Semi-Supervised Deep Learning Using Pseudo Labels for Hyperspectral Image Classification

Hao Wu, *Student Member, IEEE*, and Saurabh Prasad, *Senior Member, IEEE*

***Abstract*— Deep learning has gained popularity in a variety of computer vision tasks. Recently, it has also been successfully applied for hyperspectral image classification tasks. Training deep neural networks, such as a convolutional neural network for classification requires a large number of labeled samples. However, in remote sensing applications, we usually only have a small amount of labeled data for training because they are expensive to collect, although we still have abundant unlabeled data. In this paper, we propose semi-supervised deep learning for hyperspectral image classification—our approach uses limited labeled data and abundant unlabeled data to train a deep neural network. More specifically, we use deep convolutional recurrent neural networks (CRNN) for hyperspectral image classification by treating each hyperspectral pixel as a spectral sequence. In the proposed semi-supervised learning framework, the abundant unlabeled data are utilized with their pseudo labels (cluster labels). We propose to use all the training data together with their pseudo labels to pre-train a deep CRNN, and then fine-tune using the limited available labeled data. Further, to utilize spatial information in the hyperspectral images, we propose a constrained Dirichlet process mixture model (C-DPMM), a non-parametric Bayesian clustering algorithm, for semi-supervised clustering which includes pairwise must-link and cannot-link constraints—this produces high-quality pseudo-labels, resulting in improved initialization of the deep neural network. We also derived a variational inference model for the C-DPMM for efficient inference. Experimental results with real hyperspectral image data sets demonstrate that the proposed semi-supervised method outperforms state-of-the-art supervised and semi-supervised learning methods for hyperspectral classification.**

***Index Terms*— Convolutional recurrent neural networks, pseudo labels, constrained Dirichlet process mixture model.**

I. Introduction

In RECENT years, with the development of computing hardware like GPUs and the availability of large scale datasets, deep learning [1] techniques have gained a lot of success in a variety of machine learning tasks such as computer vision [2]–[5], speech recognition [6]–[9], and natural language processing [10]–[12] etc.

Manuscript received May 8, 2017; revised October 20, 2017; accepted October 30, 2017. Date of publication November 13, 2017; date of current version December 27, 2017. This work was supported by the NASA New Investigator (Early Career) Award under Grant NNX14AI47G. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Peter Tay. *(Corresponding author: Saurabh Prasad.)*

The authors are with the Hyperspectral Image Analysis Group, Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA (e-mail: saurabh.prasad@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2772836

Hyperspectral data, with its rich spectral information, have been playing a key role in remote sensed data analysis [13], [14] such as land cover classification [15]–[17] and anomaly detection [18]. Recently, it was shown that deep learning techniques provide robust performance for hyperspectral image analysis as it pertains to remotely sensed data. For example, stacked autoencoders were used to extract deep features from hyperspectral data in [19]. Convolutional neural networks (CNN) were utilized for hyperspectral image classification in [20]. In [21], 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) [22] were used to extract the contextual information (i.e. dependencies between different spectral bands) for hyperspectral image classification and achieved better performance than other deep learning architectures. 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 base model for hyperspectral image classification.

Deep neural networks have a large number of parameters to learn, which requires a large amount of labeled samples for training. For hyperspectral image classification problems in real-world remote sensing applications, the goal is to classify each pixel in the image given some labeled pixels for training. However, the labeled samples are usually very limited (e.g. 10 pixels per class) because labeling work is quite expensive and time-consuming. As a result, sparse representation based image analysis approaches have demonstrated to be very useful for hyperspectral image analysis in recent years [13], [14]. Further, we always have access to a large quantity of unlabeled data from the given hyperspectral images. Semi-supervised learning [23] techniques, which make use of both labeled and unlabeled data, have also been widely utilized in the classification of remotely sensed hyperspectral images. For instance, transductive Support Vector Machines (TSVM) were used in [24] and [25] for semi-supervised classification of hyperspectral data. Laplacian Support Vector Machines (LapSVM) [26] 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 [27] 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 [28] using unlabeled data in an unsupervised way, followed by supervised fine-tuning using labeled data [29], [30]. The concept of ladder networks [31] was a recently proposed method for semi-supervised learning which simultaneously minimize a supervised and an unsupervised cost during training.

Typically, we need a large set of labeled data to effectively train supervised deep neural networks. When training with limited labeled data, the deep networks will severely overfit the training data and not generalize well to unseen data. Since the objective function for training deep neural networks is non-convex and the stochastic gradient descent (SGD) algorithm only finds local minimum solutions, a good initialization can lead SGD to a better local minimum solution. 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, which helps us learn a good initialization of parameters and control overfitting. Pseudo labels [32] are defined as the cluster labels generated by a clustering algorithm, such as the Dirichlet process mixture model (DPMM) [33]–[35]. As DPMM is a non-parametric clustering algorithm, we are able to handle unknown background classes from the unlabeled samples, which are quite common in remote sensing applications. 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) [36] which adds pairwise must-link constraints enforced by superpixels [37], [38] 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. Label Propagation [39], TSVM [25], LapSVM [26], SS-LapSVM [27], Stacked Denoising Autoencoder (SDA) [28] and Ladder Networks [31]. Compared to the conventional methods [25]–[27], [39] for semi-supervised learning, the proposed method employs deep neural networks for feature extraction, which can utilize the large number of unlabeled data to extract more abstract and discriminative features. Compared to other deep learning based semi-supervised learning methods [28], [31], the proposed method makes better use of the unlabeled samples via their pseudo labels. The use of pseudo labels for pre-training is first proposed in this paper. With the pseudo labels, the pre-trained deep network is able to extract discriminative features even without any labeled data, which makes it a good initialization for the supervised deep network that is fine-tuned on limited labeled data. Another advantage of the proposed method is that it can better handle the unknown background classes which may exist in the unlabeled samples. The unknown background classes will pop up as new clusters during clustering, i.e. they will have different pseudo labels than the known classes. During pre-training, the unknown classes will not harm the discriminatility between the features of the known classes. Therefore, even if unknown classes exist in the unlabeled data, they will not hurt the performance of the proposed method.

The remainder of this paper is organized as follows. Section II provides a review of previous related work on DPMM and deep learning for hyperspectral data classification. The proposed semi-supervised deep learning method using pseudo labels is described in detail in Section III. The experimental datasets, setup and results are presented in Section IV. Section V summarizes the key contributions of this work and provides concluding remarks.

II. Related Work

A. DPMM Based Clustering

Clustering is an unsupervised learning task that separates data into different groups based on their similarities. Parametric clustering methods such as k-means and Gaussian mixture models assume the number of clusters is known a priori, which is usually set to be the number of classes under study. However, due to spatial variations caused by different light and atmosphere conditions, many classes in hyperspectral images have multi-modal distributions, so assuming the number of clusters to be the number of classes is not an optimal choice. The DPMM [33]–[35] is a non-parametric model for clustering, which has the ability to infer the number of clusters from the data. Thus it is more suitable for hyperspectral data clustering.

In a DPMM, the parameters of each mixture component are generated by a Dirichlet Process (DP) [40] parameterized by a base measure G_0 and a concentration parameter α. Each sample $\boldsymbol{x}_i$ is drawn independently in turn from the chosen component given the parameters in the following way:

$$\begin{aligned} G &\sim DP(\alpha, G_0), \\ \boldsymbol{\theta}_{z_i} &\sim G, \\ \boldsymbol{x}_i|\boldsymbol{\theta}_{z_i} &\sim p(\boldsymbol{x}_i|\boldsymbol{\theta}_{z_i}), \end{aligned} \tag{1}$$

where z_i denotes the label of the component with which the observation $\boldsymbol{x}_i$ is associated. A popular representation for

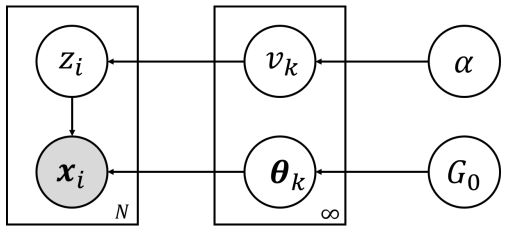


Fig. 1. A graphical model representation of the DPMM, where the shaded and unshaded nodes indicate observed and latent variables respectively, and plates indicate repetition.

the DPMM is called stick-breaking construction [41] which represents a draw G from a DP as

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\theta}_k}, \tag{2}$$

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 of the k^{th} component, which are produced in the following way:

$$v_k \sim \mathcal{B}(1, \alpha); \;\; \pi_k = v_k \prod_{l=1}^{k-1}(1 - v_l), \;\; k = 1, 2, \ldots, \infty, \tag{3}$$

where $\mathcal{B}$ is the Beta distribution, which is a family of continuous probability distributions defined on the interval [0, 1]. To generate a sample $\boldsymbol{x}_i$, the component assignment variable z_i is chosen from a multinomial distribution parameterized by $\boldsymbol{\pi} = \{\pi_k\}_{k=1}^{\infty}$. In this paper, a DPMM with Gaussian mixtures is used which has a likelihood function as

$$p\left(\boldsymbol{x}|\pi_k, \boldsymbol{\mu}_k, \boldsymbol{R}_k\right) = \sum_{k=1}^{\infty} \pi_k \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{R}_k^{-1}\right), \tag{4}$$

where $\boldsymbol{\mu}_k$ and $\boldsymbol{R}_k$ are the mean vector and precision matrix of the k^{th} component and we write $\boldsymbol{\theta}_k = \{\boldsymbol{\mu}_k, \boldsymbol{R}_k\}$ for simplicity. A Normal-Wishart distribution is used for the base distribution G_0 which is a conjugate prior for the likelihood in Eq. 4.

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

1. For $k = 1, 2, \ldots, \infty$:
   (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 $k = 1, 2, \ldots, \infty$:
   (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 $i = 1, 2, \ldots, N$:
   (a) Draw component assignment $z_i \sim Cat(\boldsymbol{\pi})$
   (b) Draw $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_{z_i}, \boldsymbol{R}_{z_i})$

The $Cat(\boldsymbol{\pi})$ in the generative process stands for a categorical distribution parameterized by $\boldsymbol{\pi}$. A graphical representation for this generative process is depicted in Fig. 1. In DPMM, α indirectly controls the number of clustering. Larger values of α usually result in more clusters. Each Gaussian component in DPMM is parameterized by the mean vector $\boldsymbol{\mu}_k$ and the precision matrix $\boldsymbol{R}_k$. $\boldsymbol{B}_0$ and ν_0 are parameters of the Wishart distribution, which is commonly used as the prior distribution for $\boldsymbol{R}_k$ in a multi-variate Gaussian distribution. Similarly, a Gaussian (Normal) distribution, parameterized by $\boldsymbol{\mu}_0$ and r_0, is usually used as the prior distribution for the mean vector $\boldsymbol{\mu}_k$.

The generative model for the DPMM described above generates observations given model parameters. For inference, given data $\boldsymbol{X}$, our goal is to infer the latent variables $\boldsymbol{Z} = \{z_1, z_2, \ldots, z_N\}$, and parameters $\boldsymbol{V} = \{v_1, v_2, \ldots\}$ and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots\}$. Markov chain Monte Carlo (MCMC) sampling approaches such as Gibbs sampling [42]–[44] can be used to infer the latent variables by obtaining samples from their posterior distribution. However, MCMC methods are slow to converge and their convergence is difficult to diagnose. To address this problem, variational inference (VI) methods [45], [46], 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 assumed to be mutually independent, each governed by a distinct factor, which can be written as [47]:

$$\begin{aligned} q(\boldsymbol{V}, \boldsymbol{\Theta}, \boldsymbol{Z}) &= q(\boldsymbol{V})q(\boldsymbol{\Theta})q(\boldsymbol{Z}) \\ &= \left[\prod_{t} q(v_t)\right]\left[\prod_{k} q(\boldsymbol{\theta}_k)\right]\left[\prod_{i=1}^{N} q(z_i)\right]. \end{aligned} \tag{5}$$

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. 5 as [47]:

$$q(\boldsymbol{V}, \boldsymbol{\Theta}, \boldsymbol{Z}) = \left[\prod_{t=1}^{T} q(v_t)\right]\left[\prod_{k=1}^{T} q(\boldsymbol{\theta}_k)\right]\left[\prod_{i=1}^{N} q(z_i)\right]. \tag{6}$$

The objective function, i.e., the KL divergence between the desired posterior and the variational distribution, can be expressed as [48]:

$$\begin{aligned} &D_{KL}\left[q(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z})||p(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z}|\boldsymbol{X})\right] \\ &= \sum_{\boldsymbol{Z}} \int_{\boldsymbol{\Theta}} \int_{\boldsymbol{V}} q(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z}) log \frac{q(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z})}{p(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z}|\boldsymbol{X})} \mathrm{d}\boldsymbol{V}\mathrm{d}\boldsymbol{\Theta} \\ &= -\sum_{\boldsymbol{Z}} \int_{\boldsymbol{\Theta}} \int_{\boldsymbol{V}} q(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z}) log \frac{p(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z})/p(\boldsymbol{X})}{q(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z})} \mathrm{d}\boldsymbol{V}\mathrm{d}\boldsymbol{\Theta} \\ &= -\sum_{\boldsymbol{Z}} \int_{\boldsymbol{\Theta}} \int_{\boldsymbol{V}} q(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z}) log \frac{p(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z})}{q(\boldsymbol{V}, \;\boldsymbol{\Theta}, \boldsymbol{Z})} \mathrm{d}\boldsymbol{V}\mathrm{d}\boldsymbol{\Theta} \\ &\quad + log P(\boldsymbol{X}) = F + log P(\boldsymbol{X}), \end{aligned} \tag{7}$$

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

$$\begin{aligned}F &= -\sum_{\boldsymbol{Z}}\int_{\boldsymbol{\Theta}}\int_{\boldsymbol{V}} q(\boldsymbol{V},\ \boldsymbol{\Theta},\boldsymbol{Z}) log\frac{p(\boldsymbol{V},\ \boldsymbol{\Theta},\boldsymbol{Z})}{q(\boldsymbol{V},\ \boldsymbol{\Theta},\boldsymbol{Z})}\mathrm{d}\boldsymbol{V}\mathrm{d}\boldsymbol{\Theta}\\ &= \mathbb{E}\left[log\frac{q(\boldsymbol{V},\ \boldsymbol{\Theta},\boldsymbol{Z})}{p(\boldsymbol{V},\ \boldsymbol{\Theta},\boldsymbol{Z})}\right]_{q(\boldsymbol{Z},\boldsymbol{V},\ \boldsymbol{\Theta})}\\ &= \sum_{k=1}^{T}\mathbb{E}\left[log\frac{q(\boldsymbol{\theta}_k)}{p(\boldsymbol{\theta}_k)}\right]_{q(\boldsymbol{\theta}_k)} + \sum_{k=1}^{T}\mathbb{E}\left[log\frac{q(v_k)}{p(v_k)}\right]_{q(v_k)} \quad (8)\\ &\quad + \sum_{i=1}^{N}\mathbb{E}\left[log\frac{q(z_i)}{p(\boldsymbol{x}_i|z_i,\boldsymbol{\Theta})p(z_i|\boldsymbol{V})}\right]_{q(z_i,\boldsymbol{V},\boldsymbol{\Theta})} \quad (9)\end{aligned}$$

The minimization of the free energy with respect to the variational distribution can be solved using the coordinate ascent variational inference (CAVI) algorithm [47], [48], which iteratively optimizes each factor in Eq. (5) while holding others fixed until the free energy converges. For more detail about this parameter inference process, readers can refer to [47]–[49].

B. CRNN

Deep neural networks such as CNNs, RNNs and CRNNs have been used for hyperspectral data classification successfully [20], [22]. 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 [22]. The CRNN is a hybrid model of convolutional and recurrent neural networks [22], [50], [51]. 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. 2. 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 $\boldsymbol{x} = (x_1, x_2, \ldots, x_T) \in \mathbb{R}^T$ where T is the length of the input vector. In the first convolutional layer, a set of d filters $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_d\}$ 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:

$$\boldsymbol{F} = (\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_T) = g(\boldsymbol{x} * \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_d\}) \quad (10)$$

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

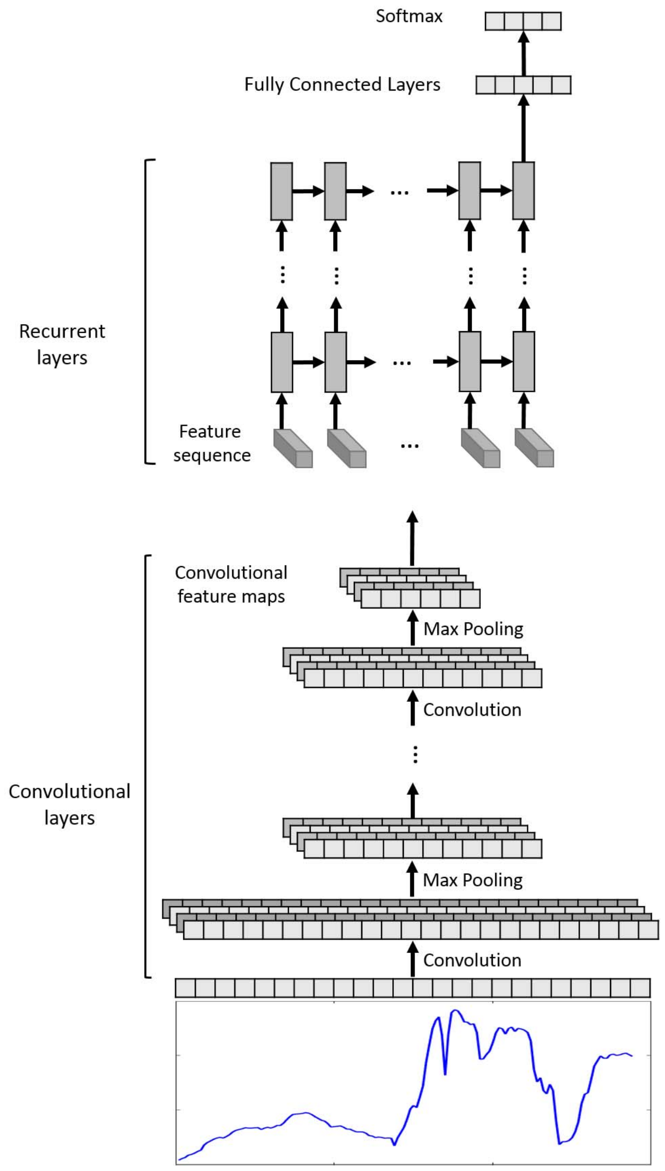


Fig. 2. Architecture of the CRNN consisting of multiple convolutional layers and recurrent layers for hyperspectral data classification.

the most widely used activation function in CNNs, and we also choose to use it in this work. The max pooling layer subsamples every depth slice of the input by a *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 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:

$$\boldsymbol{h}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = tanh(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1}) \quad (11)$$

and the outputs are calculates as:

$$o_t = softmax(\boldsymbol{V}\boldsymbol{h}_t) \tag{12}$$

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. 2, 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

$$p(y = k|\boldsymbol{x}) = \frac{exp(\boldsymbol{w}_k^\top \boldsymbol{x} + b_k)}{\sum_{k'=1}^{K} exp(\boldsymbol{w}_{k'}^\top \boldsymbol{x} + b_{k'})} \tag{13}$$

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:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{14}$$

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:

$$L_i = -log(p(y_i|\boldsymbol{x}_i)) \tag{15}$$

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 [52]. 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.

III. The Proposed Method

Section II 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. 3. 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.

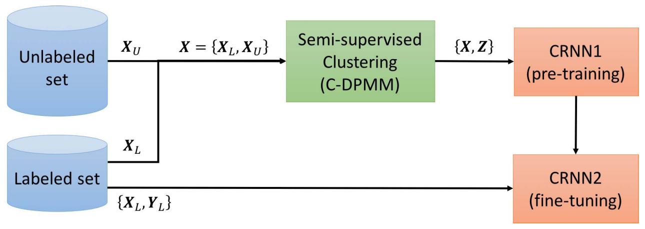


Fig. 3. The structure of the proposed semi-supervised deep learning framework.

A. Generation of Pseudo Labels Via C-DPMM

Pseudo labels [32] are defined as cluster assignments generated by some clustering algorithm. The DPMM based clustering described in Section II-A 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) [36] 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 [37], [38]. 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.

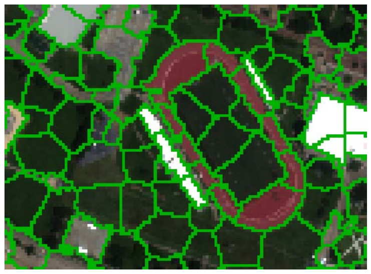


Fig. 4. A cropped region of the University of Houston hyperspectral image with superpixel segmentation.

These 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. 4. The Entropy Rate superpixel algorithm [37], [38] is used in this work, which solves the superpixel segmentation problem using an objective function based on the entropy rate of a random walk on the graph.

Compared to DPMM, the generative process of C-DPMM is modified in the following way [36]: must-linked samples are always generated by the same component, while cannot-linked samples are always by different ones. [36] derived an inference method based on Gibbs sampling. As discussed in Section II-A, 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. 6, the variational distribution for C-DPMM is modified as

$$q(\boldsymbol{V},\ \boldsymbol{\Theta}, \boldsymbol{Z}) = \left[\prod_{t=1}^{T} q(v_t)\right]\left[\prod_{k=1}^{T} q(\boldsymbol{\theta}_k)\right]\left[\prod_{l=1}^{L} q(Z_l|\mathcal{C})\right], \quad (16)$$

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

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

which can be further simplifies as

$$F = \sum_{k=1}^{T} \mathbb{E}\left[log \frac{q(\boldsymbol{\theta}_k)}{p(\boldsymbol{\theta}_k)}\right]_{q(\boldsymbol{\theta}_k)} + \sum_{k=1}^{T} \mathbb{E}\left[log \frac{q(v_k)}{p(v_k)}\right]_{q(v_k)} + \sum_{l=1}^{L} log \sum_{k=1}^{T} exp(S_{l,k}). \quad (17)$$

In Eq. 17, we have defined

$$S_{l,k} = \mathbb{E}\left[log\ \ p(\boldsymbol{X}_l|\boldsymbol{\theta}_k)\right]_{q(\boldsymbol{\theta}_k)} + \mathbb{E}\left[log\ \ p(Z_l = k|\boldsymbol{V})\right]_{q(\boldsymbol{V})}, \quad (18)$$

where

$$p(\boldsymbol{X}_l|\boldsymbol{\theta}_k) = \prod_{i=1}^{N_l} p(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{R}_k) \quad \text{and}$$

$$p(Z_l = k|\boldsymbol{V}) = \prod_{i=1}^{N_l} p(z_i = k|\boldsymbol{V}) = (v_k \prod_{j=1}^{k-1}(1-v_j))^{N_l}.$$

The free energy F is minimized using the CAVI algorithm by iteratively updating each factor in Eq. 16 until F converges. Using the properties of conjugate priors, the update rules for each factor in Eq. 16 can be calculated in the following way:

(a) For $t = 1, \ldots, T$:

$$q(v_t) = \mathcal{B}(a_{t1}, a_{t2}), \quad (19)$$

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

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

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

(b) For $k = 1, \ldots, T$:

$$q(\boldsymbol{\theta}_k) = \mathcal{NW}(r_k, \boldsymbol{m}_k, \nu_c, \boldsymbol{B}_c), \quad (20)$$

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

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

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

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

where we define

$$N_k = \sum_{i=1}^{N} q(z_i = k),$$

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

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

(c) For $l = 1, \ldots, L$:

$$q(Z_l = k) \propto \begin{cases} exp(S_{l,k}) & \text{if } \boldsymbol{X}_{k,-l} \cap \mathcal{C} = \emptyset \\ 0 & \text{otherwise,} \end{cases} \quad (21)$$

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

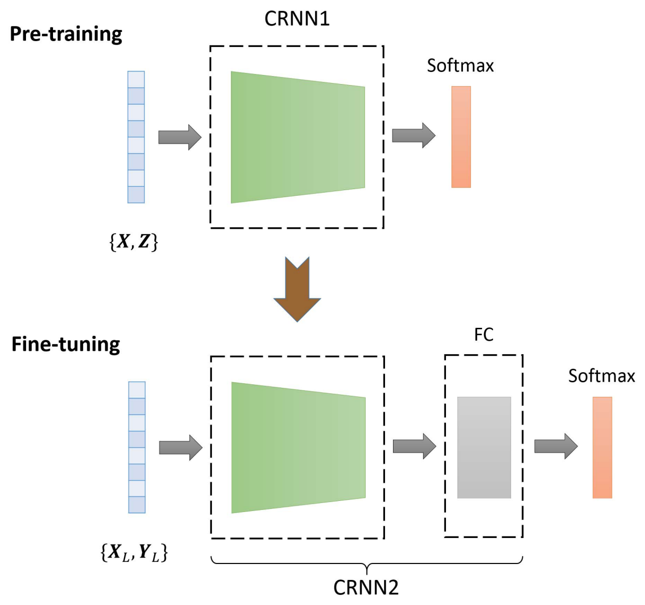


Fig. 5. The pre-training and fine-tuning stages for the proposed PL-SSDL framework.

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.

B. 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 III-A, we can use $\{\boldsymbol{X}, \boldsymbol{Z}\}$ to train a deep CRNN, which is denoted as CRNN1 in Fig. 3. 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 $\{\boldsymbol{X}, \boldsymbol{Z}\}$.

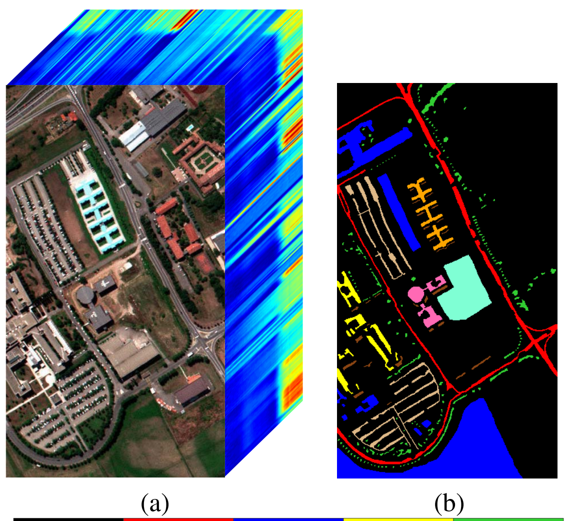
Then we construct a new CRNN, denoted as CRNN2 in Fig. 3, which has two parts: the first part has the same architecture as CRNN1 without the output layer, and the second part consists of two fully connected layers followed by a classification layer (i.e. Softmax classifier). Then we fine-tune CRNNs using 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.

IV. Experiments

A. 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.



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

Fig. 6. (a) True color image and (b) ground truth map of the University of Pavia hyperspectral imagery data.

The first hyperspectral image dataset, called “University of Pavia” (UP), was collected using the Reflective Optics System Imaging Spectrometer (ROSIS-3) sensor [53]. This dataset has 103 spectral bands collected within the 430 $nm - 860\ nm$ wavelength range. The image has a spatial size of 610 $\times$ 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. 6.

The second dataset used in this work, called “University of Houston” (UH), was acquired by the NSF-funded National Center for Airborne Laser Mapping (NCALM) over the University of Houston campus and the neighboring urban areas using the ITRES-CASI (Compact Airborne Spectrographic Imager) 1500 hyperspectral imager in 2012. This dataset contains 15 labeled land cover classes and consists of 144 spectral bands over the 364 $nm - 1046\ nm$ wavelength range. The image has a dimension of 1905 $\times$ 349 pixels with a spatial resolution of 2.5 m. Fig. 7 shows the true color image of this dataset overlaid with the ground truth map.

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. 8. 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.

B. 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

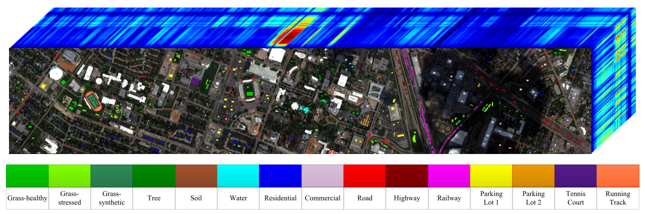


Fig. 7. True color image inset with ground truth for the University of Houston hyperspectral dataset.

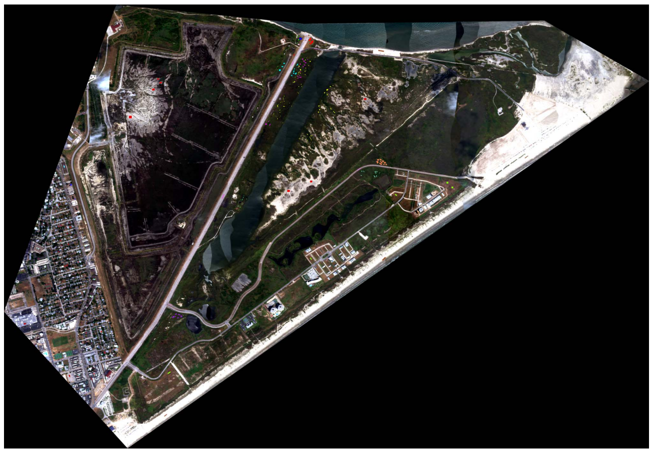


Fig. 8. True color image of the Wetland hyperspectral dataset.

TABLE I
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

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. In practical hyperspectral image classification applications, we are usually given an image with several labeled pixels (e.g. 10 30 labeled pixels per class) and the goal is to classify the unlabeled regions or new images. To do semi-supervised learning, we need a large amount of unlabeled data, which can be acquired from all the unlabeled samples from the image (usually we have lots of them, e.g. from tens of thousands to millions). That's how we select $\boldsymbol{X}_U$. 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 deep learning based methods. For traditional methods, $\boldsymbol{X}_U$ contains 10000 samples because we found that more samples won't help them improve the performance.

For the C-DPMM in pseudo label generation, we get the spatial must-link constraints from superpixels generated by the Entropy Rate algorithm [37], [38]. The Entropy Rate superpixel algorithm has two parameters: the number of superpixels K and the balancing parameter λ. As described in [37], the balancing parameter can be automatically adjusted. Thus the only parameter we need to set is K. In this work, we set K based on the spatial resolution and the spatial dimension of the hyperspectral image. For example, we tried to make each superpixel pure and keep the superpixel size not too small at the same time (e.g. we kept the average size of superpixels around 100 200 pixels). In our 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 a limited amount of labeled samples. The fine-tuning process runs the mini-batch stochastic gradient descent for 1000 epochs with an 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 I, where input layers are denoted as "input-dimension", convolutional layers are denoted as "conv$\langle$receptive field size$\rangle$-$\langle$number of filters$\rangle$", recurrent layers are denoted as "recur-$\langle$feature dimension$\rangle$", fully connected layers are denoted as "fc-$\langle$feature dimension$\rangle$", and the output layers are denoted as "output-number of classes". We implemented the neural networks using the TensorFlow [54] and Keras [55] framework. Experiments are carried out on a workstation with a 3.0 GHz Intel(R) Core i7-5960X CPU, and a 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 [39], TSVM [25], LapSVM [26], SS-LapSVM [27], Stacked Denoising Autoencoder (SDA) [28] and Ladder Networks [31].

Label propagation is a graph based semi-supervised learning algorithm which propagates labels through the 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 [56]. 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 an SDA using labeled and unlabeled data and the features extracted by this model are good representations of 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 II, III and IV 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. At the same time, we also show the training time for different approaches in Table II, III and IV. Note that since kNN doesn't have a regular training process, thus we put "NaN" in the tables.

TABLE II

CLASSIFICATION RESULTS FOR DIFFERENT METHODS ON THE UNIVERSITY OF PAVIA DATASET

Method	Accuracy	Time Complexity
kNN	73.76(1.64)	NaN
SVM	78.91(1.76)	3.1 s
Label Propagation	74.34(1.32)	31.2
TSVM	79.07(1.48)	5.5 min
LapSVM	80.09(1.54)	3.1 min
SS-LapSVM	81.17(1.88)	4.4 min
SDA	79.07(2.21)	8.3 min
Ladder Networks	79.58(1.35)	10.7 min
PL-SSDL (DPMM)	83.53(1.17)	42.1 min
PL-SSDL (C-DPMM)	88.43(1.91)	42.5 min

TABLE III

CLASSIFICATION RESULTS FOR DIFFERENT METHODS ON THE UNIVERSITY OF HOUSTON DATASET

Method	Accuracy	Time Complexity
kNN	73.52(1.41)	NaN
SVM	77.51(2.08)	5.9 s
Label Propagation	73.07(1.04)	37.7 s
TSVM	78.63(2.48)	8.8 min
LapSVM	79.15(2.67)	4.6 min
SS-LapSVM	80.24(1.52)	4.9 min
SDA	77.57(1.19)	11.2 min
Ladder Networks	80.29(1.71)	15.6 min
PL-SSDL (DPMM)	80.47(2.02)	54.1 min
PL-SSDL (C-DPMM)	82.61(1.23)	53.2 min

TABLE IV

CLASSIFICATION RESULTS FOR DIFFERENT METHODS ON THE WETLAND DATASET

Method	Accuracy	Time Complexity
kNN	83.47(1.34)	NaN
SVM	89.08(2.06)	6.7 s
Label Propagation	89.28(1.07)	1.6 min
TSVM	92.24(0.81)	37.5 min
LapSVM	94.08(0.67)	22.3 min
SS-LapSVM	95.17(0.85)	35.8 min
SDA	90.05(1.81)	16.4 min
Ladder Networks	93.17(1.49)	19.8 min
PL-SSDL (DPMM)	94.87(0.93)	79.5 min
PL-SSDL (C-DPMM)	97.33(0.48)	83.5 min

The classification results in Tables II, III and IV show that our proposed method, PL-SSDL, achieved the best performance for all datasets. Among the three hyperspectral datasets, all the algorithms performed better on the Wetland dataset than the other two datasets, which is due to the fact that the Wetland dataset has a very high spatial resolution and collected by more advanced sensors, which makes the data cleaner and easier for classification. In general, when we are given only 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

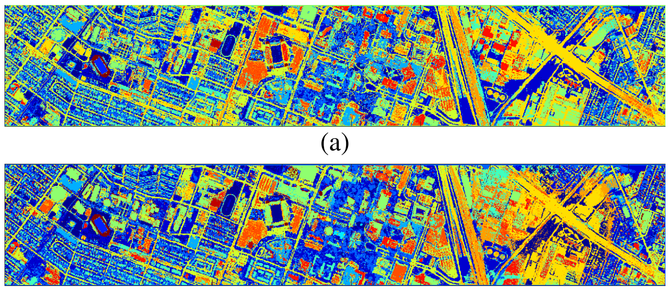

(a)
(b)

Fig. 9. Classification maps on the University of Houston dataset. (a) SS-LapSVM. (b) PL-SSDL.

TABLE V

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)

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 which is important for classification. That's why PL-SSDL achieved much better classification performances than the SDA.

The time complexity presented in Table II, III and IV shows that the proposed semi-supervised deep learning method requires more training time than the other approaches. However, this wont be a problem for practical remote sensing applications where the training process is done offline and we assume we have enough computing resource for it.

To get a visual interpretation of the classification performances, we display the classification maps for the proposed method and the best baseline approach (SS-LapSVM) on the UH dataset in Fig. 9. As we can see, the proposed method generates a better classification map than the SS-LapSVM. For example, the SS-LapSVM makes a lot errors in discriminating the buildings and the parking lot while the proposed method does a much better job.

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) [57]. 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 V. 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.

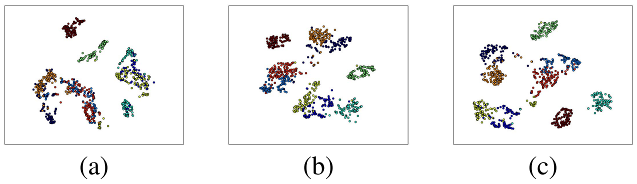

(a) (b) (c)

Fig. 10. 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).

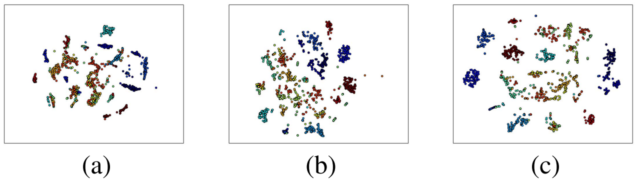

(a) (b) (c)

Fig. 11. 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).

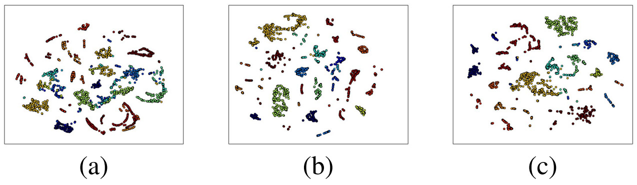

(a) (b) (c)

Fig. 12. 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).

To better understand the efficacy of the pre-trained neural network (CRNN1) using pseudo labels, we can use t-SNE [58] 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. 10 (b) where different colors stand 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. 10 (c). For comparison, we also apply t-SNE to the raw data $\boldsymbol{X}_T$ and show them in Fig. 10 (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. 11 and Fig. 12. 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 as a function of the depth of the pre-trained model in Fig. 13. The depth of the pre-trained model is defined differently for each dataset

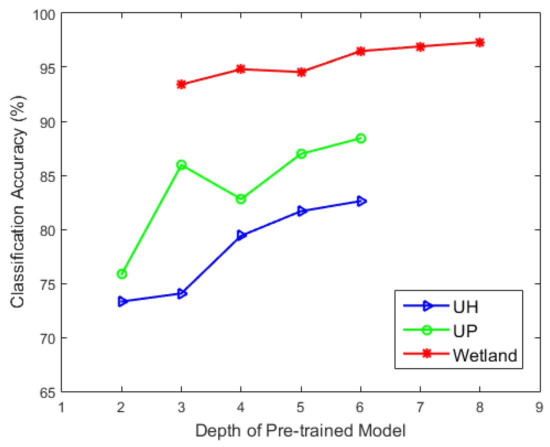


Fig. 13. Classification performance as a function of the depth of the pre-trained model. As described in the text, the depth of the pre-trained model is defined differently for each dataset.

as follows: For the UH and UP datasets, 2 represents two convolutional layers, 3 represents two convolutional layers and one recurrent layer, 4 represents four convolutional layers, 5 represents four convolutional layers and one recurrent layer, and 6 represents four convolutional layers and two recurrent layers. For the Wetland dataset, 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 this figure, 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 more discriminative features than shallower models.

V. Conclusions and Future Work

In this paper, we proposed a novel semi-supervised deep learning framework – PL-SSDL. Our method makes 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 different 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 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 sensing 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.

In the future, we plan to explore better ways for generating pseudo labels which are more consistent to the underlying true labels because the quality of the pseudo labels directly determines the quality of the features extracted by the deep networks and the final classification performance.

Acknowledgments

The authors thank Prof. Gamba for providing the University of Pavia dataset. We also thank Prof. Anna Armitage at Texas A&M University, Galveston, for helping with annotation of the wetland dataset.

References

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

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[4] K. Simonyan and A. Zisserman. (2014). “Very deep convolutional networks for large-scale image recognition.” [Online]. Available: https://arxiv.org/abs/1409.1556

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[6] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *Neural Netw.*, vol. 18, nos. 5–6, pp. 602–610, 2005.

[7] A. Graves, A.-R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[8] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *Proc. Int. Conf. Mach. Learn.*, vol. 14, 2014, pp. 1764–1772.

[9] H. Sak, A. Senior, K. Rao, and F. Beaufays. (2015). “Fast and accurate recurrent neural network acoustic models for speech recognition.” [Online]. Available: https://arxiv.org/abs/1507.06947

[10] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[11] K. Cho *et al.* (2014). “Learning phrase representations using rnn encoder-decoder for statistical machine translation.” [Online]. Available: https://arxiv.org/abs/1406.1078

[12] D. Bahdanau, K. Cho, and Y. Bengio. (2014). “Neural machine translation by jointly learning to align and translate.” [Online]. Available: https://arxiv.org/abs/1409.0473

[13] S. Prasad, D. Labate, M. Cui, and Y. Zhang, “Morphologically decoupled structured sparsity for rotation-invariant hyperspectral image analysis,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4355–4366, Aug. 2017.

[14] S. Prasad, D. Labate, M. Cui, and Y. Zhang, “Rotation invariance through structured sparsity for robust hyperspectral image classification,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 6205–6209.

[15] F. Melgani and L. Bruzzone, “Classification of hyperspectral remote sensing images with support vector machines,” *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[16] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, “Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles,” *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3804–3814, Nov. 2008.

[17] H. Wu and S. Prasad, “Dirichlet process based active learning and discovery of unknown classes for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4882–4895, Aug. 2016.

[18] S. Matteoli, M. Diani, and G. Corsini, “A tutorial overview of anomaly detection in hyperspectral images,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 25, no. 7, pp. 5–28, Jul. 2010.

[19] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, “Deep learning-based classification of hyperspectral data,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.

[20] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. Sensors*, vol. 2015, Jan. 2015. [Online]. Available: https://www.hindawi.com/journals/js/2015/258619/

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

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

[23] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1530, 2005.

[24] L. Bruzzone, M. Chi, and M. Marconcini, "Transductive SVMs for semisupervised classification of hyperspectral data," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, vol. 1. Jul. 2005, p. 4.

[25] L. Bruzzone, M. Chi, and M. Marconcin, "A novel transductive SVM for semisupervised classification of remote-sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 11, pp. 3363–3373, Nov. 2006.

[26] L. Gómez-Chova, G. Camps-Valls, J. Munoz-Mari, and J. Calpe, "Semisupervised image classification with Laplacian support vector machines," *IEEE Geosci. Remote Sens. Lett.*, vol. 5, no. 3, pp. 336–340, Jul. 2008.

[27] L. Yang, S. Yang, P. Jin, and R. Zhang, "Semi-supervised hyperspectral image classification using spatio-spectral Laplacian support vector machine," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 3, pp. 651–655, Mar. 2014.

[28] 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," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

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

[30] M. Ranzato and M. Szummer, "Semi-supervised learning of compact document representations with deep networks," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 792–799.

[31] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3546–3554.

[32] H. Wu and S. Prasad, "Semi-supervised dimensionality reduction of hyperspectral imagery using pseudo-labels," *Pattern Recognit.*, vol. 74, pp. 212–224, Feb. 2017.

[33] C. E. Antoniak, "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems," *Ann. Stat.*, vol. 2, no. 6, pp. 1152–1174, 1974.

[34] C. E. Rasmussen, "The infinite Gaussian mixture model," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12. 2000, pp. 554–560.

[35] H. Wu and S. Prasad, "Infinite Gaussian mixture models for robust decision fusion of hyperspectral imagery and full waveform lidar data," in *Proc. IEEE Global Conf. Signal Inf. (GlobalSIP)*, Dec. 2013, pp. 1025–1028.

[36] A. Vlachos, A. Korhonen, and Z. Ghahramani, "Unsupervised and constrained Dirichlet process mixture models for verb clustering," in *Proc. Workshop Geometrical Models Natural Lang. Semantics*, Mar. 2009, pp. 74–82.

[37] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 2097–2104.

[38] T. Priya, S. Prasad, and H. Wu, "Superpixels for spatially reinforced Bayesian classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 5, pp. 1071–1075, May 2015.

[39] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CALD-02-107, 2002.

[40] T. S. Ferguson, "A Bayesian analysis of some nonparametric problems," *Ann. Stat.*, vol. 1, no. 2, pp. 209–230, 1973.

[41] J. Sethuraman, "A constructive definition of Dirichlet priors," *Statistica Sinica*, vol. 4, no. 2, pp. 639–650, 1994.

[42] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, USA, Tech. Rep. CRG-TR-93-1, 1993.

[43] F. Wood, S. Goldwater, and M. J. Black, "A non-parametric Bayesian approach to spike sorting," in *Proc. IEEE 28th Annu. Int. Conf. Eng. Med. Biol. Soc.*, Sep. 2006, pp. 1165–1168.

[44] P. Orbanz and J. M. Buhmann, "Nonparametric Bayesian image segmentation," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 25–45, 2008.

[45] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, 1999.

[46] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. (2016). "Variational inference: A review for statisticians." [Online]. Available: https://arxiv.org/abs/1601.00670

[47] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Anal.*, vol. 1, no. 1, pp. 121–143, 2006.

[48] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag, 2006.

[49] K. Kurihara, M. Welling, and N. A. Vlassis, "Accelerated variational Dirichlet process mixtures," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 761–768.

[50] Z. Zuo *et al.*, "Convolutional recurrent neural networks: Learning spatial dependencies for image representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2015, pp. 18–26.

[51] Y. Xiao and K. Cho. (2016). "Efficient character-level document classification by combining convolution and recurrent layers." [Online]. Available: https://arxiv.org/abs/1602.00367

[52] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[53] P. Gamba, "A collection of data for urban area characterization," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Sep. 2004, pp. 69–72.

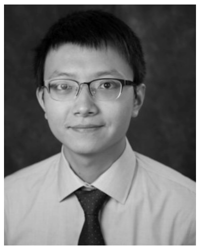
[54] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: http://tensorflow.org/

[55] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras

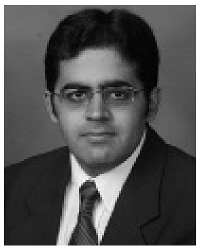
[56] F. R. Chung, *Spectral Graph Theory*. Providence, RI, USA: AMS, 1997.

[57] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[58] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.



Hao Wu (S'14) received the B.E. degree in electrical and information engineering from the University of Electronic Science and Technology of China in 2012, and the Ph.D. degree in electrical engineering from the University of Houston in 2017. His Ph.D. research was conducted under the supervision of Prof. S. Prasad in electrical and computer engineering with the Hyperspectral Image Analysis Laboratory, University of Houston. His research interests include pattern recognition and active learning for high dimensional image analysis.



Saurabh Prasad (S'05–M'09–SM'14) received the B.S. degree in electrical engineering from Jamia Millia Islamia, New Delhi, India, in 2003, the M.S. degree in electrical engineering from Old Dominion University, Norfolk, VA, USA, in 2005, and the Ph.D. degree in electrical engineering from Mississippi State University, Starkville, in 2008.

He is currently an Assistant Professor with the Electrical and Computer Engineering Department, University of Houston, Houston, TX, USA, where he leads a research group on image processing and machine learning. His current research interests include statistical pattern recognition, adaptive signal processing, compressive sensing and theoretical foundations of machine learning with applications to remote sensing and biomedical applications, design of optimal sparse representation frameworks, active learning, Bayesian inference, and kernel machines and deep learning for robust image analysis to address issues related to small-training-sample-size, mixed pixels, low SNR, and varying illumination.

Dr. Prasad was a recipient of two research excellence awards in 2007 and 2008, respectively, during his Ph.D. at Mississippi State University, including the university wide Outstanding Graduate Student Research Award. In 2008, he received the Best Student Paper Award from the IEEE International Geoscience and Remote Sensing Symposium 2008, Boston, MA, USA. In 2010, he received the State Pride Faculty Award from Mississippi State University for his academic and research contributions. In 2014, he received the NASA New Investigator (Early Career) Award, and in 2017, he received the Junior Faculty Research Award from the University of Houston. He is an active reviewer for various journals on signal processing, image processing, and machine learning. He also serves as an Associate Editor for the IEEETRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.