samples are utilized with their pseudo labels to pre-train a deep neural network. The pseudo labels are defined as the cluster labels generated by a clustering algorithm and they have been used for semi-supervised dimensionality reduction in Chapter 2 where the Dirichlet process mixture model (DPMM) [14, 15, 18] based clustering algorithm was employed. In this work, the proposed method is named as Semi-Supervised Deep Learning using Pseudo Labels (PL-SSDL). It works in the following way: We first run clustering on all the training data (both labeled and unlabeled) to get their pseudo labels. Since the clustering algorithm tends to assign samples from different classes to different groups, the discriminative information get preserved in the pseudo labels. This is why we can use unlabeled data with pseudo labels to train a deep neural network to extract discriminative features which are useful for classification. Next, all the training data together with their pseudo labels are used to pre-train a deep neural network (such as a CRNN). Then, we construct a new deep network by taking all the layers except the last layer from the pre-trained network, and add a few more fully connected layers and a classification layer afterwards. Finally, we fine-tune this new deep network using only the labeled data with the true labels. Since the labeled data are very limited in quantity, we only fine-tune the newly added layers using backpropagation and freeze the pre-trained layers. In order to further improve the performance, we propose to use the spatial information in the clustering process via the constrained DPMM (C-DPMM) [126] which adds pairwise must-link constraints enforced by superpixels [127, 128] and cannot-link constraints enforced by the labeled samples. The

C-DPMM generates pseudo labels of higher quality, which results in a better pre-trained model.

Typically, we need a large set of labeled data to effectively train supervised deep neural networks. In our proposed semi-supervised learning framework, a large set of unlabeled samples are utilized instead, wherein their *pseudo labels* are used to pre-train a deep neural network. Pseudo labels are defined as the cluster labels generated by a clustering algorithm, such as the Dirichlet process mixture model (DPMM) [14, 15, 18]. In this work, the proposed method is called Semi-Supervised Deep Learning using Pseudo Labels (PL-SSDL).

Our approach works in the following way: We first perform clustering on all the training data (both labeled and unlabeled) and obtain pseudo labels. Since the clustering algorithm tends to assign samples from different classes to different groups, the discriminative information is preserved in these pseudo labels. Hence, we suggest that one can use unlabeled data with pseudo labels to train a deep neural network to extract discriminative features which are useful for classification tasks related to the dataset. Next, all the training data together with their pseudo labels are used to pre-train a deep neural network (such as a CRNN). Following this, we construct a new deep network by taking all the layers except the last layer from the pre-trained network, and add fully connected layers and a classification layer. Finally, we fine-tune this new deep network using only the labeled data with the true class labels. Since the labeled data are very limited in quantity, we only fine-tune the newly added layers using back-propagation, while freezing the pre-trained layers. In order to further improve the performance, we propose to use the spatial information in the clustering process via the constrained DPMM (C-DPMM) [126] which adds pairwise must-link constraints enforced by superpixels [127, 128] and cannot-link constraints enforced by the labeled samples. The C-DPMM generates pseudo labels of higher quality, which results in a better pre-trained model.

The remainder of this Chapter is organized as follows. Section 5.2 provides a review of previous work on DPMM and deep learning for hyperspectral data classification that are related to our method. The proposed semi-supervised deep learning method using pseudo labels is described in detail in Section 5.3. The experimental datasets, setup and results are presented in Section 5.4. Section 5.5 summarizes the key ideas in this work and provides concluding remarks.

5.2 Related Work

5.2.1 DPMM based Clustering

Clustering is an unsupervised learning task that separates data into different groups based on their similarities. Parametric clustering models such as k-means and Gaussian mixture models assume the number of clusters is known a priori. However, due to spatial variation, many classes in a hyperspectral image have multi-modal distributions, so assuming the number of clusters to be the number of classes is not an optimal choice. The DPMM [14, 15, 18] is a nonparametric model for clustering, which has the ability to infer the number of clusters from the data. Readers can refer to Section 2.3 of Chapter 2 for more details about DPMM based clustering.

5.2.2 CRNN

Deep neural networks such as CNNs, RNNs and CRNNs have been used for hyperspectral data classification successfully [33, 110]. CNNs view the hyperspectral data samples as high dimensional input vectors to the network and are trained to extract deep features from them. RNNs are a type of sequence models which are widely used for extracting contextual information from sequential data by modeling the dependencies between different steps of the input sequence. When RNNs are applied to hyperspectral data classification, each hyperspectral sample is treated as a spectral sequence. The features extracted by RNNs contains the contextual information between different spectral bands and can be used for classification [110]. The CRNN is a hybrid model of convolutional and recurrent neural networks [96, 97, 110]. It's composed of several convolutional (and pooling) layers followed by a few recurrent layers, as indicated in the graphical illustration of the architecture of the CRNN used in this work in Fig. 5.1. The convolutional layers first extract middle-level and locally invariant features from the input sequence. The pooling layers help reduce computation and control overfitting. The recurrent layers further extract contextual information from the feature sequences generated by the convolutional layers. Contextual information captures the dependencies between different spectral bands of the input sequence, which is more stable and useful for classification.

Let's denote an input hyperspectral vector as $\boldsymbol{x} = (x_1, x_2, ..., x_T) \in \mathbb{R}^T$ where T is the length of the input vector. In the first convolutional layer, a set of d filters $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_d\}$ of receptive field size (i.e., length of convolutional filters) r, are applied to the input vector via convolution operation ($*$) to get the feature map using

$$\boldsymbol{F} = (\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_T) = g(\boldsymbol{x} \ * \ \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_d\}), \tag{5.1}$$

where $\boldsymbol{f}_t \in \mathbb{R}^d$ and g is a nonlinear activation function such as the hyperbolic tangent function or rectified linear unit (ReLU). ReLU is defined as $g(x) = max(0, x)$ which has become the most widely used activation function in CNNs, and we also choose to use it in this work. The max pooling layer subsample every depth slice of the input by max operation. In this work, we used a pooling layer with filters of length 2 applied with a stride of 2. After applying the pooling layer, the depth (dimension) of the features remains

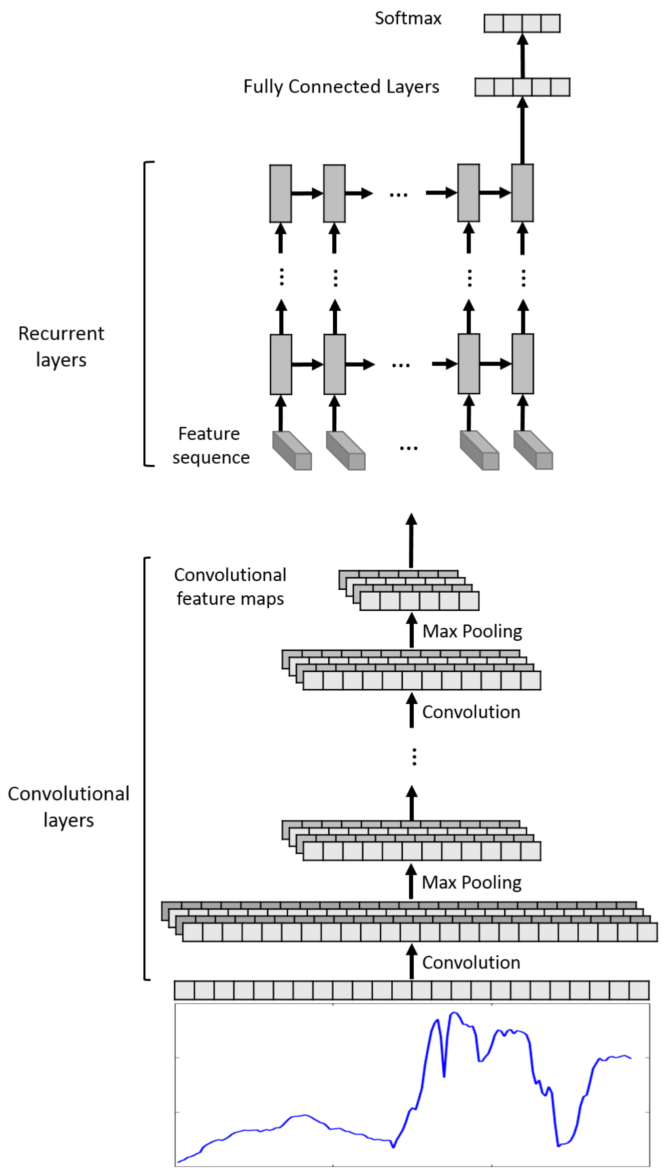


Figure 5.1: Architecture of the CRNN consisting of multiple convolutional layers and recurrent layers for hyperspectral data classification.

unchanged but the length is reduced by half.

The convolutional layers are followed by a few recurrent layers which treat the features extracted by the convolutional layers as a sequence of inputs and aim to extract contextual information from them. A recurrent layer has a recursive function f which takes as input one feature vector $\boldsymbol{f}_t$ and the previous hidden state $\boldsymbol{h}_{t-1}$, and returns the new hidden state as

$$\boldsymbol{h}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = tanh(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1}), \tag{5.2}$$

and the outputs are calculates by

$$\boldsymbol{o}_t = softmax(\boldsymbol{V}\boldsymbol{h}_t), \tag{5.3}$$

where $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{V}$ are the weight matrices which are shared across all steps, and activation function $tanh$ is the hyperbolic tangent function. The output of one recurrent layer is a sequence of features which will be feed into the next recurrent layer. The last recurrent layer will return a sequence of high-level features. As indicated in Fig. 5.1, we only need to take the last hidden state of the last recurrent layer as the input to the following fully recurrent layers because the last hidden state already contains all the useful contextual information in the previous hidden states.

Finally, the features extracted by the last fully connected layer are fully connected to the output layer for classification where we have a softmax activation function to compute the predictive probabilities for all categories. This is done by calculating

$$p(y = k|\boldsymbol{x}) = \frac{exp(\boldsymbol{w}_k^\top \boldsymbol{x} + b_k)}{\sum_{k'=1}^{K} exp(\boldsymbol{w}_{k'}^\top \boldsymbol{x} + b_{k'})}, \tag{5.4}$$

where $\boldsymbol{w}_k$'s and b_k's are the weight and bias vectors, and there are K categories.

Training a neural network is to find values for the parameters (weights of all layers) to minimize the loss function, which in a classification task measures the compatibility between a prediction (e.g., the class scores calculated by the softmax function) and the ground truth label. The loss takes the form of an average over the losses for every training example using

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i, \tag{5.5}$$

where N is the number of training samples and L_i is the loss for the i-th sample. In our study, we used the cross-entropy loss (also known as negative log likelihood), which is the most widely used loss function for the softmax activation function defined as

$$L_i = -log(p(y_i|\boldsymbol{x}_i)). \tag{5.6}$$

The network is trained using stochastic gradient descent (SGD) method. The gradients in the convolutional layers are calculated by the back-propagation algorithm and gradients in the recurrent layers are calculated by the back-propagation through time (BPTT) algorithm [108]. A mini-batch strategy is utilized in our implementation to reduce the loss fluctuation, so the gradients are calculated with respect to mini-batches. The algorithm will be running iteratively until the loss converges when the change of training and validation loss is below some threshold.

5.3 The proposed Method

Section 5.2 describes DPMM for clustering and CRNN for hyperspectral data classification. In this section, we are going to introduce a semi-supervised clustering algorithm based on C-DPMM for pseudo label generation, and a semi-supervised deep learning framework which trains two CRNNs. The structure for the proposed semi-supervised deep learning framework is shown in Fig. 5.2. In the following discussion, the labeled data set is denoted as $\{\boldsymbol{X}_L, \boldsymbol{Y}_L\}$, the unlabeled data set is denoted as $\{\boldsymbol{X}_U\}$, and the pseudo labels for the labeled and unlabeled data set $\boldsymbol{X} = \{\boldsymbol{X}_L, \boldsymbol{X}_U\}$ is denoted as $\boldsymbol{Z}$. There are mainly three steps to run this semi-supervised learning framework. Firstly, the pseudo labels for the labeled and unlabeled data are generated by the constrained DPMM (C-DPMM) where the constraints come from pairwise must-link and cannot-link constraints. Secondly, the combined data set $\boldsymbol{X}$ with the corresponding pseudo labels $\boldsymbol{Z}$ are used to pre-train a deep

CRNN which aims to discriminate data points with different pseudo labels. Thirdly, the pre-trained CRNN is used to initialize the first part of another CRNN. The second part of this new CRNN consists of a few fully connected layers and a classification layer (i.e., softmax classifier). Then we fine-tune the second part of the new CRNN using only the labeled samples. This proposed semi-supervised deep learning using pseudo labels We are going to discuss each step with more detail in the following paragraphs.

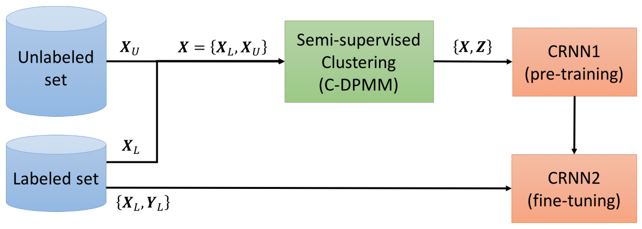


Figure 5.2: The structure of the proposed semi-supervised deep learning framework.

5.3.1 Generation of Pseudo Labels via C-DPMM

Pseudo labels are defined as cluster assignments generated by any clustering algorithm. The DPMM based clustering described in Section 5.2.1 is done in an unsupervised fashion without using any label information. In our problem, since we are given a labeled set $\{\boldsymbol{X}_L, \boldsymbol{Y}_L\}$, we propose to use the constrained DPMM (C-DPMM) [126] for semi-supervised clustering to further improve the clustering performance. C-DPMM imposes must-link and cannot-link pairwise constraints between the labeled samples in $\{\boldsymbol{X}_L, \boldsymbol{Y}_L\}$ to the standard DPMM. Specifically, the must-link pairwise constraints require that two samples with the same label should be assigned the same cluster label, whereas the cannot-link pairwise constraints specify that samples from different classes should not be assigned to the same cluster. Here we use $\mathcal{M}$ and $\mathcal{C}$ to denote the set of must-links and set of cannot-links. However, for hyperspectral data, it is common that some classes have multi-modal distributions due to spatial variations. Thus the must-link constraints may not be valid for them. Thus in our implementation, we didn't impose the must-link constraints between the labeled samples. Instead, we impose spatial must-link constraints between samples in both the labeled and unlabeled data $\boldsymbol{X} = \{\boldsymbol{X}_L, \boldsymbol{X}_U\}$ via superpixels [127, 128]. In other words, we segment the hyperspectral image into superpixels and require that pixels in $\boldsymbol{X}$ that are from the same superpixel on the image should be grouped to the same cluster. This spatial must-link constraints are valid based on the fact that pixels within the same superpixel usually belong to the same object and have similar values, as illustrated in Fig. 5.3.

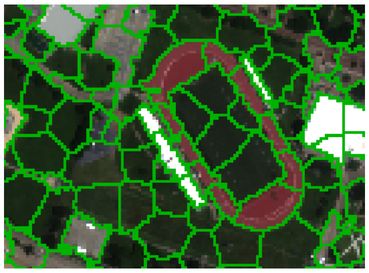


Figure 5.3: A cropped region of the University of Houston hyperspectral image with superpixel segmentation.

Compared to DPMM, the generative process of C-DPMM is modified in the following way [126]: must-linked samples are always generated by the same component, while cannot-linked samples are always by different ones. [126] derived an inference method based on Gibbs sampling. As discussed in Section 5.2.1, Gibbs sampling is usually much slower to converge than VI. So in our study, we derived an inference scheme for C-DPMM based on VI.

First, we identify linked groups of samples where each group is linked via the must-link constraints. For samples in a linked group, we compute their component assignment jointly while taking into account the cannot-link constraints. Based on Eq. 2.22, the variational distribution for C-DPMM is modified as

$$q(\boldsymbol{V}, \boldsymbol{\Theta}, \boldsymbol{Z}) = \left[\prod_{t=1}^{T} q(v_t)\right] \left[\prod_{k=1}^{T} q(\boldsymbol{\theta}_k)\right] \left[\prod_{l=1}^{L} q(Z_l|\mathcal{C})\right], \tag{5.7}$$

where l is the index of the groups enforced by the set of must-links $\mathcal{M}$, L is the total number of groups, and Z_l is the corresponding component assignment for the l-th linked group.

Then the free energy, the objective function to be optimized, for C-DPMM can be written as

$$\begin{aligned} F = &\sum_{k=1}^{T} \mathbb{E}\left[log \frac{q(\boldsymbol{\theta}_k)}{p(\boldsymbol{\theta}_k)}\right]_{q(\boldsymbol{\theta}_k)} + \sum_{k=1}^{T} \mathbb{E}\left[log \frac{q(v_k)}{p(v_k)}\right]_{q(v_k)} \\ &+ \sum_{l=1}^{L} \mathbb{E}\left[log \frac{q(Z_l)}{p(\boldsymbol{X}_l|Z_l, \Theta) p(Z_l|\boldsymbol{V})}\right]_{q(Z_l, \boldsymbol{V}, \boldsymbol{\Theta})}, \end{aligned}$$

which can be further simplifies as

$$\begin{aligned} F = &\sum_{k=1}^{T} \mathbb{E}\left[log \frac{q(\boldsymbol{\theta}_k)}{p(\boldsymbol{\theta}_k)}\right]_{q(\boldsymbol{\theta}_k)} + \sum_{k=1}^{T} \mathbb{E}\left[log \frac{q(v_k)}{p(v_k)}\right]_{q(v_k)} \\ &+ \sum_{l=1}^{L} log \sum_{k=1}^{T} exp(S_{l,k}). \end{aligned} \tag{5.8}$$

In Eq. 5.8, we have defined

$$S_{l,k} = \mathbb{E}\left[log\ p(\boldsymbol{X}_l|\boldsymbol{\theta}_k)\right]_{q(\boldsymbol{\theta}_k)} + \mathbb{E}\left[log\ p(Z_l = k|\boldsymbol{V})\right]_{q(\boldsymbol{V})}, \tag{5.9}$$

where

$$\begin{aligned} &p(\boldsymbol{X}_l|\boldsymbol{\theta}_k) = \prod_{i=1}^{N_l} p(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{R}_k) \text{ and} \\ &p(Z_l = k|\boldsymbol{V}) = \prod_{i=1}^{N_l} p(z_i = k|\boldsymbol{V}) = (v_k \prod_{j=1}^{k-1}(1 - v_j))^{N_l}. \end{aligned}$$

The free energy F is minimized using the CAVI algorithm by iteratively updating each factor in Eq. 5.7 until F converges. Using the properties of conjugate priors, the update rules for each factor in Eq. 5.7 can be calculated in the following way:

(a) For $t = 1, ..., T$:

$$q(v_t) = \mathcal{B}(a_{t1}, a_{t2}), \tag{5.10}$$

which is a Beta distribution with parameters a_{t1} and a_{t2} defined as

$$a_{t1} = 1 + \sum_{i=1}^{N} q(z_i = t) \text{ and}$$
$$a_{t2} = \alpha + \sum_{i=1}^{N} \sum_{j=t+1}^{T} q(z_i = j).$$

(b) For $k = 1, ..., T$:

$$q(\boldsymbol{\theta}_k) = \mathcal{NW}(r_k, \boldsymbol{m}_k, \nu_c, \boldsymbol{B}_c), \tag{5.11}$$

which is a Normal-Wishart distribution with the following parameters:

$$r_k = r_0 + N_k,$$
$$\boldsymbol{m}_k = \frac{N_k \overline{\boldsymbol{x}}_k + r_0 \boldsymbol{m}_0}{r_k},$$
$$\nu_k = \nu_0 + N_k, \text{ and}$$
$$\boldsymbol{B}_k = \boldsymbol{B}_0 + N_k \boldsymbol{S}_k + \frac{N_k r_0}{r_k} (\overline{\boldsymbol{x}}_k - \boldsymbol{m}_0)(\overline{\boldsymbol{x}}_k - \boldsymbol{m}_0)^\top,$$

where we define

$$N_k = \sum_{i=1}^{N} q(z_i = k),$$
$$\overline{\boldsymbol{x}}_k = \frac{1}{N_k} \sum_{i=1}^{N} q(z_i = k) \boldsymbol{x}_i, \text{ and}$$
$$\boldsymbol{S}_k = \frac{1}{N_k} \sum_{i=1}^{N} q(z_i = k)(\boldsymbol{x}_i - \overline{\boldsymbol{x}}_k)(\boldsymbol{x}_i - \overline{\boldsymbol{x}}_k)^\top.$$

(c) For $l = 1, ..., L$:

$$q(Z_l = k) \propto \begin{cases} exp(S_{l,k}), & \text{if } \boldsymbol{X}_{k,-l} \cap \mathcal{C} = \emptyset \\ 0, & \text{otherwise} \end{cases}, \tag{5.12}$$

where $\boldsymbol{X}_{k,-l}$ denotes all samples assigned to the k^{th} component excluding $\boldsymbol{X}_l$.

By iterating these updates derived in (a), (b) and (c), the CAVI algorithm finds a local minimum of the free energy upon convergence. Finally the clustering label for each group $\boldsymbol{X}_l$ is given by $Z_l = argmax_k\ q(Z_l = k)$, which will be used as pseudo labels in the following semi-supervised deep learning procedure.

5.3.2 Semi-supervised Deep Learning using Pseudo Labels (PL-SSDL)

Once we have acquired the pseudo labels $\boldsymbol{Z}$ for $\boldsymbol{X} = \{\boldsymbol{X}_L, \boldsymbol{X}_U\}$ as discussed in Section 5.3.1, we can use $\{\boldsymbol{X}, \boldsymbol{Z}\}$ to train a deep CRNN, which is denoted as CRNN1 in Fig. 5.2. This CRNN aims to classify samples in $\boldsymbol{X}$ according to their pseudo labels $\boldsymbol{Z}$. Although samples from the same class may have different pseudo labels due to within-class variation, samples from different classes tend to have different pseudo labels. In other words, the pseudo labels are mostly consistent with the unknown true labels in discriminating different classes. Thus the discriminative information can still be preserved by the features extracted by the hidden layers of CRNN1 trained using $\{\boldsymbol{X}, \boldsymbol{Z}\}$.

Then we construct a new CRNN, denoted as CRNN2 in Fig. 5.2, which has two parts: the first part has the same architecture as CRNN1 without the output layer, and the second part consists of two fully connected layers followed by a classification layer (i.e., Softmax classifier). Then we fine-tune CRNNs using only the labeled data $\{\boldsymbol{X}_L, \boldsymbol{Y}_L\}$. In our implementation, the parameters of the first part of CRNN2 is initialized using the parameters of the pre-trained CRNN1 and will be frozen during training. Fine-tuning CRNN2 only updates the parameters of the second part since we only have a limited number of samples in $\boldsymbol{X}_L$.

The pre-training and fine-tuning stages for this PL-SSDL framework are illustrated in Fig. 5.4.

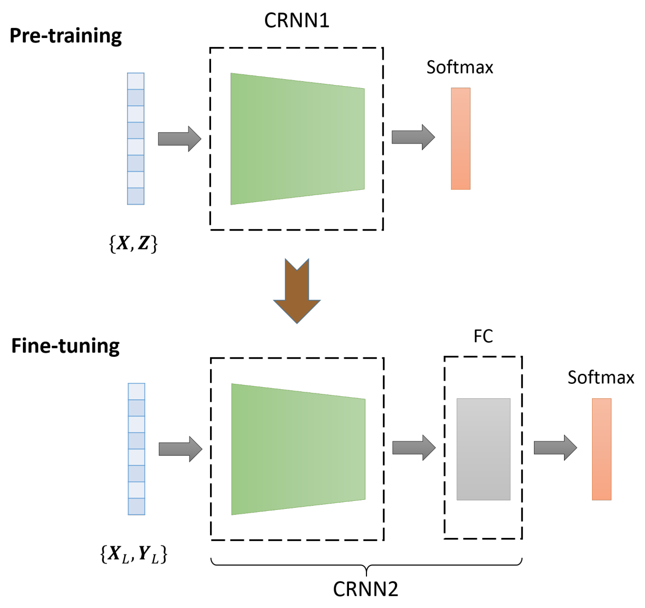


Figure 5.4: The pre-training and fine-tuning stages for the proposed PL-SSDL framework.

5.4 Experiments

5.4.1 Datasets

Three real-world hyperspectral image datasets are used to validate the efficacy of the proposed method and other state-of-the-art algorithms in our study. The first and second hyperspectral image datasets, the University of Pavia (UP) and the University of Houston (UH), have been described in Section 2.6.

The third dataset, called "Wetland", is an airborne hyperspectral image collected using the ProSpecTIR VS sensor over a wetland in Galveston, Texas in 2015. A true color image of this hyperspectral image is shown in Fig. 5.5. This dataset contains 17 labeled land cover classes (12 vegetation classes and 5 other classes) and consists of 360 spectral bands spanning the VNIR and SWIR spectral range from 400 nm to 2450 nm at a 5 nm spectral resolution. The image has a spatial dimension of 3462 $\times$ 5037 pixels at a 1 m spatial resolution.

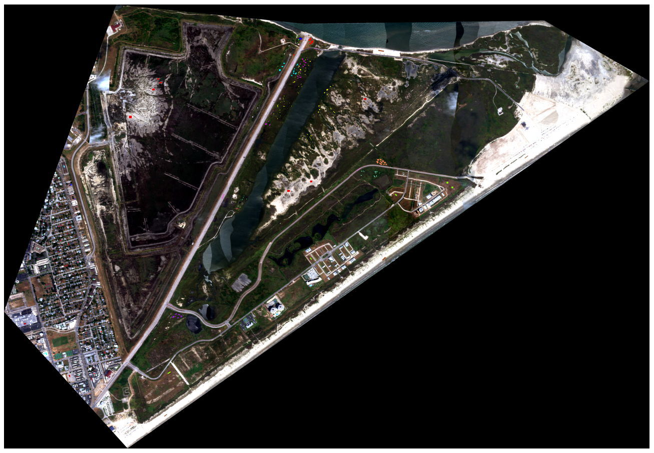


Figure 5.5: True color image of the Wetland hyperspectral dataset.

5.4.2 Experimental Setup and Results

In our experiments, the labeled set $\boldsymbol{X}_L$ contains 10 labeled pixels per class randomly selected from the labeled pixels on the hyperspectral image. A test set which includes 100 samples per class is used for validating different methods for classification. Practically, the unlabeled set $\boldsymbol{X}_U$ can include all pixels from the whole image but excluding $\boldsymbol{X}_L$ and $\boldsymbol{X}_T$. But we should note that unlabeled pixels from the whole image may contain some other background classes that are not present in the labeled pixels. We select $\boldsymbol{X}_U$ in this way because this is what we usually do in practical applications where we are given an image with some labeled pixels and all the other pixels can be used as unlabeled data for semi-supervised learning. In our experiments, to reduce the computation time, 50000 samples from the whole image excluding $\boldsymbol{X}_L$ and $\boldsymbol{X}_T$ are selected to form the unlabeled set $\boldsymbol{X}_U$.

For the C-DPMM in pseudo label generation, we get the spatial must-link constraints from superpixels generated by the Entropy Rate algorithm [127]. In the experiments, the number of superpixels selected for the UP, UH and the Wetland datasets are 2000, 10000 and 50000 respectively.

While pre-training CRNN1, 10% of the training samples in $\{\boldsymbol{X}, \boldsymbol{Z}\}$ are used as a validation set to learn the hyperparameters of the network (i.e., layer size, number of layers and learning rate) using a grid search strategy. The training process starts with the weights of all layers randomly initialized and the initial learning rate is set to 10^{-4}. Updating the parameters of the network is done by the mini-batch stochastic gradient descent algorithm with momentum and a batch size of 128. We run gradient descent for 1000 epochs and the learning rate decays by half every 200 epochs. CRNN2 copied the hidden layers of the pre-trained CRNN1 and adds two more randomly initialized fully connected layers after them. Fine-tuning CRNN2 using labeled set $\{\boldsymbol{X}_L, \boldsymbol{Y}_L\}$ only updates the parameters of the newly added fully connected layers because we only have limited amount of labeled samples. The fine-tuning process runs the mini-batch stochastic gradient descent for 1000 epochs with a initial learning rate set to 10^{-4} which will decay by half every 200 epochs. The configurations of the CRNNs used for all datasets are summarized in Tables 5.1, where input layers are denoted as "input-dimension", convolutional layers are denoted as "conv⟨receptive field size⟩-⟨number of filters⟩", recurrent layers are denoted as "recur-⟨feature dimension⟩", fully connected layers are denoted as "fc-⟨feature dimension⟩", and the output layers are denoted as "output-number of classes". We implemented the neural networks using the TensorFlow [114] and Keras [115] framework. Experiments are carried out on a workstation with a 3.0 GHz Intel(R) Core i7-5960X CPU, and an NVIDIA(R) GeForce Titan X GPU.

We compared the proposed method PL-SSDL with other state-of-the-art methods including supervised methods such as kNN and SVM (with RBF kernel), and semi-supervised methods such as Label Propagation [129], TSVM [119], LapSVM [120], SS-LapSVM [121], Stacked Denoising Autoencoder (SDA) [122] and Ladder Networks [125]. Label propagation is a graph based semi-supervised learning algorithm which propagate labels through the

Table 5.1: Summary of network configurations for Different Datasets.

UP	UH	Wetland
input-103	input-144	input-360
conv3-32	conv3-32	conv10-32
maxpool	maxpool	maxpool
conv3-32	conv3-32	conv10-32
maxpool	maxpool	maxpool
conv3-64	conv3-64	conv5-64
conv3-64	maxpool	maxpool
maxpool	conv3-64	conv5-64
recur-256	maxpool	maxpool
recur-512	recur-256	conv5-64
fc-64	recur-512	maxpool
fc-64	fc-64	recur-64
softmax-9	fc-64	recur-128
	softmax-15	recur-256
		fc-64
		fc-64
		softmax-17

dataset along high density areas defined by unlabeled data. TSVM exploits specific iterative algorithms that gradually search a reliable separating hyperplane (in the kernel space) with a transductive process that incorporates both labeled and unlabeled samples in the training phase. LapSVM is another semi-supervised extension of standard SVMs, which introduces an additional regularization term on the geometry of both labeled and unlabeled samples by using the graph Laplacian [130]. SS-LapSVM extends LapSVM by taking the spatial information of hyperspectral images into account. SDA per se is an unsupervised neural network which aims to reconstruct the input data from the hidden features it extracts. It can be used in a semi-supervised framework in a similar way to our proposed method. We first pre-train a SDA using labeled and unlabeled data and the features extracted by the this model are good representations for the input data. Then we construct another neural network by adding a few more fully connected layers and a classification layer after the pre-trained model, and fine-tune it using only the labeled data. Ladder networks was a recently proposed method for semi-supervised deep learning which simultaneously minimize a supervised and an unsupervised cost during training. The model parameters for SVM, Label Propagation, TSVM and LapSVM are selected based on the grid search cross-validation strategy. SDA and Ladder Propagation are trained using backpropagation and their hyperparameters (e.g., number of layers and size of each layer) are selected using 10% of the training data as the validation set. The size of each hidden layer in the pre-trained SDA is selected as: 64-48-32 (for UP dataset), 128-64-32 (for UH dataset), and 128-96-64 (for Wetland dataset). When fine-tuning, we added two more fully connected layers (128-128) after the hidden layers of the pre-trained network. The architecture (from the input layer to the output layer) of the Ladder Networks used in our experiments is selected as: 103-150-100-50-9 (for UP dataset), 144-150-100-50-14 (for UH dataset), and 360-200-150-100-50-17 (for Wetland dataset). The classification performances on the three datasets for all methods are presented in Tables 5.2, 5.3 and 5.4 respectively where each experiment is repeated 10 times with randomly selected training and test sets, and the average accuracy with the standard deviation is reported.

Table 5.2: Classification results for different methods on the University of Pavia dataset.

Method	Accuracy
kNN	73.76(1.64)
SVM	78.91(1.76)
Label Propagation	74.34(1.32)
TSVM	79.07(1.48)
LapSVM	80.09(1.54)
SS-LapSVM	81.17(1.88)
SDA	79.07(2.21)
Ladder Networks	79.58(1.35)
PL-SSDL	88.43(1.91)

The classification results in Tables 5.2, 5.3 and 5.4 show that our proposed method, PL-SSDL, achieved the best performance for all datasets. In general, when we are given only

Table 5.3: Classification results for different methods on the University of Houston dataset.

Method	Accuracy
kNN	73.52(1.41)
SVM	77.51(2.08)
Label Propagation	73.07(1.04)
TSVM	78.63(2.48)
LapSVM	79.15(2.67)
SS-LapSVM	80.24(1.52)
SDA	77.57(1.19)
Ladder Networks	80.29(1.71)
PL-SSDL	82.61(1.23)

Table 5.4: Classification results for different methods on the Wetland dataset.

Method	Accuracy
kNN	83.47(1.34)
SVM	89.08(2.06)
Label Propagation	89.28(1.07)
TSVM	92.24(0.81)
LapSVM	94.08(0.67)
SS-LapSVM	95.17(0.85)
SDA	90.05(1.81)
Ladder Networks	93.17(1.49)
PL-SSDL	97.33(0.48)

a small number of the labeled data, semi-supervised learning methods are better than the supervised methods because they make use of the abundant unlabeled data during training. We should note that since the unlabeled data $\boldsymbol{X}_U$ used in our experiments are pixels from the whole image which contains some other background classes that are not present in $\boldsymbol{X}_L$ and $\boldsymbol{X}_T$, the performance of all the semi-supervised methods may get badly affected. However, this won't be a big problem for our proposed method since the background classes are usually different from the labeled classes in our library. Thus the pseudo labels for the background classes are different from the pseudo labels for the labeled classes, and the pre-trained network can still extract the useful information from the unlabeled data.

Both SDA and the proposed PL-SSDL use unlabeled data for pre-training, but they differ in the way of using the unlabeled data. For SDA, the pre-training aims to reconstruct the input data from features extracted by the hidden layer. The features extracted by the SDA preserve the most important information for reconstruction but they may not be useful for classification. For PL-SSDL, the pre-training stage aims to discriminate samples according to their pseudo labels, which are consistent to their true labels (although unknown), i.e., samples belonging to different classes tend to have different pseudo labels. Thus the pre-trained network is able to extract discriminative information that are important for classification. That's why PL-SSDL achieved much better classification performances than the SDA.

The classification performance of the proposed PL-SSDL framework depends on the quality of the pseudo labels, which are generated by the C-DPMM based clustering algorithm. To understand the benefits brought by the pairwise constraints in C-DPMM, we can evaluate the clustering quality of both DPMM and C-DPMM using Normalized Mutual Information (NMI) [81]. We run this experiment using 10 labeled samples per class and 200 unlabeled samples per class randomly chosen from the ground truth data pool. The clustering experiments are repeated 10 times and the average number of clusters and average values of NMI with standard deviation for DPMM and C-DPMM with the University of Houston dataset are reported in Table 5.5. It is clear that the clustering quality for C-DPMM is better than DPMM, implying that the pseudo labels generated by C-DPMM are more consistent to the true labels than those generated by DPMM. With pseudo labels of higher quality, the pre-trained model is able to extract more discriminative features.

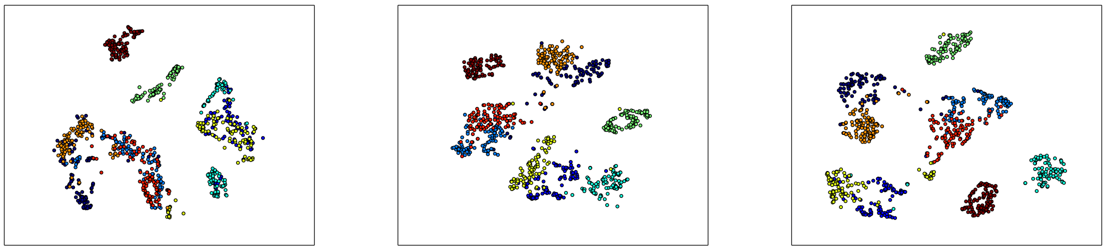
To better understand the efficacy of the pre-trained neural network (CRNN1) using pseudo labels, we can use use t-SNE [116] to reduce the dimensionality of features learned by the last hidden layer of the pre-trained CRNN for the test data $\boldsymbol{X}_T$ to 2. Then we can visualize the 2-dimensional representation, which is shown in Fig. 5.6 (b) where different

Table 5.5: Comparison of clustering performance between DPMM and C-DPMM for the University of Houston Dataset.

Method	Number of Clusters	NMI
DPMM	20.3(1.2)	83.15(1.23)
C-DPMM	24.1(1.6)	89.91(1.09)

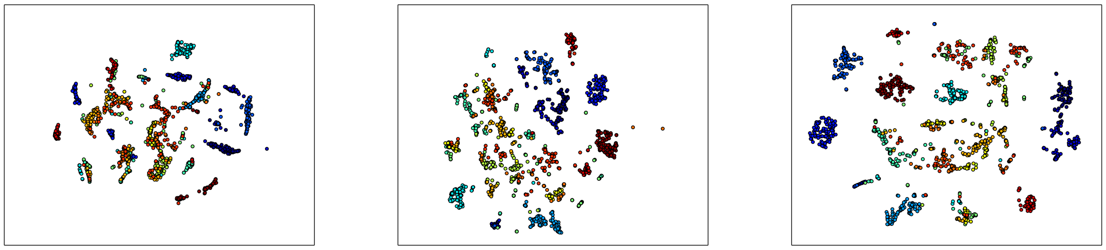
colors stands for different classes in the University of Pavia dataset. Similarly, we can apply t-SNE to the features extracted by the last hidden layer of the fine-tuned network (CRNN2), as shown in Fig. 5.6 (c). For comparison, we also apply t-SNE to the raw data $\boldsymbol{X}_T$ and show them in Fig. 5.6 (a). Similarly, we visualize the input data and features extracted by the neural networks for the University of Houston dataset and the Wetland dataset in Fig. 5.7 and Fig. 5.8. As we can observe from these visualizations, the deep features extracted by the two networks (CRNN1 and CRNN2) are much more discriminative than the original hyperspectral data, which means that the separability between different classes in the feature space of the deep features is significantly improved compared to the input feature space. Furthermore, since CRNN2 is fine-tuned using the labeled data, the separability for its features between different classes is improved a little bit compared to the features extracted by the pre-trained CRNN1.

To study the influence of the depth (number of convolutional and recurrent layers) of the pre-trained CRNN1 on the classification performance of the fine-tuned CRNN2, we plot the classification accuracy of CRNN2 on the UH dataset as a function of the depth of the pre-trained model, which ranges from 2 to 6, in Fig. 5.9. On the horizontal axis of Fig. 5.9, 2 represents two convolutional layers, 3 represents two convolutional layers and one recurrent layer, 4 represents four convolutional layers, 5 represents four convolutional



(a) (b) (c)

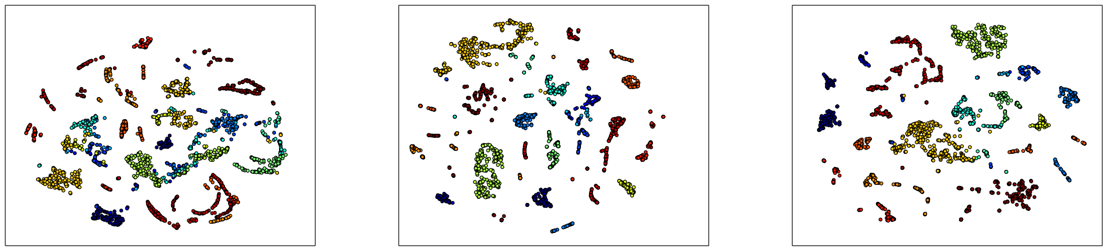
Figure 5.6: University of Pavia dataset: two-dimensional embeddings extracted by t-SNE for (a) input data, and features extracted by: (b) CRNN1 (the pre-trained model) and (c) CRNN2 (the fine-tuned model).



(a) (b) (c)

Figure 5.7: University of Houston dataset: two-dimensional embeddings extracted by t-SNE for (a) input data, and features extracted by: (b) CRNN1 (the pre-trained model) and (c) CRNN2 (the fine-tuned model).

layers and one recurrent layer, and 6 represents four convolutional layers and two recurrent layers. Similarly, we show classification accuracy as a function of depth of the pre-trained model for UP and Wetland dataset in Fig. 5.10 and 5.11. On the horizontal axis of Fig. 5.11, 3 represents three convolutional layers, 4 represents three convolutional layers and one recurrent layer, 5 represents five convolutional layers, 6 represents five convolutional layers and one recurrent layer, 7 represents five convolutional layers and two recurrent layers, and 8 represents five convolutional layers and three recurrent layers. From these figures, it is easy to observe that deeper pre-trained models are more beneficial to the final performances of the fine-tuned models. This is because deeper pre-trained models can be trained to extract



(a) (b) (c)

Figure 5.8: Wetland dataset: two-dimensional embeddings extracted by t-SNE for (a) input data, and features extracted by: (b) CRNN1 (the pre-trained model) and (c) CRNN2 (the fine-tuned model).

more discriminative features than shallower models.

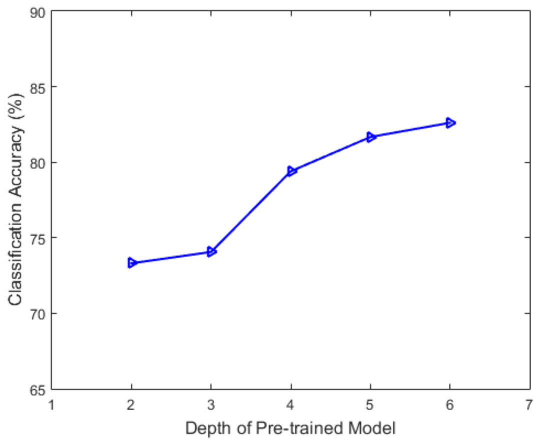


Figure 5.9: UH dataset: Classification performance as a function of the depth of the pre-trained model.

5.5 Conclusions

In this Chapter, we proposed a novel semi-supervised deep learning framework – PL-SSDL. Our method make use of the unlabeled data with pseudo labels generated by the C-DPMM based clustering algorithm. Since the pseudo labels preserved the differences between samples belonging to difference classes, pre-training a deep CRNN using data with pseudo labels can help us extract discriminative features which are useful for classification. Fine-tuning the second deep CRNN will further adjust the features from the pre-trained

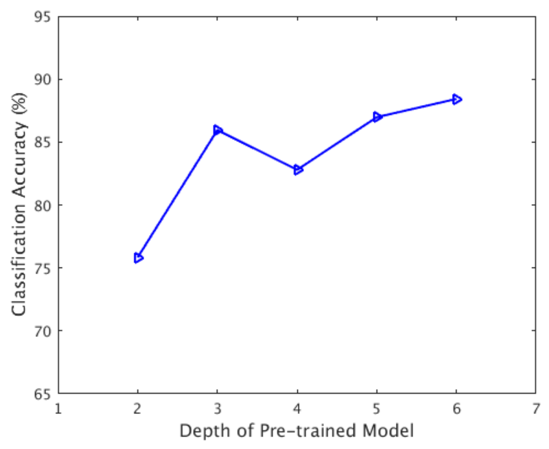


Figure 5.10: UP dataset: Classification performance as a function of the depth of the pre-trained model.

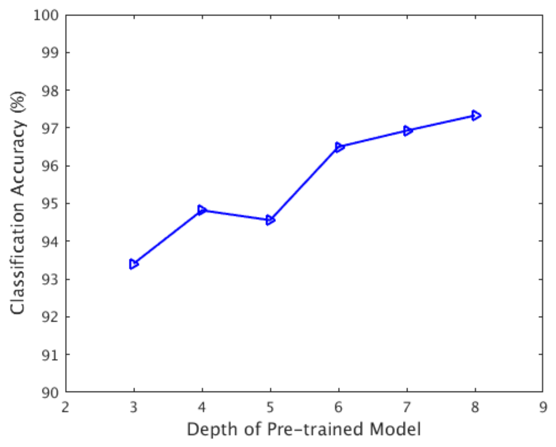


Figure 5.11: Wetland dataset: Classification performance as a function of the depth of the pre-trained model.

CRNN to make them more beneficial to classification. Experimental results have shown that the proposed method significantly outperforms the other state-of-the-art supervised and semi-supervised methods on three practical hyperspectral imagery datasets. This semi-supervised deep learning framework can be very meaningful in practical remote sending applications where we are usually given hyperspectral images with very limited pixels labeled, which makes it quite difficult to train a complicated model (such as a deep neural network) without suffering from severe overfitting problems. By making use of the abundant unlabeled pixels from the hyperspectral images, the proposed semi-supervised learning method is able to train deep neural networks effectively.

Chapter 6

Summary and Conclusion

In this dissertation, we proposed several semi-supervised and deep learning methods for hyperspectral image analysis. The main contributions of this dissertation are summarized as follows:

- To perform dimensionality reduction on hyperspectral data with limited labeled samples, a semi-supervised dimensionality reduction algorithm called SSLFDA and its kernel extension are proposed in Chapter 2, aiming to perform discriminant analysis on both labeled and unlabeled samples. SPLFDA performs discriminative analysis on unlabeled data by utilizing the pseudo labels generated by the Dirichlet process mixture model (DPMM). This method avoids the overfitting problems for supervised dimensionality reduction algorithms and has profound benefits for various applications when we don't have enough labeled samples for training but the unlabeled samples are abundant.

- An active learning system with a new query strategy called local information density (LID) is proposed in Chapter 3, which provides robust classification performance with minimum manual labeling effort while simultaneously discovering unknown classes. The unknown classes are discovered as new clusters when we apply DPMM-based clustering analysis on the whole dataset including both labeled and unlabeled samples. This is the first work that demonstrates a successful active learning paradigm that seeks out discovery of unseen classes, which has profound benefits for a variety of practical remote sensing applications, including image analysis of big geospatial data cubes, where it is often not possible to have every class on the ground represented in the initial training library.

- To perform classification on hyperspectral data when we have enough labeled samples for training, a deep convolutional recurrent neural network (CRNN) is designed in Chapter 4, which is the first work to extract contextual information from hyperspectral data via recurrent neural networks. By utilizing large amount of labeled training data, CRNN is able to extract more discriminative features and achieves better performances than existing methods. This is a very effective and practical method for hyperspectral data classification applications when we have access to large labeled training datasets.

- A deep sensor fusion system based on deep learning techniques is proposed in Chapter **??**, which combines hyperspectral and LiDAR data for land cover classification. The method has profound benefits for modern remote sensing applications where we have multiple data sources and enough training samples to train deep neural networks.

- To perform deep learning based classification under the circumstance that very limited labeled samples are provided but a large amount of unlabeled data are available, a new semi-supervised deep learning framework using pseudo labels is proposed in Chapter 5, which involves pre-training a deep neural network using labeled and unlabeled data with pseudo labels followed by fine-tuning using labeled data with true labels. This semi-supervised deep learning framework provides significant improvement on classification performances compared to state-of-the-art methods, and has profound benefits for real-world applications when we don't have enough labeled samples for training deep neural networks but the unlabeled samples are quite abundant.

References

[1] R. A. Schowengerdt, *Remote Sensing: Models and Methods for Image Processing.* Academic Press, 2006.

[2] J. R. Schott, *Remote Sensing.* Oxford University Press, 2007.

[3] T. Lillesand, R. W. Kiefer, and J. Chipman, *Remote Sensing and Image Interpretation.* John Wiley & Sons, 2014.

[4] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* John Wiley & Sons, 2012.

[6] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis," *The Journal of Machine Learning Research*, vol. 8, pp. 1027–1061, 2007.

[7] I. Jolliffe, *Principal Component Analysis.* Wiley Online Library, 2002.

[8] X. He and P. Niyogi, "Locality preserving projections," in *Advances in Neural Information Processing Systems*, 2004, pp. 153–160.

[9] D. Cai, X. He, and J. Han, "Semi-supervised discriminant analysis," in *IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–7.

[10] M. Sugiyama, T. Idé, S. Nakajima, and J. Sese, "Semi-supervised local fisher discriminant analysis for dimensionality reduction," *Machine Learning*, vol. 78, no. 1-2, pp. 35–61, January 2010.

[11] B. Settles, "Active learning literature survey," *Computer Sciences Technical Report 1648, University of Wisconsin–Madison*, 2009.

[12] D. Tuia, M. Volpi, L. Copa, M. Kanevski, and J. Munoz-Mari, "A survey of active learning algorithms for supervised remote sensing image classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 3, pp. 606–617, 2011.

[13] T. S. Ferguson, "A bayesian analysis of some nonparametric problems," *The Annals of Statistics*, pp. 209–230, 1973.

[14] C. E. Antoniak, "Mixtures of dirichlet processes with applications to bayesian nonparametric problems," *The Annals of Statistics*, pp. 1152–1174, 1974.

[15] C. E. Rasmussen, "The infinite gaussian mixture model," *Advances in Neural Information Processing Systems*, vol. 12, pp. 554–560, 2000.

[16] F. Wood, S. Goldwater, and M. J. Black, "A non-parametric bayesian approach to spike sorting," in *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2006, pp. 1165–1168.

[17] P. Orbanz and J. M. Buhmann, "Nonparametric bayesian image segmentation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 25–45, 2008.

[18] H. Wu and S. Prasad, "Infinite gaussian mixture models for robust decision fusion of hyperspectral imagery and full waveform lidar data," in *IEEE Global Conference on Signal and Information Processing*, 2013, pp. 1025–1028.

[19] H. Wu and S. Prasad, "Dirichlet process based active learning and discovery of unknown classes for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4882–4895, 2016.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[22] R. Girshick, "Fast r-cnn," in *IEEE International Conference on Computer Vision*, December 2015, pp. 1440–1448.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[25] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[26] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[27] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks." in *International Conference on Machine Learning*, vol. 14, 2014, pp. 1764–1772.

[28] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *arXiv preprint arXiv:1507.06947*, 2015.

[29] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[30] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[31] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[32] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[33] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, 2015.

[34] J. Yue, W. Zhao, S. Mao, and H. Liu, "Spectral–spatial classification of hyperspectral images using deep convolutional neural networks," *Remote Sensing Letters*, vol. 6, no. 6, pp. 468–477, 2015.

[35] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1349–1362, 2016.

[36] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4544–4554, 2016.

[37] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016.

[38] X. Zhu, "Semi-supervised learning literature survey," Department of Computer Science, University of Wisconsin-Madison, Tech. Rep., 2005.

[39] Y. Song, F. Nie, C. Zhang, and S. Xiang, "A unified framework for semi-supervised dimensionality reduction," *Pattern Recognition*, vol. 41, no. 9, pp. 2789–2799, 2008.

[40] F. Nie, S. Xiang, Y. Jia, and C. Zhang, "Semi-supervised orthogonal discriminant analysis via label propagation," *Pattern Recognition*, vol. 42, no. 11, pp. 2615–2627, 2009.

[41] M. Zhao, Z. Zhang, T. W. Chow, and B. Li, "A general soft label based linear discriminant analysis for semi-supervised dimensionality reduction," *Neural Networks*, vol. 55, pp. 83–97, 2014.

[42] S. Wang, J. Lu, X. Gu, H. Du, and J. Yang, "Semi-supervised linear discriminant analysis for dimension reduction and classification," *Pattern Recognition*, vol. 57, pp. 179–189, 2016.

[43] A. K. Jain, “Data clustering: 50 years beyond k-means,” *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[44] R. M. Neal, “Probabilistic inference using markov chain monte carlo methods,” *University of Toronto, Tech. Rep. CRG-TR-93-1*, 1993.

[45] R. M. Neal, “Markov chain sampling methods for dirichlet process mixture models,” *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 249–265, 2000.

[46] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, “An introduction to variational methods for graphical models,” *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[47] D. M. Blei and M. I. Jordan, “Variational inference for dirichlet process mixtures,” *Bayesian Analysis*, vol. 1, no. 1, pp. 121–143, 2006.

[48] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *arXiv preprint arXiv:1601.00670*, 2016.

[49] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, “Unsupervised feature selection using nonnegative spectral analysis,,” in *AAAI Conference on Artificial Intelligence*, 2012.

[50] M. Qian and C. Zhai, “Robust unsupervised feature selection,,” in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[51] J. Tang, X. Hu, H. Gao, and H. Liu, “Discriminant analysis for unsupervised feature selection,” in *Proceedings of the SIAM International Conference on Data Mining*, April 2014, pp. 938–946.

[52] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed Problems.* W.H. Winston, 1977.

[53] J. Sethuraman, “A constructive definition of dirichlet priors,” *Statistica Sinica*, vol. 4, no. 2, pp. 639–650, 1994.

[54] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[55] K. Kurihara, M. Welling, and N. A. Vlassis, “Accelerated variational dirichlet process mixtures,” in *Advances in Neural Information Processing Systems*, 2006, pp. 761–768.

[56] B. Schölkopf, A. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[57] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2001.

[58] P. Gamba, “A collection of data for urban area characterization,” in *IEEE International Geoscience and Remote Sensing Symposium*, September 2004, pp. 69–72.

[59] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, “Active learning methods for remote sensing image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2218–2232, 2009.

[60] S. Rajan, J. Ghosh, and M. M. Crawford, “An active learning approach to hyperspectral data classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 4, pp. 1231–1242, 2008.

[61] S. Tong and E. Chang, “Support vector machine active learning for image retrieval,” in *Proceedings of the 9th ACM International Conference on Multimedia*, 2001, pp. 107–118.

[62] D. Hakkani-Tur, G. Riccardi, and A. Gorin, “Active learning for automatic speech

recognition," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, 2002, pp. 3904–3907.

[63] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 1070–1079.

[64] A. McCallum and K. Nigam, "Employing em and pool-based active learning for text classification," in *Proceedings of the 15th International Conference on Machine Learning*, 1998, pp. 350–358.

[65] M. D. Escobar and M. West, "Bayesian density estimation and inference using mixtures," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 577–588, 1995.

[66] Y. W. Teh, "Dirichlet process," in *Encyclopedia of Machine Learning.* Springer, 2010, pp. 280–287.

[67] H. Wu, S. Prasad, M. Cui, N. T. Nguyen, and Z. Han, "Hyperspectral image classification based on dirichlet process mixture models," in *IEEE International Geoscience and Remote Sensing Symposium*, 2013, pp. 1043–1046.

[68] A. Dubey, S. Hwang, C. Rangel, C. E. Rasmussen, Z. Ghahramani, and D. L. Wild, "Clustering protein sequence and structure space with infinite gaussian mixture models." in *Pacific Symposium on Biocomputing*, vol. 9, 2004, pp. 399–410.

[69] G. Jun and J. Ghosh, "Semisupervised learning of hyperspectral data with unknown land-cover classes," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 1, pp. 273–282, 2013.

[70] H. Wu, S. Prasad, and T. Priya, "Detecting new classes via infinite warped mixture models for hyperspectral image analysis," in *IEEE International Conference on Image Processing*, 2014, pp. 5027–5031.

[71] T. S. Haines and T. Xiang, "Active rare class discovery and classification using dirichlet processes," *International Journal of Computer Vision*, vol. 106, no. 3, pp. 315–331, 2014.

[72] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[73] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

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

[75] X. He and P. Niyogi, "Locality preserving projections," in *Advances in Neural Information Processing Systems*, vol. 16, 2003, pp. 153–160.

[76] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," *Journal of Shanghai University (English Edition)*, vol. 8, no. 4, pp. 406–424, 2004.

[77] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Proceedings of the 11th International Conference on Machine Learning*, 1994, pp. 148–156.

[78] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the ACM 5th Annual Workshop on Computational Learning Theory*, 1992, pp. 287–294.

[79] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *Advances in Neural Information Processing Systems*, 2008, pp. 1289–1296.

[80] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1, pp. 19–41, 2000.

[81] T. M. Cover and J. A. Thomas, *Elements of information theory.* John Wiley & Sons, 2012.

[82] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[83] 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, 1998.

[84] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8614–8618.

[85] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[86] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[87] D. Landgrebe, "Hyperspectral image data analysis," *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 17–28, 2002.

[88] G. Shaw and D. Manolakis, “Signal processing for hyperspectral image exploitation,” *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 12–16, 2002.

[89] D. Manolakis, C. Siracusa, and G. Shaw, “Hyperspectral subpixel target detection using the linear mixing model,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 7, pp. 1392–1409, 2001.

[90] C. Huang, L. Davis, and J. Townshend, “An assessment of support vector machines for land cover classification,” *International Journal of Remote Sensing*, vol. 23, no. 4, pp. 725–749, 2002.

[91] J. Li, P. R. Marpu, A. Plaza, J. M. Bioucas-Dias, and J. A. Benediktsson, “Generalized composite kernel framework for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 9, pp. 4816–4829, 2013.

[92] H. Wu and S. Prasad, “Dirichlet process based active learning and discovery of unknown classes for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4882–4895, 2016.

[93] P. W. Yuen and M. Richardson, “An introduction to hyperspectral imaging and its application for security, surveillance and target acquisition,” *The Imaging Science Journal*, vol. 58, no. 5, pp. 241–253, 2010.

[94] T. J. Malthus and P. J. Mumby, “Remote sensing of the coastal zone: an overview and priorities for future research,” *International Journal of Remote Sensing*, vol. 24, no. 13, pp. 2805–2015, Novenber 2003.

[95] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, “Greedy layer-wise training

of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, p. 153, 2007.

[96] Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, and Y. Chen, "Convolutional recurrent neural networks: Learning spatial dependencies for image representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 18–26.

[97] Y. Xiao and K. Cho, "Efficient character-level document classification by combining convolution and recurrent layers," *arXiv preprint arXiv:1602.00367*, 2016.

[98] M. Dalponte, L. Bruzzone, and D. Gianelle, "Fusion of hyperspectral and lidar remote sensing data for classification of complex forest areas," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 5, pp. 1416–1427, 2008.

[99] C. Debes, A. Merentitis, R. Heremans, J. Hahn, N. Frangiadakis, T. van Kasteren, W. Liao, R. Bellens, A. Pižurica, S. Gautama *et al.*, "Hyperspectral and lidar data fusion: Outcome of the 2013 grss data fusion contest," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2405–2418, 2014.

[100] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Transactions on geoscience and remote sensing*, vol. 42, no. 8, pp. 1778–1790, 2004.

[101] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using svms and morphological profiles," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 11, pp. 3804–3814, 2008.

[102] S. Matteoli, M. Diani, and G. Corsini, "A tutorial overview of anomaly detection in hyperspectral images," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 5–28, 2010.

[103] W. Y. Yan, A. Shaker, and N. El-Ashmawy, "Urban land cover classification using airborne lidar data: A review," *Remote Sensing of Environment*, vol. 158, pp. 295–310, 2015.

[104] X. Li, W. Zeng, and Y. Duan, "A hybrid approach for tree classification in airborne lidar data," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 2183–2187.

[105] X. Zhang, C. Glennie, and A. Kusari, "Change detection from differential airborne lidar using a weighted anisotropic iterative closest point algorithm," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 7, pp. 3338–3346, 2015.

[106] H. Wu and S. Prasad, "Infinite gaussian mixture models for robust decision fusion of hyperspectral imagery and full waveform lidar data," in *IEEE Global Conference on Signal and Information Processing*, 2013, pp. 1025–1028.

[107] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[108] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[109] J. A. Benediktsson and J. R. Sveinsson, "Multisource remote sensing data classification based on consensus and pruning," *IEEE transactions on Geoscience and Remote Sensing*, vol. 41, no. 4, pp. 932–936, 2003.

[110] H. Wu and S. Prasad, "Convolutional recurrent neural networks forhyperspectral data classification," *Remote Sensing*, vol. 9, no. 3, p. 298, 2017.

[111] J. Wagner, V. Fischer, M. Herman, and S. Behnke, "Multispectral pedestrian detection using deep fusion convolutional neural networks," in *European Symposium on Artificial Neural Networks*, 2016.

[112] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.

[113] E. Park, X. Han, T. L. Berg, and A. C. Berg, "Combining multiple sources of knowledge in deep cnns for action recognition," in *IEEE Winter Conference on Applications of Computer Vision*, 2016, pp. 1–8.

[114] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[115] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[116] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[117] H. Lyu, H. Lu, and L. Mou, "Learning a transferable change rule from a recurrent neural network for land cover change detection," *Remote Sensing*, vol. 8, no. 6, p. 506, 2016.

[118] L. Bruzzone, M. Chi, and M. Marconcini, "Transductive svms for semisupervised classification of hyperspectral data," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, vol. 1, 2005.

[119] L. Bruzzone, M. Chi, and M. Marconcini, "A novel transductive svm for semisupervised classification of remote-sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 11, pp. 3363–3373, 2006.

[120] L. Gómez-Chova, G. Camps-Valls, J. Munoz-Mari, and J. Calpe, "Semisupervised image classification with laplacian support vector machines," *IEEE Geoscience and Remote Sensing Letters*, vol. 5, no. 3, pp. 336–340, 2008.

[121] 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, 2014.

[122] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[123] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[124] M. Ranzato and M. Szummer, "Semi-supervised learning of compact document representations with deep networks," in *Proceedings of the 25th International Conference on Machine learning*, 2008, pp. 792–799.

[125] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 3546–3554.

[126] A. Vlachos, A. Korhonen, and Z. Ghahramani, "Unsupervised and constrained dirichlet process mixture models for verb clustering," in *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, 2009, pp. 74–82.

[127] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2097–2104.

[128] T. Priya, S. Prasad, and H. Wu, "Superpixels for spatially reinforced bayesian classification of hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 5, pp. 1071–1075, 2015.

[129] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon University, Tech. Rep., 2002.

[130] F. R. Chung, *Spectral Graph Theory.* American Mathematical Society, 1997.