Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks

Yushi Chen, *Member, IEEE*, Hanlu Jiang, Chunyang Li, Xiuping Jia, *Senior Member, IEEE*, and Pedram Ghamisi, *Member, IEEE*

***Abstract*—Due to the advantages of deep learning, in this paper, a regularized deep feature extraction (FE) method is presented for hyperspectral image (HSI) classification using a convolutional neural network (CNN). The proposed approach employs several convolutional and pooling layers to extract deep features from HSIs, which are nonlinear, discriminant, and invariant. These features are useful for image classification and target detection. Furthermore, in order to address the common issue of imbalance between high dimensionality and limited availability of training samples for the classification of HSI, a few strategies such as L2 regularization and dropout are investigated to avoid overfitting in class data modeling. More importantly, we propose a 3-D CNN-based FE model with combined regularization to extract effective spectral–spatial features of hyperspectral imagery. Finally, in order to further improve the performance, a virtual sample enhanced method is proposed. The proposed approaches are carried out on three widely used hyperspectral data sets: Indian Pines, University of Pavia, and Kennedy Space Center. The obtained results reveal that the proposed models with sparse constraints provide competitive results to state-of-the-art methods. In addition, the proposed deep FE opens a new window for further research.**

***Index Terms*—Convolutional neural network (CNN), deep learning, feature extraction (FE), hyperspectral image (HSI) classification.**

I. Introduction

HYPERSPECTRAL images (HSIs) are usually composed of several hundreds of spectral data channels of the same scene. The detailed spectral information provided by hyperspectral sensors increases the power of accurately differentiating materials of interest with increased classification accuracy. Moreover, with respect to advances in hyperspectral technology, the fine spatial resolution of recently operated sensors makes the analysis of small spatial structures in images possible [1]. The aforementioned advances make the hyperspectral data a useful tool for a wide variety of applications. By increasing the dimensionality of the images in the spectral domain, theoretical and practical problems may arise. In this manner, conventional techniques which are developed for multispectral data are no longer efficient for the processing of high-dimensional data mostly due to the so-called curse of dimensionality [2]. In order to address the curse of dimensionality, feature extraction (FE) is considered as a crucial step in HSI processing [3]. However, due to the spatial variability of spectral signatures, HSI FE is still a challenging task [4].

In the early stage of the study on HSI FE, the focus was on spectral-based methods, including principal component analysis (PCA) [5], independent component analysis (ICA) [6], linear discriminant analysis [7], etc. [8], [9]. These methods apply linear transformations to extract potentially better features of the input data in the new domain. With respect to the complex light-scattering mechanisms of nature objects (e.g., vegetation), hyperspectral data are inherently nonlinear [10], [11], which make linear transformation-based methods not that suitable for the analysis of such data.

Since 2000, when two papers on manifold learning were published in *Science* [12], [13], manifold learning has become a hot topic in many research areas, including hyperspectral remote sensing. Manifold learning attempts finding the intrinsic structure of nonlinearly distributed data, which is expected to be highly useful for hyperspectral FE [14].

Alternatively, the nonlinear problem can be addressed by kernel-based algorithms for data representation [15]. Kernel methods map the original data into a higher dimensional Hilbert space and offer a possibility of converting a nonlinear problem to a linear one [16].

Recent studies have suggested incorporating spatial information into a spectral-based FE system [17]. With the development of imaging technology, hyperspectral sensors can provide good spatial resolution. As a result, detailed spatial information has become available [18]. It has been found that spectral–spatial FE methods provide good improvement in terms of classification performance [19]. In [20], a method was introduced based on the fusion of morphological operators and support vector machine (SVM), which leads to high classification

Manuscript received August 1, 2015; revised February 16, 2016; accepted June 12, 2016. Date of publication July 18, 2016; date of current version August 11, 2016. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant HIT.NSRIF.2013028 and in part by the National Natural Science Foundation of China under Grant 61301206. (*Corresponding author: Yushi Chen.*)

Y. Chen, H. Jiang, and C. Li are with the Department of Information Engineering, School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China (e-mail: chenyushi@hit.edu.cn; halo91@163.com; lcy_buzz@mail.dlut.edu.cn).

X. Jia is with the School of Engineering and Information Technology, The University of New South Wales, Canberra, A.C.T. 2600, Australia (e-mail: x.jia@adfa.edu.au).

P. Ghamisi is with Signal Processing in Earth Observation, Technische Universität München, 80333 Munich, Germany, and also with the Remote Sensing Technology Institute (IMF), German Aerospace Center (DLR), 82234 Weßling, Germany (e-mail: p.ghamisi@gmail.com).

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/TGRS.2016.2584107

accuracy. In [21], the proposed framework extracted the spatial and spectral information using loopy belief propagation and active learning. The sparse representation [22] of extended morphological attribute profile was investigated to incorporate spatial information in remote sensing image classification in [23], which further improves classification accuracy. In the hyperspectral remote sensing community, most of the current FE methods consider only one-layer processing, which downgrades the capacity of feature learning.

Most of FE and classification methods are not based on a "deep" manner. The widely used PCA and ICA are single-layer learning methods [24]. Classifiers such as linear SVMs and logistic regression (LR) can be attributed as single-layer classifiers, whereas decision tree or kernel SVMs are believed to have two layers [24].

On the other hand, it is found in neuroscience that the visual system of primate human is characterized by a sequence of different levels of processing (on the order of 10), and this kind of learning system performs very well in the tasks of object recognition [25]. Deep learning-based methods, which include two or more layers to extract new features, are designed to simulate the process from the retina to the cortex, and these deep architectures have a potential to yield high performances in image classification and target detection [26], [27].

Undesired scattering from other objects may deform the spectral characteristics of the object of interest. Furthermore, other factors such as different atmospheric scattering conditions and intraclass variability make it extremely difficult to extract the features of hyperspectral data effectively. To address such issues, deep architecture is known as a promising option since it can potentially lead to more abstract features at high levels, which are generally robust and invariant [28].

Very recently, some deep models have been proposed for hyperspectral remote sensing image processing [49]. To the best of our knowledge, a deep learning method, i.e., stacked autoencoder (SAE), was proposed for HSI classification in 2014 [29]. Later, an improved autoencoder was proposed based on sparse constraint [50]. In 2015, another deep model, entitled deep belief network (DBN), was proposed [30]. The deep models could extract the robust features and outperform other methods in terms of classification accuracy. However, due to the full connection of different layers in the aforementioned approaches, they demand to train a lot of parameters, which is an undesirable factor due to the lack of available training samples. Furthermore, SAE and DBN cannot extract the spatial information efficiently because they need to represent the spatial information into a vector before the training stage.

Convolutional neural network (CNN) uses local connections to effectively extract the spatial information and shared weights to significantly reduce the number of parameters. Very recently, an unsupervised convolutional network has been proposed for remote sensing image analysis. This method uses greedy layerwise unsupervised pretraining to formulate a deep CNN model [31].

Compared with the unsupervised method, supervised CNN may extract more effective features with the help of class-specific information, which can be provided by training samples. To extract the spectral and spatial information of hyperspectral data simultaneously, it is reasonable to formulate a 3-D CNN. Furthermore, to address the problem of overfitting caused by limited training samples of hyperspectral data, we design a combined regularization strategy, including rectified linear unit (ReLU) and dropout to achieve better model generalization.

In this paper, we investigate the application of supervised CNN, which is one of the deep models, in HSI FE and develop a 3-D CNN model for effective spectral-and-spatial-based HSI classification. It is challenging to apply deep learning to HSI since its data structure is complex and the number of training samples is limited. In computer vision, the number of training samples varies from tens of thousands to tens of millions [32], [33], whereas having such a large number of training samples is not common in hyperspectral remote sensing classification. In general, a neural network has a powerful representation capability with abundant training samples. Without enough training samples, a neural network faces a problem of "overfitting," which means that the classification performance of test data will be downgraded. This problem is expected when deep learning is applied to remote sensing data while this paper presents a solution to make such approaches feasible for situations when only a limited number of training samples is available. We use several regularization methods, including L2 regularization, and dropout strategies to handle the overfitting issue.

The main goal of this paper is to propose a deep FE method for HSI classification. With the help of training samples, the proposed CNN models extract the abstract and robust features of HSI, which are important for classification. In more detail, the main contributions are listed as follows.

1) Three deep FE architectures based on a CNN are proposed to extract the spectral, spatial, and spectral–spatial features of HSI. The designed 3-D CNN can extract the spectral–spatial features effectively, which leads to better classification performance.
2) To address the problem of overfitting caused by the limited number of training samples, some regularization strategies, including L2 regularization and dropout, are used in the training process.
3) In order to further improve the performance, a virtual sample enhanced method is proposed to create training samples from the imaging procedure perspective.
4) The hierarchical features of different depth extracted from HSI are visualized and analyzed for the first time.
5) The proposed methods are applied on three well-known hyperspectral data sets. In this context, we compared the proposed methods with some traditional methods from a different perspective such as classification accuracy, analysis of complexities, and processing time.

The remainder of this paper is organized as follows: Section II presents the description of CNN and 1-D CNN-based HSI spectral FE frameworks. Sections III and IV present the spatial and spectral–spatial FE frameworks for HSI classification, respectively. The virtual sample enhanced CNN is introduced in Section V. The experiments conducted are reported in Section VI. We conclude this paper in Section VII with some discussions.

II. One-Dimensional CNN-Based HSI FE and Classification

A. Neural Network and Deep Learning

How to find effective features is the core issue in image classification and pattern recognition. Humans have an amazing skill in extracting meaningful features, and a lot of research projects have been undertaken to build an FE system as smart as human in the last several decades. Deep learning is a newly developed approach aiming for artificial intelligence.

Deep learning-based methods build a network with several layers, typically deeper than three layers. Deep neural network (DNN) can represent complicated data. However, it is very difficult to train the network. Due to the lack of a proper training algorithm, it was difficult to harness this powerful model until Hinton and Salakhutdinov proposed a deep learning idea [27].

Deep learning involves a class of models that try to learn multiple levels of data representation, which helps to take advantage of input data such as image, speech, and text. Deep learning model is usually initialized via unsupervised learning and followed by fine-tuning in a supervised manner. The high-level features can be learnt from the low-level features. This kind of learning leads to the extraction of abstract and invariant features, which is beneficial for a wide variety of tasks such as classification and target detection.

There are a few deep learning models in the literature, including DBN [34], [35], SAE [36], and CNN [28]. Recently, CNNs have been found to be a good alternative to other deep learning models in classification [38], [39] and detection [40]. In this paper, we investigate the application of deep CNN for HSI FE.

B. CNN (1-D CNN)

The human visual system can tackle classification, detection, and recognition issues very effectively. Therefore, machine learning researchers have developed advanced data processing methods in recent years based on the inspirations from biological visual systems [25].

CNN is a special type of DNN that is inspired by neuroscience. From Hubel's earlier work, we know that the cells in the cortex of the human vision system are sensitive to small regions. The responses of cells within receptive fields have a strong capability to exploit the local spatial correlation in images.

Additionally, there are two types of cells within the visual cortex, i.e., simple cells and complex cells. While simple cells detect local features, complex cells "pool" the outputs of simple cells within a neighborhood. In other words, simple cells are sensitive to specific edge-like patterns within their receptive field, whereas complex cells have large receptive fields and they are locally invariant.

The architecture of CNN is different from other deep learning models. There are two special aspects in the architecture of CNN, i.e., local connections and shared weights. CNN exploits the local correlation using local connectivity between the neurons of near layers. We illustrate this in Fig. 1, where the neurons in the mth layer are connected to three adjacent

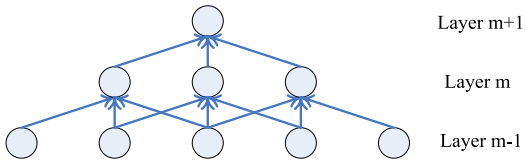


Fig. 1. Local connections in the architecture of the CNN.

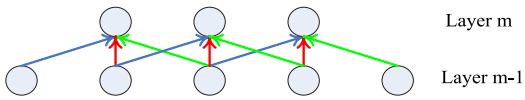


Fig. 2. Shared weights in the architecture of the CNN.

neurons in the $(m-1)$th layer, as an example. In CNN, some connections between neurons are replicated across the entire layer, which share the same weights and biases. In Fig. 2, the same color indicates the same weight. Using a specific architecture like local connections and shared weights, CNN tends to provide better generalization when facing computer vision problems.

A complete CNN stage contains a convolution layer and a pooling layer. Deep CNN is constructed by stacking several convolution layers and pooling layers to form deep architecture.

The convolutional layer is introduced first. The value of a neuron v_{ij}^{x} at position x of the jth feature map in the ith layer is denoted as follows:

$$v_{ij}^{x} = g\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1} w_{ijm}^{p} v_{(i-1)m}^{x+p}\right) \tag{1}$$

$$g(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

where m indexes the feature map in the previous layer ($(i-1)$th layer) connected to the current feature map, w_{ijm}^{p} is the weight of position p connected to the mth feature map, P_i is the width of the kernel toward the spectral dimension, and b_{ij} is the bias of jth feature map in the ith layer.

Pooling can offer invariance by reducing the resolution of the feature maps [37]. Each pooling layer corresponds to the previous convolutional layer. The neuron in the pooling layer combines a small $N \times 1$ patch of the convolution layer. The most common pooling operation is max pooling, which is used throughout this paper. The max pooling is as follows:

$$a_j = \max_{N\times 1}\left(a_i^{n\times 1} u(n,1)\right) \tag{3}$$

where $u(n,1)$ is a window function to the patch of the convolution layer, and a_j is the maximum in the neighborhood.

All layers, including the convolutional layers and pooling layers of the deep CNN model, are trained using a back-propagation algorithm.

C. Spectral FE Framework for HSI Classification

In this section, we present a 1-D FE method considering only spectral information. This method stacks several CNNs to develop a deep CNN model with L2 regularization. Generally, the classification of HSI includes two procedures, including FE

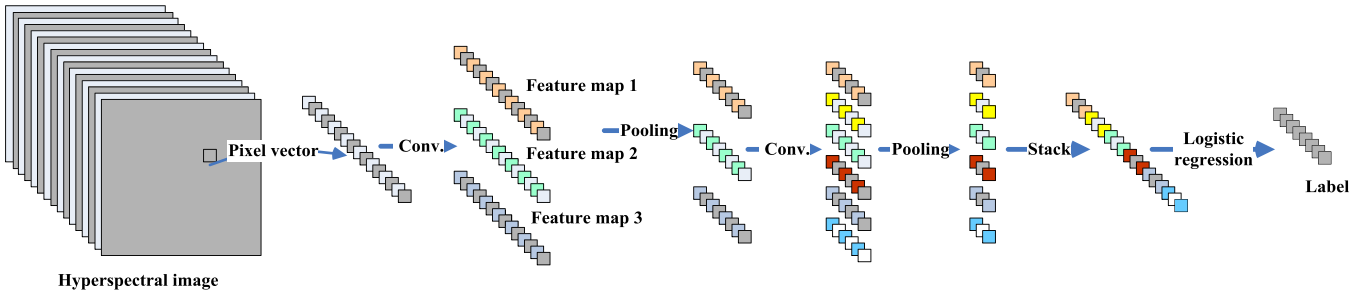


Fig. 3. Architecture of deep CNN with spectral FE of HSI.

and classification. In the FE procedure, LR is taken into account to adjust the weights and biases in the back-propagation. After the training, the learned features can be used in conjunction with classifiers such as LR, K-nearest neighbor (KNN), and SVMs [1].

The proposed architecture is shown in Fig. 3. The input of the system is a pixel vector of hyperspectral data, and the output of the system is the label of the pixel vector. It consists of several convolutional and pooling layers and an LR layer. In Fig. 3, as an example, the flexible CNN model includes two convolution layers and two pooling layers. There are three feature maps in the first convolution layer and six feature maps in the second convolution layer.

After several layers of convolution and pooling, the input pixel vector can be converted into a feature vector, which captures the spectral information in the input pixel vector. Finally, we use LR or other classifiers to fulfill the classification step.

The power of CNN depends on the connections (weights) of the network; hence, it is very important to find a set of proper weights. Gradient back-propagation is the core fundamental algorithm for all kinds of neural networks. In this paper, the model parameters are initialized randomly and trained by an error back-propagation algorithm.

Before setting an updating rule for the weights, one needs to properly set an "error" measure, i.e., a cost function. There are several ways to define such a cost function. In our implementation, a mini-batch update strategy is adopted, which is suitable for large data set processing, and the cost is computed on a mini-batch of inputs [37]

$$c_0 = -\frac{1}{m}\sum_{i=1}^{m}\left[x_i \log(z_i) + (1-x_i)\log(1-z_i)\right]. \quad (4)$$

Here, m denotes the mini-batch size. Two variables x_i and z_i denote the ith predicted label and the label in the mini-batch, respectively. The i summation is done over the whole mini-batch. Our hope turns to optimize (4) using mini-batch stochastic gradient descent.

LR is a type of probabilistic statistical classification model. It measures the relation between a categorical variable and the input variables using probability scores as the predicted values of the input variables.

To perform classification by utilizing the learned features from the CNN, we employ an LR classifier, which uses softmax as its output-layer activation. Softmax ensures that the activation of each output unit sums to 1 so that we can deem the output as a set of conditional probabilities. For given input vector R, the probability that the input belongs to category i can be estimated as follows:

$$P(Y=i|R,W,b) = s(WR+b) = \frac{e^{W_i R+b_i}}{\sum_j e^{W_j R+b_j}} \quad (5)$$

where W and b are the weights and biases of the LR layer, and the summation is done over all the output units.

In the LR, the size of the output layer is set to be the same as the total number of classes defined, and the size of the input layer is set to be the same as the size of the output layer of the CNN. Since the LR is implemented as a single-layer neural network, it can be merged with the former layers of networks to form a deep classifier.

D. L2 Regularization of CNN

Overfitting is a common problem of neural network approaches, which means that the classification results can be very good on the training data set but poor on the test data set. In this case, HSI will be classified with low accuracy. The number of training samples is limited in HSI classification, which often leads to the problem of overfitting.

To avoid overfitting, it is necessary to adopt additional techniques such as regularization. In this section, we introduce L2 regularization in the proposed model, which is a penalizing model with extreme parameter values [41].

L2 regularization encourages the sum of the squares of the parameters to be small, which can be added to learning algorithms that minimize a cost function. Equation (4) is then modified to

$$c = c_0 + \frac{\lambda}{2m}\sum_{j=1}^{N} w_j^2 \quad (6)$$

where m denotes the mini-batch size, N is the number of weights, and λ is a free parameter that needs to be tuned empirically. In addition, the coefficient, 1/2, is used to simplify the process of the derivation.

In (6), one can see that L2 regularization can make w small. In most cases, it can help with the reduction of the bias of the model to mitigate the overfitting problem.

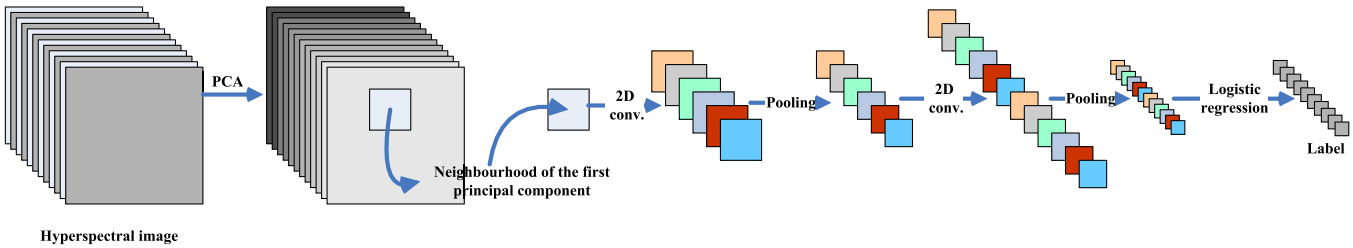


Fig. 4. Architecture of CNN with spatial features for HSI classification. The first step of processing is PCA along with spectral dimension, and then CNN is introduced to extract layerwise deep features.

III. Two-Dimensional CNN-Based HSI FE and Classification

A. Two-Dimensional CNN

A complete 2-D CNN layer contains a convolutional layer and a pooling layer. The 2-D convolutional layer is obtained by the extension of (1). The value of a neuron v_{ij}^{xy} at position (x, y) of the jth feature map in the ith layer is denoted as follows:

$$v_{ij}^{xy} = g\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}\right) \quad (7)$$

where m indexes the feature map in the $(i-1)$th layer connected to the current (jth) feature map, w_{ijm}^{pq} is the weight of position (p, q) connected to the mth feature map, P_i and Q_i are the height and the width of the spatial convolution kernel, and b_{ij} is the bias of the jth feature map in the ith layer.

Pooling is carried out in the similar way to the 1-D CNN. The neuron in the pooling layer combines a small $n \times n$ patch of the convolutional layer.

B. Fine-Tuning and Classification

Based on the theory described previously, a variety of CNN architectures can be developed. In this section, we present the designed CNN for a single band (the first principal component) of HSI. The architecture is shown in Fig. 4.

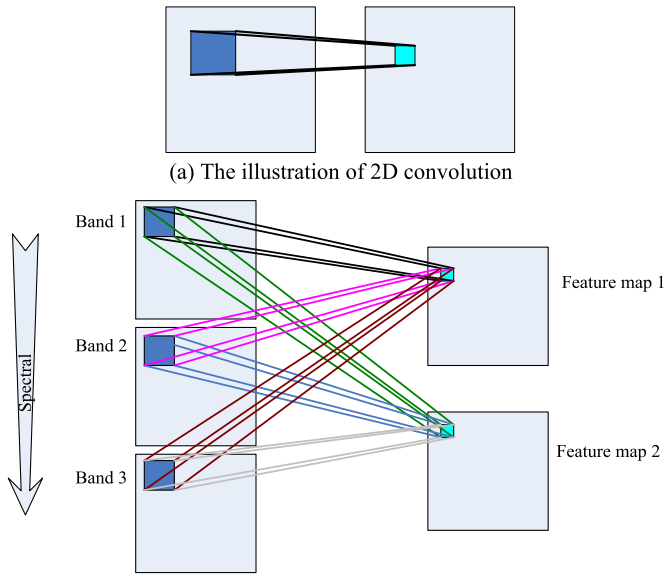
We choose $K \times K$ neighborhoods of a current pixel as the input to the 2-D CNN model. Then, we build deep CNN to extract the useful features. Each layer of CNN contains 2-D convolution and pooling. When the spatial resolution of the image is not very high, 4×4 kernel or 5×5 kernel can be selected to run convolution and 2×2 kernel for pooling.

After several layers of convolution and pooling, the input image can be represented by some feature vectors, which capture the spatial information contained in the $K \times K$ neighborhood region of the input pixel. Then, the learned features are fed to the LR for classification.

IV. Three-Dimensional CNN-Based HSI FE and Classification

A. Three-Dimensional CNN

We can see from Sections II and III that the 1-D CNN extracts spectral features and the 2-D CNN extracts the local spatial features of each pixel. We further develop 3-D CNN to learn both spatial and spectral features of HSI. Fig. 5 shows the comparison of 2-D and 3-D convolutions.



(a) The illustration of 2D convolution

(b) The illustration of 3D convolution

Fig. 5. Comparison of (a) 2-D and (b) 3-D convolutions. In (b), the size of the convolution kernel in the spectral dimension is 3 and the weights are color-coded.

The value of a neuron v_{ij}^{xyz} at position (x, y, z) of the jth feature map in the ith layer is given by

$$v_{ij}^{xyz} = g\left(\sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)} + b_{ij}\right) \quad (8)$$

where m indexes the feature map in the $(i-1)$th layer connected to the current (jth) feature map, and P_i and Q_i are the height and the width of the spatial convolution kernel. R_i is the size of the kernel along toward spectral dimension, w_{ijm}^{pqr} is the value of position (p, q, r) connected to the mth feature map, and b_{ij} is the bias of the jth feature map in the ith layer.

Through 3-D convolution, CNN can extract the spatial and spectral information of hyperspectral data simultaneously. The learned features are useful for the further image classification step.

B. Spectral–Spatial FE Framework

Hyperspectral remote sensing images contain both spatial and spectral information. In this section, we integrate the spectral and spatial features together to construct a joint spectral–spatial classification framework using a 3-D CNN.

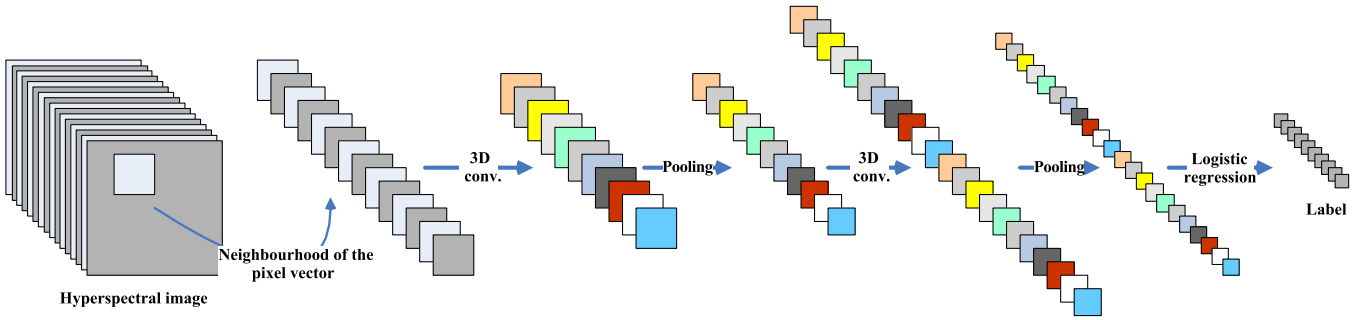


Fig. 6. Architecture of 3-D CNN with spectral–spatial features for HSI classification.

Fig. 6 shows the architecture of 3-D CNN for HSI classification. We choose $K \times K \times B$ neighborhoods of a pixel as an input to the 3-D CNN model, in which B is the number of bands. Each layer of CNN contains 3-D convolution and pooling. As an example, a $4 \times 4 \times 32$ kernel or a $5 \times 5 \times 32$ kernel can be applied to 3-D convolution, and a 2×2 kernel can be applied for subsampling. After performing a deep 3-D CNN, the LR approach is conducted for the classification step.

C. Regularizations Based on Sparse Constraints

The issue of high dimensionality and limited number of training samples makes the overfitting a serious problem, particularly when the input is a 3-D cube. The dimensionality of the spectral-based CNN, which is presented in Section II-C, is around a couple of hundreds (the number of bands); the dimensionality of the spatial-based CNN, which is presented in Section III-B, is around several hundreds ($K \times K$, e.g., $K = 27$); the dimensionality of the spectral-and-spatial-based CNN, which is presented in Section IV-B, is around several thousands ($K \times K \times B$). It is easy to obtain that the high dimensionality of the input data may lead to an overfitting situation. In order to handle the issue of 3-D CNN, a combined regularization strategy based on sparse constraint is developed, which includes ReLU and dropout, and applies dropout in the fully connected layer.

There are different kinds of ReLUs available to apply. In this paper, the adopted ReLU is a simple nonlinear operation that accepts the input of a neuron if it is positive, whereas it returns to 0 if the input is negative. In many applications, ReLUs in CNNs can improve the performances [42].

Dropout is a recently introduced method to handle overfitting. It sets the output of some hidden neurons to zero, which means that the dropped neurons do not contribute in the forward pass and they are not used in the back-propagation procedure. In different training epochs, the deep CNN forms a different neural network by dropping neurons randomly. The dropout method prevents complex co-adaptations [43].

By using ReLU and dropout, the outputs of many neurons are 0. We use several ReLUs and dropouts at several layers to achieve powerful sparse-based regularization for the deep network and address the overfitting problem in HSI classification.

V. Virtual Sample Enhanced CNN

As a matter of fact, CNN has a lot of weights needed to be trained. Inappropriate weights may cause getting trapped in a local minimal of the loss function, which results in poor performance. To obtain proper weights, a lot of samples are required in the training procedure. However, these samples are usually obtained by manual labeling of a small number of pixels in an image or based on some field measurements. Therefore, the collection of these samples is both expensive and time demanding. Consequently, the number of available training samples is usually limited, which is a challenging issue in supervised classification. To solve the dilemma, we utilize virtual sample as a promising tool from a different perspective.

The virtual sample method tries to create new training samples from given training samples. The critical issue is how to generate proper samples while we figure out a solution from the imaging procedure perspective. Because of the complex situation of lighting in the large scene, objects of the same class show different characteristics in different locations. Therefore, we can simulate a virtual sample by multiplying a random factor to a training sample and adding random noise. Furthermore, we can generate a virtual sample from two given samples of the same class with proper ratios. The virtual sample idea is helpful in the training of a CNN.

To tackle the problem of having limited training samples, instead of regularization such as L2 regularization and dropout, virtual samples have been generated and added to the training samples.

A. Changing Radiation-Based Virtual Samples

Remote sensing, including hyperspectral imaging, usually contains a large scene, whereas the objects of the same class in different locations are affected by different radiation. Virtual samples can be created by simulating the imaging procedure. New virtual sample y_n is obtained by multiplying a random factor and adding random noise to a training sample $\boldsymbol{x_m}$

$$\boldsymbol{y_n} = \alpha_m \boldsymbol{x_m} + \beta \boldsymbol{n}. \tag{9}$$

The training sample $\boldsymbol{x_m}$ is a cube extracted from the hyperspectral cube, which contains the spectral and spatial information of pixel to be classified.

In (9), α_m indicates the disturbance of light intensity, which can vary under many situations such as seasons and atmospheric conditions, whereas β controls the weight of the random Gaussian noise $\boldsymbol{n}$, which may result from the interaction of adjacent pixels and instrumental error.

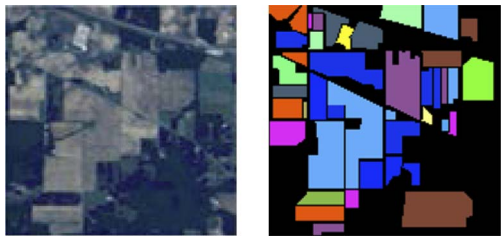


Fig. 7. Indian Pines data set. (Left) False color composite image (bands 28, 19, and 10) and (right) ground truth.

B. Mixture-Based Virtual Samples

Because of the long distance between the object and the sensor, mixture is very common in remote sensing. Inspired by the phenomenon, it is possible to generate a virtual sample $\boldsymbol{y_k}$ from two given samples of the same class with proper ratios

$$\boldsymbol{y_k} = \frac{\alpha_i \boldsymbol{x_i} + \alpha_j \boldsymbol{x_j}}{\alpha_i + \alpha_j} + \beta \boldsymbol{n}. \tag{10}$$

In (10), $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ are two training samples from the same class, and $\boldsymbol{y_k}$ is the virtual sample generated by the two training samples, whereas β controls the weight of the random Gaussian noise $\boldsymbol{n}$. Based on the fact that the hyperspectral characteristics of one class vary within a certain range, it is reasonable to assume that results of mixture within this range still belong to the same class. Therefore, here, we assign the same label of training samples to the virtual sample $\boldsymbol{y_k}$.

Then, the real training samples and virtual samples are used together as training samples to get the proper weights in the network.

Although there are many other methods that can generate the virtual samples, the changing radiation and mixture-based methods are simple yet effective ways.

VI. Experimental Results

A. Data Description and Experiment Design

In our study, three widely used hyperspectral data sets with different environmental settings were adopted to validate the proposed methods. They are a mixed vegetation site over the Indian Pines test area in Northwestern Indiana (Indian Pines), an urban site over the city of Pavia, Italy (University of Pavia), and a site over Kennedy Space Center (KSC), Florida.

The first data set was acquired by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). The data set was obtained from an aircraft flown, with a size of 145 pixels $\times$ 145 pixels and 220 spectral bands in the wavelength range of 0.4–2.5 μm. The false color image is shown in Fig. 7(a). The number of bands is reduced to 200 by removing water absorption bands. Sixteen different land-cover classes are provided in the ground truth, as shown in Fig. 7. The number of samples of each class is listed in Table I.

The second data set was gathered by a sensor known as the Reflective Optics System Imaging Spectrometer (ROSIS-3) over the city of Pavia, Italy, with 610 pixels $\times$ 340 pixels and 115 bands in the range of 0.43–0.86 μm. The high spatial resolution of 1.3 m/pixel aims to avoid a high percentage of mixed

TABLE I
Land-Cover Classes and Numbers of Pixels on the Indian Pines Data Set

Class			Samples	
No.	Color	Name	Train	Test
1		Alfalfa	30	16
2		Corn-notill	150	1198
3		Corn-min	150	232
4		Corn	100	5
5		Grass-pasture	150	139
6		Grass-trees	150	580
7		Grass-pasture-mowed	20	8
8		Hay-windrowed	150	130
9		Oats	15	5
10		Soybean-notill	150	675
11		Soybean-mintill	150	2032
12		Soybean-clean	150	263
13		Wheat	150	55
14		Woods	150	793
15		Buildings-Grass-Trees	50	49
16		Stone-Steel-Towers	50	43
Total			1765	6223

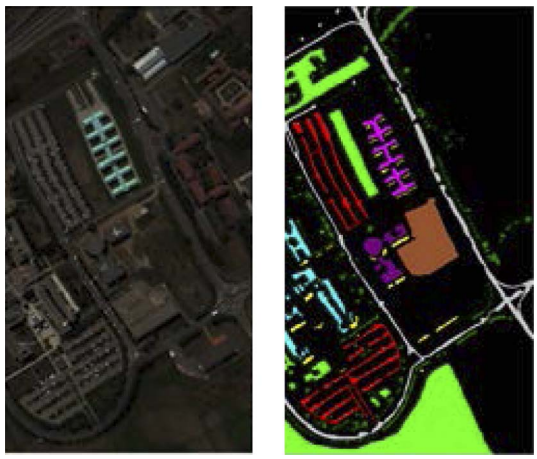


Fig. 8. University of Pavia data set. (Left) False color composite (bands 10, 27, and 46) and (right) representing nine land-cover classes.

pixels. In the experiment, noisy bands have been removed and the remaining 103 channels were used for classification. Nine land-cover classes were selected, which are shown in Fig. 8, and the numbers of samples for each class are given in Table II.

The third data set was acquired by the AVIRIS instrument over KSC, Florida, on March 23, 1996. The KSC data set has an altitude of approximately 20 km, with a spatial resolution of 18 m. The data set includes 176 bands used for the analysis after removing water absorption and low-signal-to-noise-ratio bands. For classification purposes, 13 classes were defined for the site. The samples are listed in Table III and shown in Fig. 9.

For all three data sets, we split the labeled samples into two subsets, i.e., training and test samples, and the details are listed in Tables I–III. During the training procedure of CNN, we used 90% of the training samples to learn weights and biases of each neuron and the remaining 10% of the training samples to guide the design of proper architectures. In other words, we use

TABLE II
LAND-COVER CLASSES AND NUMBERS OF PIXELS ON THE UNIVERSITY OF PAVIA DATA SET

Class			Samples	
No.	Color	Name	Train	Test
1		Asphalt	548	5472
2		Meadows	540	13750
3		Gravel	392	1331
4		Trees	542	2573
5		Metal Sheets	256	1122
6		Bare Soil	532	4572
7		Bitumen	375	981
8		Bricks	514	3363
9		Shadow	231	776
Total			3930	33940

TABLE III
LAND-COVER CLASSES AND NUMBERS OF PIXELS IN THE KSC DATA SET

Class			Samples	
No.	Color	Name	Train	Test
1		Scrub	33	314
2		Willow swamp	23	220
3		CP hammock	24	232
4		Slash pine	24	228
5		Oak/Broadleaf	15	146
6		Hardwood	22	207
7		Swamp	9	96
8		Graminoid marsh	38	352
9		Spartina marsh	51	469
10		Cattail marsh	39	365
11		Salt marsh	41	378
12		Mud flats	49	454
13		Water	91	836
Total			459	4297

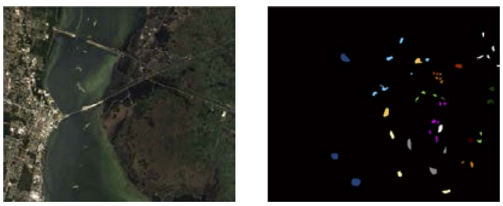


Fig. 9. KSC data set. (Left) False color composite (bands 28, 19, and 10) and (right) ground truth.

the classification results of the remaining 10% of the training samples to identify if the network is overfitted. This is important for designing the network. The test set is used to assess final classification performance.

The experiments were conducted in four scenarios. The first scenario aims at extracting the deep spectral features of HSI. The second scenario tests the usefulness of deep spatial FE. After this, the effectiveness of deep spectral–spatial FE is investigated. In the last scenario, the results of virtual sample methods are presented.

In order to quantitatively compare the capabilities of the proposed models, overall accuracy (OA), average accuracy (AA), and Kappa coefficient K are used as performance measures. We run the experiments 20 times with different initial random training samples, and then confidence intervals of OA, AA, and K are reported.

TABLE IV
ARCHITECTURES OF THE 1-D CNN ON THREE DATA SETS

Layer Name		*I1*	*C2* *S3*	*C4* *S5*	*C6* *S7*	*C8* *S9*	*C10* *S11*	*F12*	*O13*
Kernel Size	Indian Pines	1×200	1×5 1×2	1×5 1×2	1×4 1×2	1×5 1×2	1×4 1×1	Fully connected	1×16
	Pavia University	1×103	1×8 1×2	1×7 1×2	1×8 1×2	-	-	Fully connected	1×9
	KSC	1×176	1×9 1×2	1×9 1×2	1×9 1×2	1×10 1×2	-	Fully connected	1×13
Feature Map		1	6	12	24	48	96	256	1

B. Design CNN With Spectral Features

Spectral feature-based HSI classification is a traditional and widely used method, in which the pixel vector of HSI is the input. The primary objective of this section is to design a CNN model to evaluate the effectiveness of deep FE in the spectral domain. The experiments include the design and visualization of spectral information-based CNN and the comparisons with other FE methods and typical classifiers.

1) Architecture Design of the 1-D CNN: Optimization of CNN was performed using the trial-and-error approach again to determine the parameters of model on the number of nodes in hidden layers, learning rate, kernel size, and the number of convolution layers.

Table IV shows the architectures of deep CNNs for three data sets. As an example, for the Indian Pines data set, there are 13 layers, denoted as $I1$, $C2$, $S3$, $C4$, $S5$, $C6$, $S7$, $C8$, $S9$, $C10$, $S11$, $F12$, and $O13$ in sequence. $I1$ is the input layer. C refers to the convolution layers, and S refers to the pooling layers. $F12$ is a fully connected layer, and $O13$ is the output layer of the whole neural network.

The input data are normalized into $[-1\ \ 1]$. For the LR, the learning rate is set to 0.005, and the training epoch is 700 for the Indian Pines data set. For the University of Pavia data set, we set the learning rate to 0.01 and the number of epochs to 300. For the KSC data set, the learning rate is 0.001 with 600 epochs. A generalized cross-validation method is applied to estimate the normalization parameter of L2 regularization [44].

Fig. 10 shows the classification results of the Indian Pines, University of Pavia and KSC data sets. In Fig. 10, we can see that the depth does help improve classification accuracy. However, too much layers may downgrade classification results. The numbers of proper convolution layers are 5, 3, and 4 for the Indian Pines, University of Pavia, and KSC data sets, respectively. This is affected by the dimensionalities of inputs, which are 200, 103, and 176, respectively.

2) Visualization and Analysis of the 1-D CNN: In order to gain detailed understanding of the 1-D CNN, visualization of CNN for HSI is provided in this section. In the visualization and analysis part, the University of Pavia data set is used as an example.

Weights play a key role in a neural network; hence, they are displayed in grayscale images for visualization. Every row in the figures represents a convolutional kernel, and the intensities

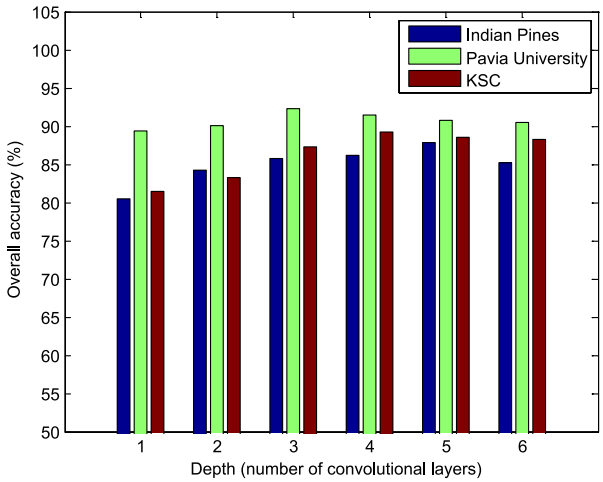
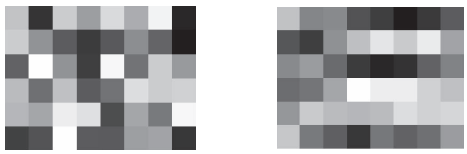


Fig. 10. Spectral information-based classification results of different depths using CNN.



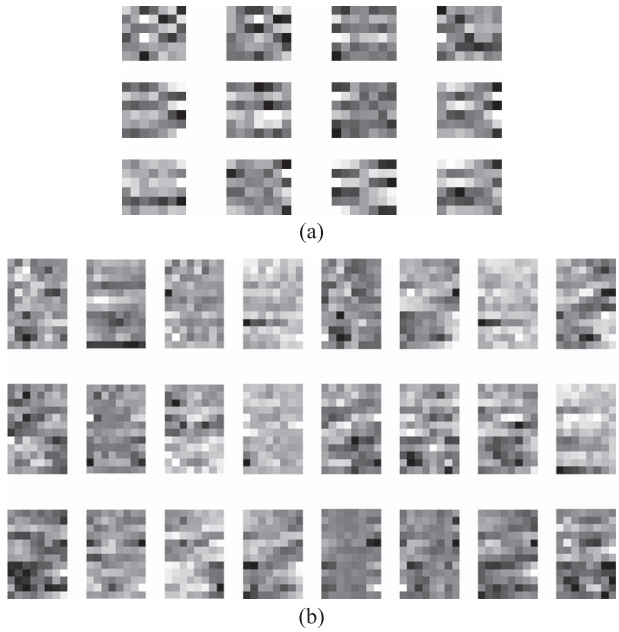
(a) (b)

Fig. 11. Weights of the first convolutional layer on the University of Pavia data set. Each tiny image (1×8) stands for the weights of a convolutional kernel. There are six convolutional kernels in the first convolutional layer. The intensity of each pixel stands for the value of corresponding weight. (a) Randomly initialized weights of first convolution layer. (b) Learned weights of first convolution layer.

in a row represent the connection intensity of the network. Each convolutional kernel can extract the unique feature of the input. Fig. 11 shows the weights of the first convolutional layer on the University of Pavia data set. The weights are randomly initialized and trained using back-propagation methods. From Fig. 11(b), the learned weights show some structures. For example, the intensities of the first row are high on the left side and low on the right side. Fig. 14 shows the weights of the 2-D CNN, and it is helpful for the understanding. Different convolutional kernels can extract the features from different perspectives, and the abundant features are helpful for further processing.

Fig. 12 shows the weights learned at the second and third convolutional layers in an image form where the brightness is proportional to the value of the weights. There are 12 and 24 convolutional kernels at layers 2 and 3, respectively. The numbers of weights, i.e., 42 at layer 2 and 96 at layer 3, are arranged in an image form artificially. Different convolutional kernels can extract the features from different perspectives. The abundant features are helpful for further processing.

The learned features, which are obtained by the convolution of inputs and kernels, on the University of Pavia data set are illustrated as curves in Fig. 13. The class of Meadows is selected for visualization, and the extracted features after each convolutional layer are shown with a different color. It is shown that these different features are extracted by different convolution kernels. The extracted features become more abstract after the third convolutional and pooling layers.



(a) (b)

Fig. 12. Weights of the second and third convolutional layers on the University of Pavia data set. In the first column of the image, there are 12 filters, and each tiny image contains $42(6 \times 7)$ weights of a convolutional kernel. The second one shows 24 filters and $96(12 \times 8)$ weights in a tiny image. (a) Learned weights of the second convolutional layer. (b) Learned weights of the third convolutional layer.

In order to evaluate the effectiveness of the extracted features, the similarity in the same class and the divisibility between different classes are shown in Table V in a quantitative way. We selected three classes for calculation and calculated the divisibility of different classes with $J-M$ distance. The $J-M$ distance is defined as [45]

$$J_{ij} = \sqrt{2(1-e^{-B_{ij}})} \tag{11}$$

$$B_{ij} = \frac{1}{8}(m_i - m_j)^T \left(\frac{c_i + c_j}{2}\right)^{-1} (m_i - m_j) + \frac{1}{2}\log\left(\frac{\left|\frac{(c_i+c_j)}{2}\right|}{\sqrt{|c_i||c_j|}}\right) \tag{12}$$

where m_i and c_i are sample's average vector and covariance matrix. B_{ij} is the Bhattacharyya distance between the two classes.

The similarity in the same class is evaluated with the correlation coefficient on a scale of -1 to 1. The correlation coefficient calculation formula is defined as follows:

$$\rho_{x,y} = \frac{C(x,y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{13}$$

where x and y are two feature vectors, whereas $C(x,y)$ is a covariance matrix. $D(x)$ and $D(y)$ are the variances of two vectors. We use the mean of all correlation coefficients to evaluate the similarity in the same class.

The higher similarity within class and the higher divisibility between classes make the classification step smoother. From Table V, by comparing the calculated results in different layers, one can see that features have a high similarity in the same class and large divisibility in different classes as the number of

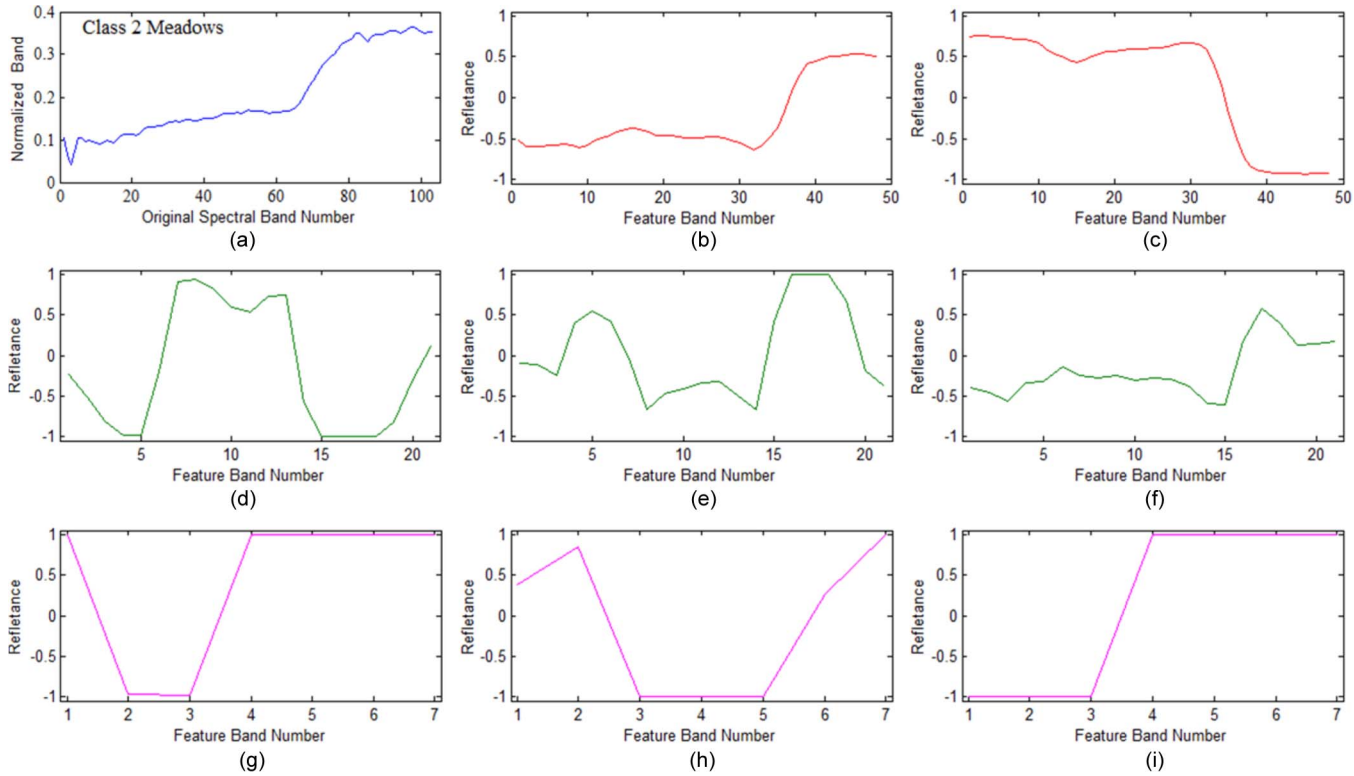


Fig. 13. Extracted features after convolution and pooling layers on the University of Pavia data set. (a) Original spectral information. (b) and (c) Features after the first convolutional layer. (d)–(f) Features after the second convolutional layer. (g)–(i) Features after the third convolutional layer.

TABLE V
SIMILARITY AND DIVISIBILITY OF SPECTRAL FEATURES ON THE UNIVERSITY OF PAVIA DATA SET

Quantitative Methods	Mean of Correlation Coefficients (similarity)			J-M distance (divisibility)		
	Class1	Class2	Class5	Class (1,2)	Class (1,5)	Class (2,5)
Specture input	0.5766	0.6849	0.7415	0.3153	0.6373	0.5863
Features after layer1	0.7640	0.7830	0.8402	0.5842	0.8534	0.7562
Features after layer2	0.8410	0.8660	0.9074	0.8342	0.9663	0.9423
Features after layer3	0.8706	0.9147	0.9155	0.9442	1.1335	1.0532

convolutional layers increases. Therefore, the results infer that the extracted features are valid and efficient.

3) Comparisons With Different FE Methods and Classifiers: In this set of experiments, CNN was compared with the PCA, factor analysis (FA), and locally linear embedding (LLE) in order to investigate the potential of CNN for hyperspectral spectral FE. PCA is a widely used FE method. FA is a linear statistical method designed for potential factors from observed variables to replace original data [46]. LLE is a popular nonlinear dimension reduction method, which is considered as a kind of manifold learning algorithm [47]. In this paper, the effectiveness of different FE methods is evaluated mainly through classification results. We also classify the features using several classifiers such as KNN classifier and a nonlinear SVM based on radial basis function (RBF-SVM). Using the same features with different classifiers, we can evaluate the effectiveness of the extracted features.

Tables VI–VIII show that the CNN-based FE methods always provide the best performances of OA, AA, and Kappa for all three data sets. The classification accuracy values are given in the form of mean $\pm$ standard deviation from the perspective of statistics, which is used as a measurement of volatility.

In order to have a fair comparison, we used 10% of the training samples to find the best parameters of FE methods using grid search. The result reported in Tables VI–VIII are the best classification results when the number of features was properly selected for each FE method. On the selection of parameters, the number of features was chosen in the range of 10 to N (i.e., the number of hyperspectral bands) with an interval of 10. The number of neighbors in LLE has been changed in a range from 1 to 10. The final classification results such as OA, AA, and Kappa were calculated on the test data set. In this set of experiments, CNN was compared with the PCA, FA, and LLE in order to investigate the potential of CNN for hyperspectral spectral FE. PCA is a widely used FE method. FA is a linear statistical method designed for potential factors from observed variables to replace original data [46]. LLE is a popular nonlinear dimension reduction method, which is considered as a kind of manifold learning algorithm [47]. In this paper, the effectiveness of different FE methods is evaluated mainly through classification results. We also classify the features using several classifiers such as KNN classifier and an RBF-SVM. Using the same features with different classifiers, we can evaluate the effectiveness of the extracted features.

TABLE VI
CLASSIFICATION RESULTS OBTAINED BY DIFFERENT FE APPROACHES ON THE INDIAN PINES DATA SET

Classifier	KNN					LR					RBF-SVM				
FE method	\	PCA	FA	LLE	CNN	\	PCA	FA	LLE	CNN	\	PCA	FA	LLE	CNN
OA(%)	66.87 ±0.53	66.65 ±0.42	82.02 ±0.49	79.96 ±0.45	84.30 ±0.26	64.93 ±1.47	65.44 ±1.17	81.31 ±0.49	80.67 ±1.20	**87.81 ±0.32**	83.89 ±2.07	83.20 ±1.17	84.76 ±0.50	83.13 ±1.22	86.62 ±0.66
AA(%)	73.37 ±2.86	73.30 ±2.80	87.77 ±1.45	84.41 ±1.78	89.52 ±0.76	59.17 ±1.24	60.57 ±1.21	87.91 ±1.15	85.23 ±1.73	**93.12 ±0.71**	89.24 ±1.92	88.15 ±2.35	89.66 ±0.95	87.80 ±1.24	91.17 ±0.83
K×100	61.24 ±0.54	61.00 ±0.42	78.55 ±0.56	76.17 ±0.51	79.69 ±0.28	57.97 ±1.57	58.79 ±1.37	77.69 ±0.57	76.96 ±1.32	**85.30 ±0.36**	80.7 ±2.37	79.86 ±1.35	82.25 ±0.58	79.46 ±1.33	85.07 ±0.77
Alfalfa	68.75 ±12.3	75.63 ±9.97	80.63 ±4.61	75.63 ±12.3	87.00 ±3.06	64.25 ±12.57	71.48 ±11.5	83.13 ±5.93	76.88 ±17.7	89.58 ±5.89	88.13 ±6.88	88.13 ±4.61	90.00 ±5.27	88.46 ±7.31	**91.25 ±3.42**
Corn-notill	54.17 ±3.77	54.70 ±4.42	82.95 ±1.89	79.97 ±5.01	83.59 ±0.69	33.36 ±3.86	36.48 ±3.35	81.79 ±0.73	77.41 ±5.92	85.68 ±0.92	79.62 ±4.49	78.58 ±2.92	85.75 ±0.98	76.53 ±5.52	**86.16 ±1.16**
Corn-mintill	56.29 ±3.30	55.95 ±3.43	79.48 ±2.03	74.31 ±6.77	78.47 ±1.58	35.00 ±6.88	38.23 ±3.38	75.69 ±2.69	71.34 ±7.94	**87.36 ±1.73**	82.33 ±3.21	79.53 ±2.79	86.59 ±2.40	72.26 ±8.98	85.52 ±2.51
Corn	66.00 ±28.4	68.00 ±28.6	88.00 ±9.32	76.00 ±16.73	78.00 ±14.97	60.00 ±23.1	64.00 ±18.4	82.00 ±17.5	78.00 ±17.5	**93.33 ±9.43**	88.00 ±12.0	80.00 ±19.8	88.00 ±12.0	82.00 ±16.3	84.00 ±14.7
Grass-pasture	89.93 ±2.17	90.07 ±2.24	94.68 ±2.23	91.65 ±3.59	95.96 ±1.68	74.39 ±4.37	76.26 ±1.95	93.88 ±2.36	92.30 ±3.19	**96.88 ±1.07**	96.76 ±0.85	96.69 ±1.18	96.91 ±0.96	94.81 ±2.73	96.69 ±0.82
Grass-trees	97.00 ±0.99	96.98 ±1.01	99.00 ±0.44	96.24 ±0.97	98.03 ±0.86	95.07 ±0.84	95.07 ±0.94	97.81 ±1.13	96.69 ±1.18	**98.99 ±0.46**	98.19 ±0.86	98.14 ±0.91	97.83 ±0.66	98.95 ±1.25	98.79 ±0.89
Grass-pasture-mowed	90.00 ±7.91	90.00 ±7.91	92.5 ±6.46	85.00 ±13.7	89.50 ±7.91	90.00 ±7.19	90.00 ±7.19	90.00 ±7.19	87.50 ±10.2	91.67 ±5.89	87.50 ±11.8	88.75 ±9.22	90.50 ±6.46	87.42 ±12.3	**92.50 ±6.85**
Hay-windrowed	95.54 ±1.81	95.69 ±1.78	98.69 ±0.89	98.77 ±0.88	98.69 ±0.14	99.62 ±0.16	99.62 ±0.40	99.46 ±0.52	98.31 ±1.69	**99.49 ±0.36**	98.23 ±0.96	97.69 ±1.09	97.38 ±0.49	95.07 ±2.11	99.23 ±0.55
Oats	60.00 ±21.1	58.00 ±22.0	90.00 ±9.54	92.00 ±6.95	94.00 ±2.80	90.00 ±6.71	90.00 ±9.54	92.00 ±4.51	90.00 ±6.71	100.00 ±0.00	96.00 ±3.43	96.00 ±2.23	100.0 ±0.00	88.67 ±16.3	100.0 ±0.00
Soybean-notill	80.24 ±3.21	79.90 ±4.00	86.86 ±1.24	85.09 ±4.81	88.01 ±0.69	71.04 ±7.38	75.10 ±6.68	85.60 ±0.77	86.87 ±14.6	**90.35 ±1.28**	86.24 ±1.80	85.60 ±2.46	90.41 ±1.41	89.80 ±2.41	90.19 ±1.59
Soybean-mintill	50.40 ±2.71	49.69 ±2.14	66.01 ±1.55	64.38 ±2.76	67.71 ±1.41	66.48 ±3.09	64.30 ±1.99	65.81 ±1.28	67.59 ±5.04	**77.90 ±1.41**	73.02 ±3.53	72.85 ±3.15	74.91 ±1.67	72.19 ±4.60	77.48 ±2.08
Soybean-clean	50.72 ±2.06	49.96 ±2.39	90.91 ±1.48	85.78 ±26.72	92.11 ±2.51	39.73 ±7.78	43.04 ±3.94	90.65 ±2.78	86.92 ±5.59	95.82 ±1.82	89.96 ±3.52	85.97 ±3.18	95.67 ±2.23	90.78 ±5.96	**95.89 ±2.68**
Wheat	96.55 ±2.49	96.73 ±2.54	98.73 ±1.23	96.00 ±2.17	97.91 ±0.89	98.55 ±1.15	98.00 ±1.81	98.73 ±0.88	97.09 ±2.99	**98.59 ±0.85**	98.00 ±1.81	98.00 ±1.81	98.55 ±1.15	98.97 ±3.58	98.18 ±1.28
Woods	93.33 ±1.41	93.23 ±1.45	96.99 ±0.89	97.32 ±0.73	97.81 ±0.39	93.27 ±1.78	91.97 ±2.08	95.94 ±0.59	96.87 ±0.87	98.55 ±0.60	98.54 ±0.41	97.98 ±1.08	98.58 ±0.53	98.57 ±0.81	**98.64 ±0.28**
Buildings-Grass-Trees	23.88 ±10.2	24.08 ±10.8	71.02 ±7.62	68.57 ±7.16	81.59 ±4.08	6.330 ±4.13	12.65 ±3.82	78.98 ±4.82	67.55 ±5.13	**87.41 ±4.15**	69.39 ±9.33	68.98 ±9.66	86.53 ±4.64	70.37 ±4.96	84.90 ±4.47
Stone-Steel-Towers	94.19 ±3.68	94.19 ±3.27	87.91 ±3.25	90.70 ±3.67	95.95 ±3.48	93.95 ±3.83	94.42 ±3.67	95.12 ±3.55	92.33 ±3.81	**98.06 ±1.09**	97.91 ±2.31	97.44 ±2.78	96.98 ±3.30	95.02 ±3.89	95.35 ±3.67
Time (min.)	0.29	0.45	1.16	1.32	4.24	3.05	6.31	5.85	5.53	7.63	3.32	4.23	4.36	5.2	7.12

TABLE VII
CLASSIFICATION RESULTS OBTAINED BY DIFFERENT FE APPROACHES ON THE UNIVERSITY OF PAVIA DATA SET

Classifier	KNN					LR					RBF-SVM				
FE method	\	PCA	FA	LLE	CNN	\	PCA	FA	LLE	CNN	\	PCA	FA	LLE	CNN
OA(%)	84.00 ±0.29	85.32 ±0.91	88.13 ±0.20	84.08 ±1.61	90.23 ±0.23	82.04 ±0.48	78.75 ±0.58	84.24 ±0.2	83.39 ±0.11	**92.28 ±0.17**	90.02 ±0.32	90.10 ±0.22	90.56 ±0.17	85.46 ±0.21	91.52 ±0.13
AA(%)	84.08 ±0.59	85.71 ±1.45	87.79 ±0.26	89.41 ±3.43	89.91 ±0.32	68.92 ±0.20	84.38 ±0.37	87.40 ±0.05	87.09 ±0.40	**92.55 ±1.57**	90.16 ±0.24	90.17 ±0.13	89.55 ±1.41	86.60 ±1.57	91.25 ±1.75
K×100	79.46 ±0.39	81.18 ±2.00	85.28 ±0.26	64.99 ±7.92	87.41 ±0.30	75.10 ±0.60	73.50 ±0.66	80.12 ±0.28	79.13 ±0.21	**90.37 ±1.01**	88.00 ±0.41	88.06 ±0.45	89.05 ±0.87	81.18 ±2.20	89.66 ±0.97
Asphalt	85.67 ±1.40	85.89 ±2.28	86.16 ±0.82	96.39 ±1.50	87.48 ±0.83	73.86 ±0.81	76.76 ±0.87	80.85 ±0.27	80.54 ±0.30	**92.06 ±0.59**	87.67 ±0.64	87.73 ±0.61	89.58 ±0.54	83.21 ±2.43	91.07 ±0.90
0Meadows	91.84 ±0.93	92.32 ±1.65	92.04 ±0.37	91.82 ±2.12	93.99 ±0.44	61.77 ±1.70	71.27 ±1.43	80.59 ±1.02	78.70 ±0.66	**92.80 ±0.20**	91.24 ±0.65	91.30 ±0.52	91.50 ±0.57	91.14 ±1.46	92.20 ±0.67
Gravel	81.31 ±1.99	80.46 ±1.64	77.66 ±1.48	89.83 ±1.98	81.14 ±1.68	39.89 ±3.13	60.92 ±2.96	69.49 ±1.28	70.25 ±0.48	**83.67 ±1.30**	80.67 ±0.82	80.73 ±0.98	78.78 ±2.71	80.47 ±1.86	79.22 ±3.78
Trees	88.00 ±1.58	90.05 ±2.71	90.15 ±0.52	83.89 ±12.3	92.07 ±0.41	94.34 ±0.30	93.17 ±0.66	95.85 ±0.9	94.75 ±1.70	93.85 ±0.44	93.67 ±0.45	93.47 ±0.69	93.92 ±0.37	92.54 ±1.75	**95.47 ±0.30**
Metal Sheets	98.14 ±0.36	99.14 ±0.41	97.57 ±0.10	81.45 ±0.23	**99.64 ±0.08**	98.79 ±0.18	98.81 ±0.37	99.12 ±0.20	99.38 ±0.95	98.91 ±0.26	96.43 ±0.31	96.47 ±0.30	96.72 ±0.24	98.48 ±0.38	97.55 ±0.39
Bare Soil	64.69 ±2.08	68.63 ±2.07	88.09 ±0.88	98.91 ±1.57	90.79 ±1.02	86.05 ±0.80	82.87 ±0.73	88.85 ±0.27	88.98 ±0.65	**94.17 ±0.61**	90.90 ±1.02	91.17 ±0.78	92.10 ±0.62	70.34 ±3.24	92.95 ±0.90
Bitumen	85.32 ±1.32	86.25 ±1.92	89.45 ±1.14	87.78 ±2.14	89.81 ±0.97	90.76 ±0.80	87.51 ±1.58	87.74 ±1.78	86.14 ±1.01	**92.68 ±1.21**	90.37 ±0.70	90.49 ±0.47	89.34 ±1.36	89.60 ±1.97	89.54 ±1.83
Bricks	65.73 ±3.16	68.98 ±2.50	72.60 ±2.88	76.03 ±3.03	85.28 ±3.35	63.91 ±1.02	74.99 ±0.57	77.57 ±0.55	71.47 ±2.17	**89.09 ±1.09**	85.09 ±0.77	85.27 ±0.79	85.08 ±1.02	74.43 ±0.72	86.85 ±1.02
Shadow	96.00 ±1.14	97.57 ±1.28	96.54 ±0.40	90.60 ±1.10	**98.23 ±0.40**	98.97 ±0.14	98.98 ±0.35	98.51 ±0.32	98.58 ±0.64	97.84 ±0.85	95.41 ±0.74	95.55 ±0.56	95.62 ±0.73	99.18 ±0.13	96.37 ±0.11
Time(min)	2.78	2.9	3.27	3.17	14.37	9.64	10.47	11.98	12.58	16.58	5.28	5.85	6.12	6.23	15.13

TABLE VIII
CLASSIFICATION RESULTS OBTAINED BY DIFFERENT FE APPROACHES ON THE KSC DATA SET

Classifier	KNN					LR					RBF-SVM				
FE method	\	PCA	FA	LLE	CNN	\	PCA	FA	LLE	CNN	\	PCA	FA	LLE	CNN
OA(%)	80.29 ±1.00	80.07 ±0.92	85.42 ±0.74	82.96 ±0.45	86.80 ±2.43	81.33 ±1.17	77.86 ±0.92	86.60 ±0.94	82.67 ±1.20	**89.23 ±0.91**	87.72 ±0.73	86.78 ±0.70	87.50 ±1.33	83.13 ±1.22	87.64 ±0.96
AA(%)	72.27 ±1.30	72.07 ±1.34	78.95 ±1.57	77.41 ±1.78	79.98 ±4.04	71.62 ±1.96	67.52 ±1.79	81.04 ±1.96	79.23 ±1.73	**83.32 ±1.28**	83.19 ±1.10	81.86 ±1.27	83.24 ±2.24	80.80 ±1.24	81.56 ±1.53
$K\times100$	78.09 ±1.11	77.85 ±1.02	84.01 ±0.82	76.17 ±0.51	85.31 ±2.71	79.20 ±1.30	75.31 ±1.03	85.44 ±1.04	76.96 ±1.32	**86.91 ±1.01**	87.35 ±0.81	87.31 ±0.78	86.89 ±1.48	79.46 ±1.35	86.24 ±1.37
Scrub	90.00 ±3.90	90.06 ±4.21	89.24 ±2.24	88.63 ±2.13	**94.30 ±4.37**	96.37 ±0.74	94.90 ±1.11	85.22 ±3.48	79.88 ±5.70	93.35 ±4.80	91.56 ±3.07	89.71 ±3.27	84.48 ±4.53	87.43 ±3.18	85.44 ±3.18
Willow swamp	85.27 ±3.92	86.01 ±3.42	90.27 ±4.69	79.97 ±5.01	92.79 ±5.49	94.19 ±1.68	91.73 ±2.84	88.00 ±5.18	89.41 ±5.92	92.52 ±4.46	87.68 ±3.03	85.41 ±4.55	90.18 ±3.84	84.37 ±5.21	**94.39 ±1.16**
CP hammock	82.37 ±4.23	82.95 ±3.86	87.78 ±2.73	81.31 ±3.67	93.89 ±1.55	79.96 ±8.20	77.37 ±8.97	83.47 ±6.77	74.34 ±7.24	**96.07 ±3.01**	83.92 ±7.93	86.12 ±2.26	86.35 ±3.67	72.26 ±8.98	92.58 ±2.51
Slash pine	48.55 ±5.20	47.66 ±5.95	54.80 ±7.19	46.00 ±6.13	54.31 ±7.93	48.11 ±15.1	49.61 ±5.82	61.47 ±5.04	45.78 ±6.51	**59.22 ±5.64**	64.17 ±6.92	68.99 ±5.54	67.31 ±5.83	58.40 ±6.33	53.44 ±14.7
Oak/ Broadleaf	40.48 ±5.48	40.94 ±5.56	44.03 ±5.79	43.65 ±3.59	**61.70 ±11.3**	38.63 ±14.8	33.63 ±15.8	51.11 ±7.10	42.32 ±3.19	58.86 ±6.30	50.83 ±4.54	55.55 ±6.48	58.58 ±8.00	54.81 ±3.73	57.45 ±5.82
Hardwood	29.42 ±9.16	29.2 ±9.37	38.92 ±7.09	26.24 ±10.9	22.27 ±18.4	32.90 ±10.2	31.64 ±11.5	63.91 ±3.20	36.89 ±1.68	**50.23 ±5.65**	56.81 ±5.94	62.68 ±8.8	60.53 ±11.0	56.95 ±7.25	45.49 ±0.89
Swamp	46.98 ±13.6	45.83 ±13.6	69.56 ±21.0	54.00 ±13.7	55.91 ±15.7	19.90 ±15.5	6.56 ±10.3	67.31 ±20.1	57.50 ±11.2	**89.25 ±9.39**	65.00 ±7.17	65.42 ±12.1	65.65 ±15.7	67.42 ±12.3	87.74 ±6.85
Graminoid marsh	79.66 ±5.16	79.8 ±5.45	88.34 ±2.65	81.77 ±2.88	**90.96 ±3.29**	60.57 ±5.19	49.74 ±3.34	86.89 ±3.05	79.31 ±1.29	86.01 ±2.71	88.52 ±2.33	86.88 ±2.3	89.13 ±2.58	81.07 ±2.11	85.42 ±0.55
Spartina marsh	85.67 ±4.91	85.03 ±4.53	94.50 ±1.60	92.00 ±6.95	92.44 ±1.80	90.41 ±2.70	85.54 ±1.98	93.97 ±1.83	90.03 ±1.71	94.60 ±1.98	94.01 ±2.03	88.27 ±1.62	92.50 ±1.66	86.67 ±1.30	**94.81 ±0.00**
Cattail marsh	85.51 ±3.61	85.24 ±3.71	94.22 ±1.44	85.09 ±4.81	94.50 ±1.95	89.59 ±2.45	85.53 ±2.43	92.89 ±2.28	86.87 ±3.46	**95.15 ±1.63**	94.66 ±1.80	93.97 ±1.75	95.06 ±1.01	87.80 ±2.41	92.03 ±1.59
Salt marsh	90.40 ±3.64	89.77 ±3.30	89.43 ±3.98	94.38 ±2.76	**97.88 ±2.77**	95.00 ±2.36	92.67 ±3.03	90.60 ±4.34	87.59 ±5.04	97.35 ±3.08	95.69 ±2.66	96.06 ±2.92	92.82 ±2.66	91.19 ±4.60	97.62 ±2.08
Mud flats	75.35 ±3.92	74.52 ±3.41	87.27 ±4.24	85.78 ±6.72	85.93 ±5.87	85.02 ±1.95	78.79 ±2.79	91.69 ±1.79	86.92 ±1.59	90.77 ±1.52	90.44 ±2.01	90.11 ±1.49	92.00 ±1.83	90.87 ±1.96	**94.07 ±2.68**
Water	99.88 ±0.00	99.88 ±0.00	100.0 ±0.00	94.70 ±0.61	100.0 ±0.00	100.0 ±0.00	100.0 ±0.00	96.95 ±0.06	95.33 ±0.81	100.0 ±0.00	100.0 ±0.00	100.0 ±0.00	100.0 ±0.00	95.02 ±0.89	99.76 ±0.67
Time (min.)	0.14	0.72	0.56	1.09	3.94	1.15	1.31	1.26	2.15	4.38	2.48	2.87	2.54	3.02	4.79

Tables VI–VIII show that the CNN-based FE methods always provide the best performances of OA, AA, and Kappa for all three data sets. The classification accuracy values are given in the form of mean ± standard deviation from the perspective of statistics, which is used as a measurement of volatility.

In order to have a fair comparison, we used 10% of the training samples to find the best parameters of FE methods using grid search. The result reported in Tables VI–VIII are the best classification results when the number of features was properly selected for each FE method. On the selection of parameters, the number of features was chosen in the range of 10 to N (i.e., the number of hyperspectral bands) with an interval of 10. The number of neighbors in LLE has been changed in a range from 1 to 10. For KNN, the range of the nearest neighbors has been changed from 1 to 30 with the interval of two. In RBF-SVM, there are two parameters, i.e., C and γ [48]; thus, we applied 2-D grid search from a wide range (i.e., $C=2^{-5}, 2^{-4}, \ldots, 2^{19}$; $\gamma = 2^{-15}, 2^{-14}, \ldots 2^{4}$) to get the best parameters. The learning rate and the number of epochs for LR were selected empirically.

Table VI also shows the results obtained in a situation when the models were trained using the original complete set of spectral bands (200 bands) of the Indian Pines data set. Due to the imbalance between the numbers of training samples and the numbers of bands used, the accuracy and its corresponding variance have a wide range from one class to another one. Compared with PCA, FA, and LLE, CNN-based FE leads to better performance, particularly when it combines with LR. The CNN-LR exhibits the highest OA, AA, and K, the highest percentage of correctly classified pixels among all the test pixels considered, with an improvement of 3.92%, 3.88%, and 0.046 over the RBF-SVM, respectively.

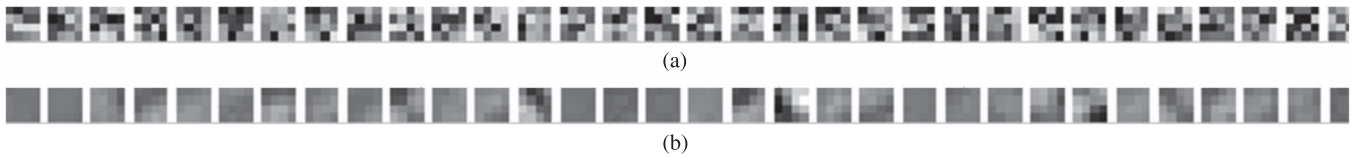
TABLE IX
ARCHITECTURE OF THE 2-D CONVOLUTION NEURAL NETWORK

No.	Convolution	ReLU	Pooling	Dropout
1	4×4×32	Yes	2×2	No
2	5×5×64	Yes	2×2	50%
3	4×4×128	Yes	No	50%

Table VII shows the experimental results for the University of Pavia data set. It is shown that the CNN-LR provides better results again and outperforms RBF-SVM by 2.26%, 2.39%, and 0.0237 on average in terms of OA, AA, and K, respectively. It is worth noting that the obtained variance is very small. In terms of class accuracy values, the class "Bricks" was the most difficult one to be classified. The CNN-LR still exhibits the best accuracy (89.09 ± 1.18) for this class. Concerning computational cost, CNN has the longest processing time (given by the sum of the training and test times) compared with the other methods.

C. CNN With Spatial Features

In this section, we investigate the effectiveness of the 2-D CNN for hyperspectral data FE and classification. There are two reasons. On one hand, the original CNN is designed for 2-D image classification. The usefulness of 2-D CNN for HSI classification should be tested. In this paper, 1-D CNN and 3-D CNN are designed for spectral classification and spectral–spatial classification, respectively. On the other hand, to maintain the integrity, 2-D CNN should be investigated. There are several



(a)

(b)

Fig. 14. Weights of the first convolutional layer. Each tiny image (4×4) stands for a convolutional kernel. There are 32 kernels in the first convolutional layer. The intensity of each pixel stands for the value of corresponding weight. (a) Randomly initialized weights of the first convolution layer of the University of Pavia data set. (b) Learned weights of the first convolution layer of the University of Pavia data set.

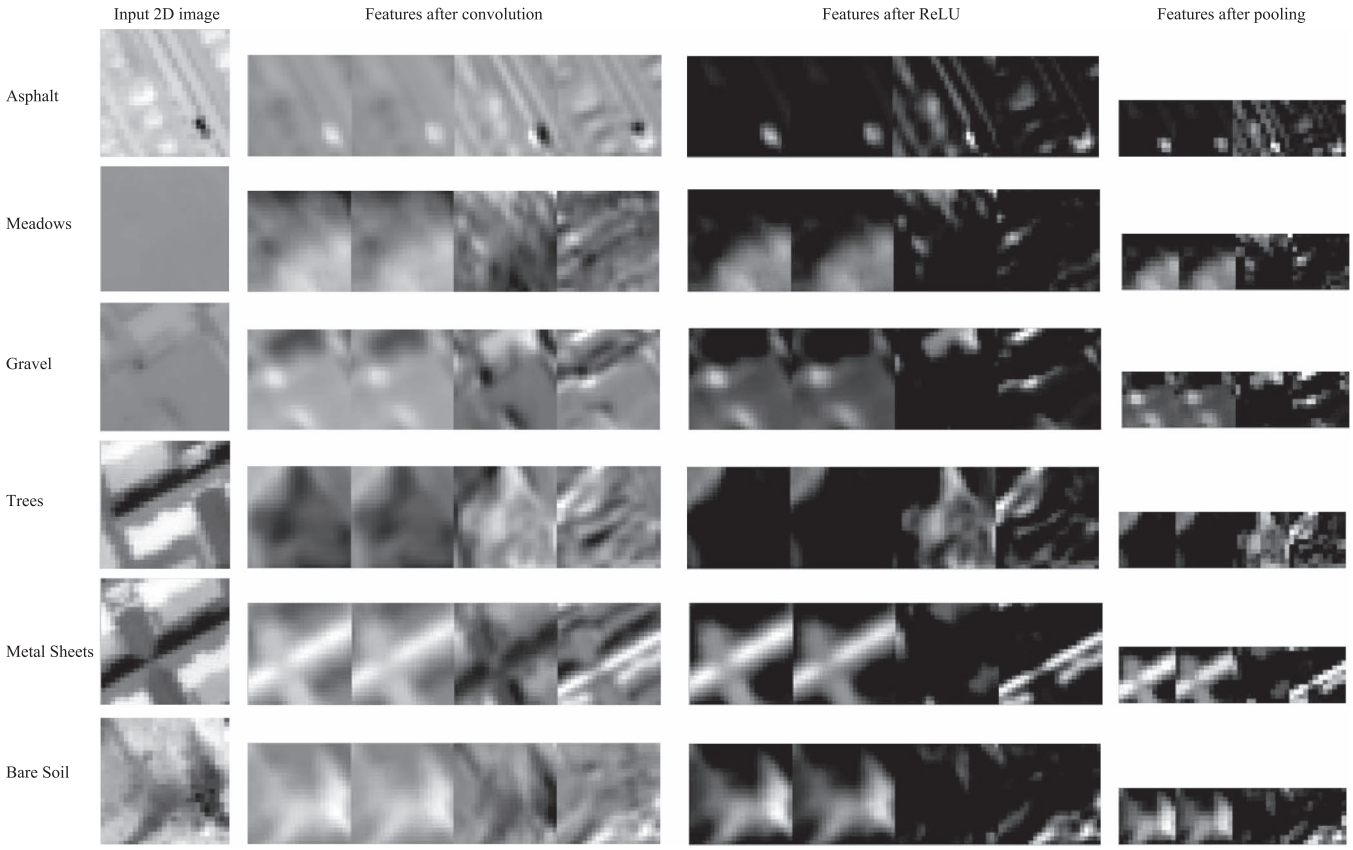


Fig. 15. Extracted features of the University of Pavia data set. There are six rows, and each row of images represents one class. There are four columns in the figure. The first column is allocated to the input images. The second column is allocated to the four feature maps after the first convolution. The third column is allocated to the four feature maps after the first ReLU operation. The last column is composed of the four features of the first pooling operation.

approaches to create 2-D input such as choose one band of HSI randomly. The first principal component is used to create the 2-D input because the first principal component contains the most energy of the whole HSI. The learned spatial features are used for further classification.

1) Architecture Design, Visualization, and Analysis: There are several factors that need to be selected in the experiments. The input images were normalized into $[-0.5\ 0.5]$. We used a large neighborhood window (27×27) for the first principal component as the input 2-D image for the three data sets.

The details of the architecture of the CNN are listed in Table IX. Because of the small size of the input image, we used three convolution layers and three pooling layers. After the CNN, the input image was converted into a vector with 128 dimensions.

In the training procedure, we used a mini-batch-based back-propagation method, whereas the size of mini-batch is 100. The learning rate of all CNNs is set to be 0.01. In this part of the experiment, the number of training epochs of the CNN is 200.

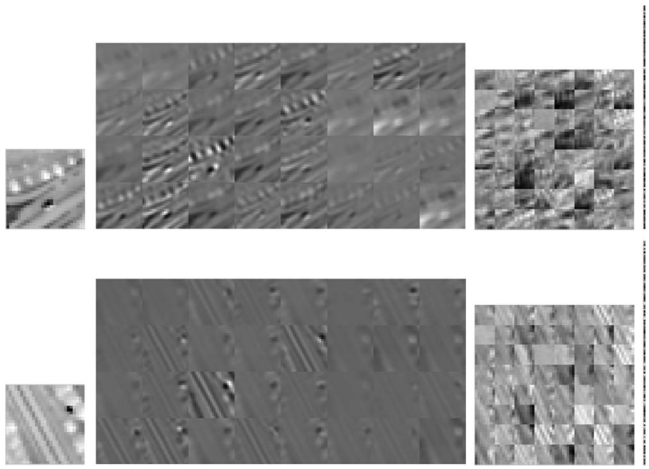


Fig. 16. Extracted features after three convolutional layers on two asphalt samples. The number of feature maps after the first convolution layer is 32, and the size of each feature map is 24×24; the number of feature maps after the second convolution layer is 64, and the size of each feature map is 8×8; and the number of feature maps after the second convolution layer is 128, and the size of each feature map is 1×1.

TABLE X
SIMILARITY AND DIVISIBILITY OF SPATIAL FEATURES ON THE UNIVERSITY OF PAVIA DATA SET

Quantitative Methods	Mean of Correlation Coefficients (similarity)			J-M distance (divisibility)		
	Class1	Class2	Class5	Class (1,2)	Class(1,5)	Class(2,5)
Spatial input	0.2462	0.5045	0.3543	0.3452	0.6479	0.5375
Features after layer1	0.4354	0.3856	0.5434	0.8424	0.5241	0.3457
Features after layer2	0.3943	0.7549	0.4671	0.5246	0.4654	0.9454
Features after layer3	0.8437	0.8764	0.8535	0.8742	0.9345	1.0941

TABLE XI
CLASSIFYING WITH SPATIAL FEATURES ON THE INDIAN DATA SET

Method	RBF-SVM	EMP-RBF-SVM	2D-CNN-LR
OA(%)	81.14±1.47	84.47±0.99	**89.99±1.62**
AA(%)	85.28±1.84	87.17±1.29	**97.19±0.38**
K×100	83.73±1.23	86.72±1.07	**87.95±1.90**
Alfalfa	78.83±3.52	82.57±2.69	**99.65±1.47**
Corn-notill	71.37±2.47	77.28±1.69	**90.64±1.75**
Corn-mintill	84.43±1.89	86.38±1.57	**99.11±0.82**
Corn	82.00±16.3	88.00±10.2	**100.00±0.00**
Grass-pasture	96.25±0.72	97.37±0.54	**98.48±1.61**
Grass-trees	95.61±0.79	93.25±1.16	**97.95±2.13**
Grass-pasture-mowed	75.25±11.7	78.57±5.94	**100.00±0.00**
Hay-windrowed	94.36±1.65	95.85±1.06	**100.00±0.00**
Oats	92.00±7.21	95.00±0.87	**100.00±0.00**
Soybean-notill	85.32±2.06	86.82±1.59	**95.33±2.64**
Soybean-mintill	74.21±3.36	**78.23±2.30**	78.21±4.93
Soybean-clean	84.47±2.50	82.59±1.62	**99.39±0.67**
Wheat	98.11±1.10	98.31±1.35	**100.00±0.00**
Woods	94.25±2.70	91.38±2.54	**97.71±1.30**
Buildings-Grass-Trees	62.45±1.59	66.44±1.28	**99.31±1.40**
Stone-Steel-Towers	95.62±0.36	96.67±0.69	**99.22±1.12**
Runtime (min.)	2.77	3.22	5.95

TABLE XII
CLASSIFYING WITH SPATIAL FEATURES ON THE UNIVERSITY OF PAVIA DATA SET

Method	RBF-SVM	EMP-RBF-SVM	2D-CNN-LR
OA(%)	91.43±0.61	93.03±0.47	**94.04±0.69**
AA(%)	93.87±0.87	96.88±0.79	**97.52±0.25**
K×100	89.83±1.90	90.28±1.23	**92.43±0.86**
Asphalt	95.25±1.23	96.35±0.92	**97.11±1.04**
Meadows	87.28±1.18	**90.18±1.39**	87.66±1.47
Gravel	97.56±0.50	97.15±0.79	**99.69±0.28**
Trees	96.93±0.40	98.26±0.52	**98.49±0.36**
Metal Sheets	99.23±0.63	100.00±0.00	100.00±0.00
Bare Soil	92.25±0.72	96.25±0.76	**98.00±0.73**
Bitumen	94.92±0.35	99.19±0.72	**99.89±0.15**
Bricks	97.50±0.79	98.38±0.82	**99.70±0.27**
Shadow	95.11±1.10	96.17±1.35	**97.11±1.46**
Runtime (min.)	6.57	8.27	10.12

In the visualization part, we used the University of Pavia data set as an example. The power of the neural network lies in the weights. In the beginning, the weights are randomly initialized with standard deviation 0.001. The weights of different convolutional kernels in the first convolutional layer are shown in Fig. 14. The initial weights, which are shown in Fig. 14(a), are random; the learned weights are with obvious structures.

TABLE XIII
CLASSIFYING WITH SPATIAL FEATURES ON THE KSC DATA SET

Method	RBF-SVM	EMP-RBF-SVM	2D-CNN-LR
OA(%)	88.09±0.58	93.15±0.85	**94.11±0.90**
AA(%)	82.51±1.07	90.51±1.11	**91.98±1.34**
K×100	86.74±0.64	85.36±1.73	**93.44±1.00**
Scrub	78.05±4.77	**92.02±2.79**	89.43±4.20
Willow swamp	75.72±4.65	**90.98±3.48**	87.55±9.13
CP hammock	82.04±5.72	83.07±3.50	**88.19±7.65**
Slash pine	72.20±7.25	79.19±6.91	**81.00±6.68**
Oak/Broadleaf	69.86±9.35	83.76±6.65	**96.37±5.18**
Hardwood	71.91±7.89	**84.80±4.34**	84.42±6.78
Swamp	61.52±14.7	**86.74±9.39**	86.67±9.58
Graminoid marsh	84.80±5.02	93.19±2.51	**96.65±2.49**
Spartina marsh	95.84±2.68	95.30±1.85	**97.05±2.10**
Cattail marsh	91.88±3.63	95.98±2.76	**96.50±2.42**
Salt marsh	91.09±3.60	94.81±5.16	**94.92±2.95**
Mud flats	**97.76±2.85**	96.91±2.62	96.94±2.72
Water	100.00±0.00	99.89±0.04	100.00±0.00
Runtime (min.)	1.21	1.96	3.01

TABLE XIV
ARCHITECTURE OF THE 3-D CONVOLUTION NEURAL NETWORK

Data set	No.	Convolution	ReLU	Pooling	Dropout
Indian Pines	1	4×4×32×128	Yes	2×2	No
	2	5×5×32×192	Yes	2×2	50%
	3	4×4×32×256	Yes	No	50%
Pavia University	1	4×4×32×32	Yes	2×2	No
	2	5×5×32×64	Yes	2×2	50%
	3	4×4×32×128	Yes	No	50%
KSC	1	4×4×32×32	Yes	2×2	No
	2	5×5×32×64	Yes	2×2	50%
	3	4×4×32×128	Yes	No	50%

Fig. 15 shows the features after the convolutional layer, ReLU layer, and pooling layer. The first column shows the inputs of different classes, which contains nine 27 × 27 small images. After convolution processing of 32 kernels, which are shown in Fig. 14(b), the input is converted into 32 feature maps. Four of the feature maps, which are 24 × 24 images after the 4 × 4 convolution processing, are shown in the second column in Fig. 15. The ReLU layer cuts off the features that are less than 0, and the features are showed in the third column in Fig. 15. After the max pooling operation, the size of feature maps is 12 × 12. Different convolutional kernels extract different features of the input, and the abundant features give a lot of potential for further processing.

Extracted features after three convolutional layers of two asphalt samples are shown in Fig. 16. The convolutional operation can extract different features according to different

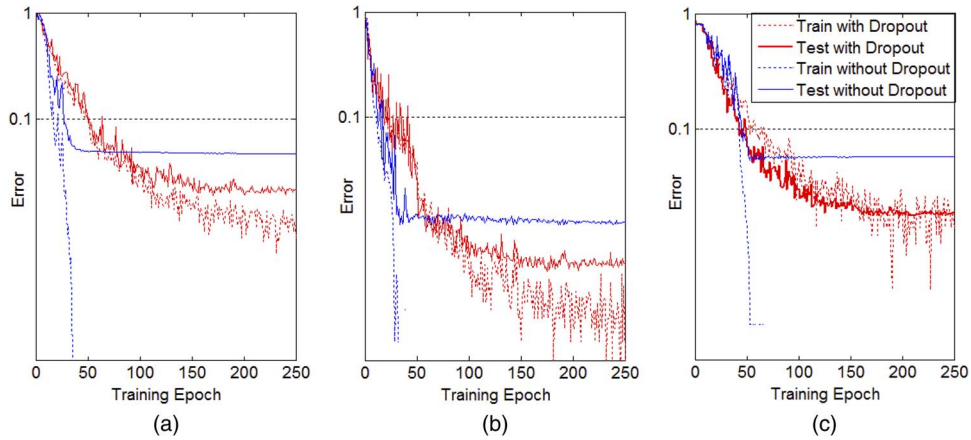


Fig. 17. Classification results with and without dropout on the (left) Indian Pines, (middle) University of Pavia, and (right) KSC data sets.

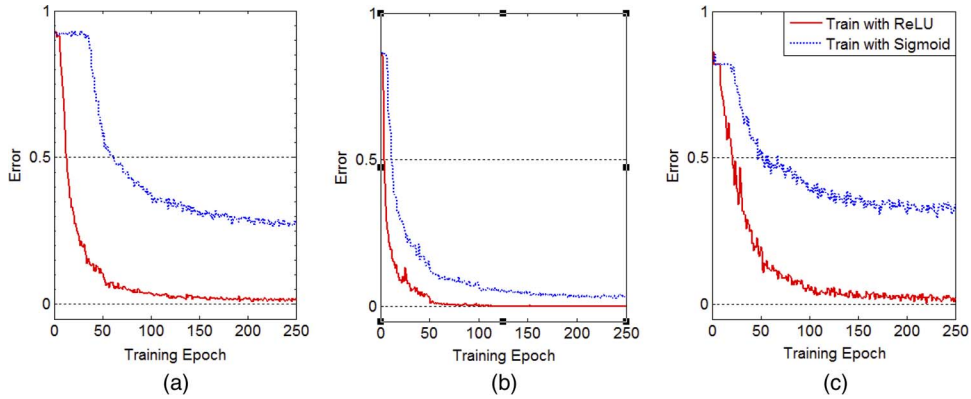


Fig. 18. Training error with and without ReLU on the (left) Indian Pines, (middle) University of Pavia, and (right) KSC data sets.

convolutional kernels. The correlation coefficient of the two input images is 0.2975, which means that they are quite different. After three convolutions, the correlation coefficient of the two inputs is 0.8324, which means they are quite similar. This kind of processing is very useful for further classification.

In order to evaluate the effectiveness of the extracted features, the similarity in the same class and the divisibility between the different classes are shown in Table X in a quantitative way. From Table X, after convolution operations, the similarity in the same class and the divisibility in different classes are increased. While some features in the middle layers have relatively low similarity in the same class and relatively low divisibility in the different classes, the features in the middle layers are not suitable for classification.

In Tables XI–XIII, we can see that the deep CNN method outperforms RBF-SVMs in terms of the OA, AA, and Kappa. In this case, the CNN significantly improves RBF-SVM for all three data sets.

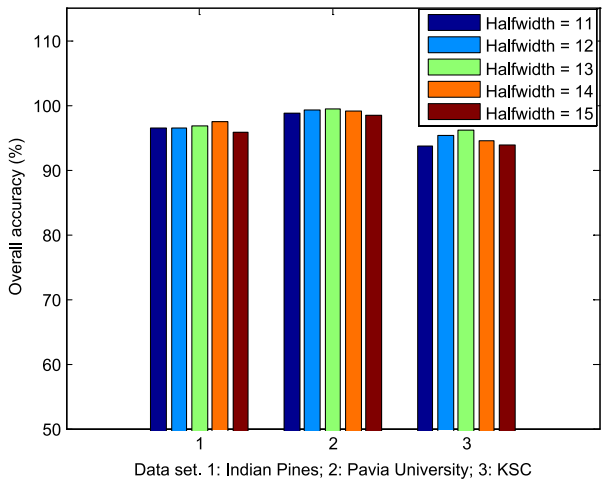


Fig. 19. Influence of spatial size on the (left) Indian Pines, (middle) University of Pavia, and (right) KSC data sets. Best view in color.

D. CNN With Spatial–Spectral Features

In this part of the experiments, we investigate the advantage of 3-D CNN for HSI FE and classification. With the help of proper CNN architecture, we used the neighbors of the pixel in all bands. The CNN learns the spectral and spatial features by itself, and learned features were used in classification.

1) Architecture Design and Parameter Analysis: For the Indian Pines, University of Pavia, and KSC data sets, we use $27 \times 27 \times 200$, $27 \times 27 \times 103$, and $27 \times 27 \times 176$ neighbors of each pixel as the input 3-D images, respectively. The input images are normalized into $[-0.5\ 0.5]$. The structures' details are given in Table XIV. After the 3-D CNN, the input image was converted into a vector. The size of mini-batch was 100,

TABLE XV
SIMILARITY AND DIVISIBILITY OF SPECTRAL–SPATIAL FEATURES ON THE UNIVERSITY OF PAVIA DATA SET

Quantitative Methods	Mean of Correlation Coefficients (similarity)			J-M distance (divisibility)		
	Class1	Class2	Class5	Class (1,2)	Class (1,5)	Class (2,5)
Spectral-spatial input	0.3712	0.6496	0.5201	0.3153	0.3145	0.5934
Features after layer1	0.6122	0.8317	0.4390	0.9842	0.7124	0.6424
Features after layer2	0.3898	0.7834	0.5143	0.7742	0.8424	0.6287
Features after layer3	0.9306	0.9447	0.8720	1.0742	1.1943	1.2458

and the learning rate was 0.003. In this set of experiments, the number of training epochs CNNs is 400.

There are three factors (dropout, ReLU, and the size of the spatial window) that influence the final classification accuracy significantly, and they are analyzed in the following.

In the proposed architecture, dropout plays an important role to address overfitting. In this experiment, the results (classification error) with and without dropout on the three data sets are presented in Fig. 17. In the figure, the training errors without dropout regularization are very low after dozens of epochs, whereas the test errors without dropout are very high. This is the problem of overfitting. For the training and test errors with dropout, the training errors are relatively high, whereas the test errors are relatively low. This means that the model with dropout has a good capability of generalization.

The effectiveness of the dropout can be explained in two ways. The first one is to prevent co-adaptations of the units on the training samples, and the second one is to average the predictions of many different networks [43]. If a hidden unit knows its collaborative units, it leads to good performance on the training data. However, these units might not perform well on the test data set. However, if a hidden unit adapts well on many different collaborative units, it will be more dependent on itself rather than depending on some certain combinations of hidden units. Dropout strategy makes it possible to train different networks, and each network gets a classification result. As the training procedure continues, most of the networks give the correct results to eliminate incorrect results on the final classification results.

ReLU is another important factor that is influential to final performance. Krizhevsky *et al.* claimed that the nonsaturating nonlinear function as ReLU can gain better performances than these saturating nonlinearities such as sigmoid function [26]. The classification errors with and without ReLU on the three data sets are demonstrated in Fig. 18. From Fig. 18, convergence of the models with sigmoid function are slower than convergence of the models with ReLU. In particular, on the Indian Pines data set, a CNN with ReLU (red solid lines) reaches a 50% error rate six times faster than the same network with sigmoid (blue dashed lines). On the other hand, the models with ReLU can lead to lower training error (close to 0) at the end of training. In summary, CNN with ReLU can accelerate convergence and improve the training accuracy.

The size of 3-D input is an important parameter too. The dimensionality toward spectral dimension is fixed, whereas the dimensionalities toward spatial dimension are changeable. A set of experiments is organized to get a proper size of 3-D inputs. Fig. 19 shows the results using different sizes of spatial window. The half widths of spatial size are set to $W = [11, 12, 13, 14, 15]$, and the full width is $2W + 1$. To have a fair comparison, we resize other spatial sizes to 27×27 and get classification accuracy values using the models aforementioned. For the Indian Pines data set, the OA can reach the highest and the value is nearly 98% when the half width is 14. For the University of Pavia and KSC data sets, the results show that the best accuracy values are obtained when the half width is 13.

TABLE XVI
CLASSIFICATION WITH SPECTRAL–SPATIAL FEATURES ON THE INDIAN PINES DATA SET

Method	3D-RBF-SVM	3D-EMP-RBF-SVM	3D-CNN-LR
OA(%)	92.42±1.10	96.92±0.81	**97.56±0.43**
AA(%)	92.14±1.44	95.07±0.88	**99.23±0.19**
$K\times 100$	94.83±1.35	96.21±0.91	**97.02±0.52**
Alfalfa	85.35±1.50	95.94±0.39	**100.00±0.00**
Corn-notill	82.52±1.06	91.39±1.28	**96.34±1.11**
Corn-mintill	84.43±0.73	92.53±0.59	**99.49±0.70**
Corn	100.00±0.00	100.00±0.00	100.00±0.00
Grass-pasture	97.12±0.39	98.03±0.47	**99.91±0.23**
Grass-trees	94.53±0.54	92.35±1.16	**99.75±0.30**
Grass-pasture-mowed	94.28±2.96	95.93±1.52	**100.00±0.00**
Hay-windrowed	95.27±1.36	96.35±0.58	**100.00±0.00**
Oats	100.00±0.00	100.00±0.00	100.00±0.00
Soybean-notill	82.32±0.51	87.92±1.39	**98.72±0.95**
Soybean-mintill	86.28±3.36	92.23±0.96	**95.52±1.23**
Soybean-clean	93.86±1.61	96.85±0.51	**99.47±0.39**
Wheat	99.25±0.42	99.23±0.28	**100.00±0.00**
Woods	95.18±1.47	96.85±1.19	**99.55±0.58**
Buildings-Grass-Trees	86.98±1.28	89.28±1.01	**99.54±1.31**
Stone-Steel-Towers	96.82±0.35	96.24±0.67	**99.34±1.08**
Runtime(min.)	5.45	11.12	27.92

In order to evaluate the effectiveness of the extracted spectral–spatial features, Table XV presents the similarity in the same class and the divisibility between the different classes. Compared with Tables V and X, after convolution operations, the spectral–spatial features get the highest similarity in the same class and the highest divisibility between the different classes, which shows that the spectral–spatial features have the potential for accurate classification.

2) Comparative Experiments With Other Spectral–Spatial Methods: We also conducted RBF-SVM with the original data sets and extended morphological profile (EMP) for comparison. EMP followed by SVM is an advanced spatial–spectral classification method for hyperspectral data. We used opening and closing operations on the first five, seven, and three principal components of the Indian Pines, University of Pavia, and KSC data sets to extract structural information, respectively. In the experiments, the structuring element used was a disk and

TABLE XVII
CLASSIFICATION WITH SPECTRAL–SPATIAL FEATURES ON THE UNIVERSITY OF PAVIA DATA SET

Method	3D-RBF-SVM	3D-EMP-RBF-SVM	3D-CNN-LR
OA(%)	96.05±0.91	97.72±0.61	**99.54±0.11**
AA(%)	95.73±0.61	97.17±0.47	**99.66±0.11**
$K\times100$	95.33±1.68	95.21±0.23	**99.41±0.15**
Asphalt	93.15±0.50	94.35±0.69	**99.36±0.36**
Meadows	93.57±0.89	96.87±0.37	**99.36±0.17**
Gravel	97.56±0.50	94.76±0.51	**99.69±0.32**
Trees	96.93±0.40	98.72±0.82	**99.63±0.15**
Metal Sheets	98.23±0.63	99.02±0.36	**99.95±0.08**
Bare Soil	92.25±0.91	94.85±0.36	**99.96±0.10**
Bitumen	94.82±0.33	100.00±0.00	100.00±0.00
Bricks	95.50±0.57	96.63±0.22	**99.65±0.22**
Shadow	98.11±0.50	99.33±0.37	**99.38±0.61**
Runtime(min.)	14.10	17.23	46.15

TABLE XVIII
CLASSIFICATION WITH SPECTRAL–SPATIAL FEATURES ON THE KSC DATA SET

Method	3D-RBF-SVM	3D-EMP-RBF-SVM	3D-CNN-LR
OA(%)	93.73±0.67	95.66±0.61	**96.31±1.25**
AA(%)	90.55±1.52	93.82±0.93	**94.68±1.97**
$K\times100$	93.03±0.75	95.17±0.01	**95.90±1.39**
Scrub	87.82±4.24	**95.96±2.51**	91.71±3.71
Willow swamp	80.14±7.06	**92.14±3.38**	89.73±10.1
CP hammock	87.64±6.19	88.89±3.62	**92.16±5.33**
Slash pine	86.64±4.79	**89.28±3.43**	86.94±6.27
Oak/Broadleaf	77.02±8.87	85.25±7.48	**94.79±6.89**
Hardwood	89.66±5.82	**91.81±2.69**	90.92±7.90
Swamp	83.37±17.8	91.52±7.95	**91.57±6.14**
Graminoid marsh	91.78±2.82	95.53±2.16	**96.22±3.13**
Spartina marsh	97.12±1.57	97.25±1.11	**99.53±0.99**
Cattail marsh	97.06±1.85	97.89±2.08	**99.81±0.37**
Salt marsh	99.64±1.12	95.96±5.30	**99.79±0.30**
Mud flats	99.24±2.11	**98.28±1.66**	97.69±2.31
Water	100.00±0.00	99.90±0.05	100.00±0.00
Runtime(min.)	2.56	3.78	7.93

the structure sizes were progressively increased from 1 to 4. Therefore, 40, 56, and 24 spatial features were generated. The generated spatial features and original spectral features are used for classification. Wide ranges of c and g values for the SVM were searched in the EMP with the RBF-SVM method; for the Indian Pines data set, they were configured as $c = 2^{18}$ and $g = 2^1$, whereas those in the Pavia data set were $c = 2^{19}$ and $g = 2^1$, and in the KSC data set, they were $c = 2^{10}$ and $g = 2^{-1}$.

Tables XVI–XVIII provide information about the classification results compared with the typical SVMs. From the results, we can see that the classification accuracy values of CNN in terms of OA, AA, and Kappa coefficient are higher than those of other FE and classification methods. The results show that the designed CNN can help improve the classification accuracy of HSI.

3) Computation Cost of the 3-D CNN: In general, we concede that neural networks take longer time to train the network compared with other machine learning algorithms such as KNN or SVM, and so does the proposed deep learning methods.

TABLE XIX
RUNNING TIME COMPARISON

Data set	Indian Pines	Pavia University	KSC
Training time (min.)	27.04	45.13	7.63
Test time (min.)	0.88	1.02	0.30

TABLE XX
CLASSIFICATION ACCURACY VALUES ON THE INDIAN PINES DATA SET

Method	RBF-SVM	3D-EMP-SVM	3D-CNN	3D-CNN + Method A	3D-CNN + Method B
OA (%)	83.89±2.07	96.92±0.81	97.56±0.43	98.46±0.32	**98.53±0.29**
AA (%)	89.24±1.92	95.07±0.88	99.23±0.19	99.46±0.81	**99.50±0.080**
$K\times100$	80.7±2.37	96.21±0.91	97.02±0.52	98.11±0.39	**98.20±0.35**

TABLE XXI
CLASSIFICATION ACCURACY VALUES ON THE UNIVERSITY OF PAVIA DATA SET

Method	RBF-SVM	3D-EMP-SVM	3D-CNN	3D-CNN + Method A	3D-CNN + Method B
OA (%)	90.02±0.32	97.72±0.61	99.54±0.11	**99.66±0.03**	99.64±0.09
AA (%)	90.16±0.24	97.17±0.47	99.66±0.11	**99.77±0.08**	99.71±0.16
$K\times100$	88.00±0.41	95.21±0.23	99.41±0.15	**99.56±0.04**	99.55±0.13

TABLE XXII
CLASSIFICATION ACCURACY VALUES ON THE KSC DATA SET

Method	RBF-SVM	3D-EMP-SVM	3D-CNN	3D-CNN + Method A	3D-CNN + Method B
OA (%)	87.72±0.73	95.66±0.61	96.31±1.25	**97.07±1.02**	96.72±1.05
AA (%)	83.19±1.10	93.82±0.93	94.68±1.97	**96.17±1.31**	95.76±1.39
$K\times100$	87.35±0.81	95.17±0.01	95.90±1.39	**96.74±1.13**	96.35±1.17

On the other hand, the advantage of deep learning algorithms is that they are superfast on testing.

The training and test times are shown in Table XIX. In the table, we can see that the test time is only 0.88, 1.02, and 0.30 min for the Indian Pines, University of Pavia, and KSC data sets, respectively. Fast test time is very important in real applications.

With the quick development of hardware technology, particularly on graphic processing units, the drawback of long training time of a deep learning method can be mitigated in the near future.

E. CNN With Virtual Sample

1) Classification Results: In this part of the experiments, the advantages of 3-D CNN with virtual samples for HSI FE and classification are investigated. The two proposed virtual sample methods (Method A and Method B) in Section V are tested on the three data sets.

For every virtual sample generated by (9), α_m is a uniformly distributed random number in [0.9, 1.1], and β, which is the weight of noise n, is set to 1/25. Meanwhile, for every virtual sample generated by (10), α_i and α_j are uniformly distributed random numbers on the interval [0, 1], whereas $\mathbf{x_i}$ and $\mathbf{x_j}$ are randomly chosen from the same class.

The CNN architecture and the training procedure in this section are the same as in the previous section. Classification accuracy values obtained by different approaches on the Indian Pines, University of Pavia, and KSC data sets are shown in Tables XX–XXII.

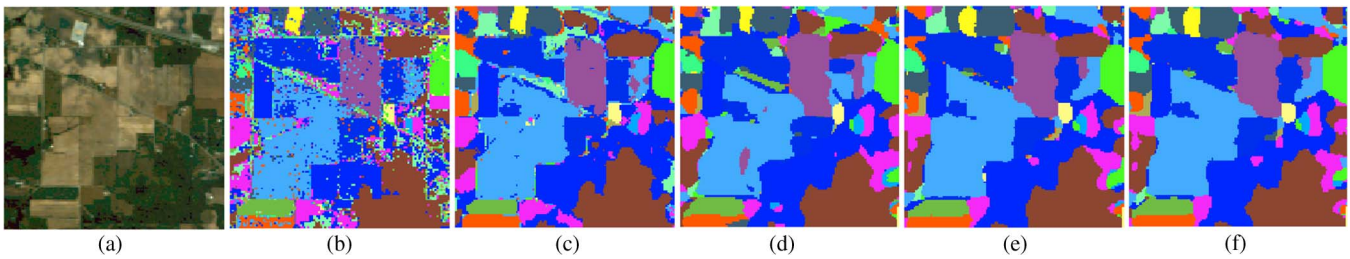


Fig. 20. Indian Pines. (a) False color image. (b)–(f) Classification maps for different classifiers: (b) 1D-SVM, (c) 3D-EMP-SVM, (d) 3D-CNN, (e) 3D-CNN with Method A, and (f) 3D-CNN with Method B.

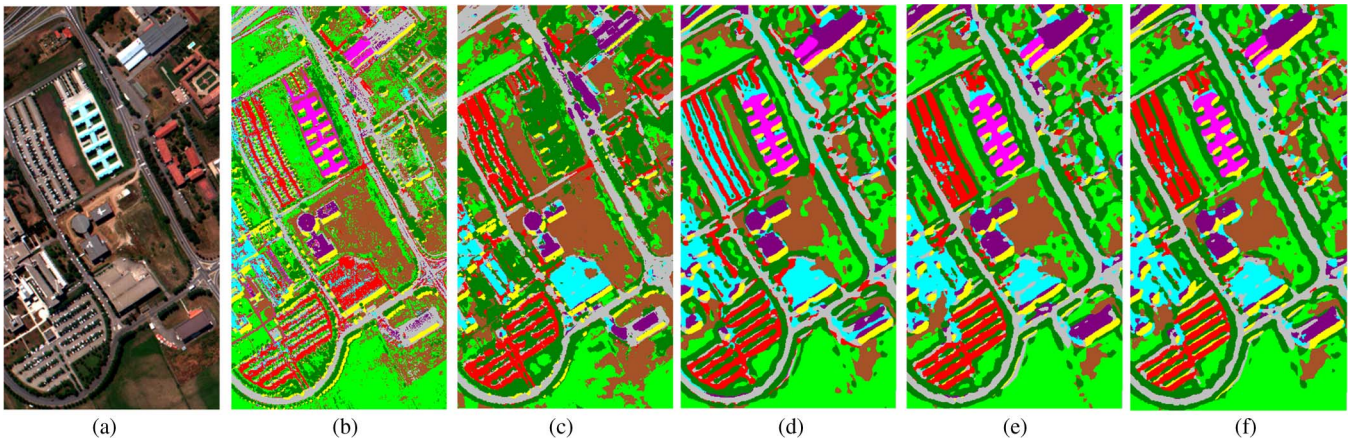


Fig. 21. University of Pavia. (a) False color image. (b)–(f) Classification maps for different classifiers: (b) 1D-SVM, (c) 3D-EMP-SVM, (d) 3D-CNN, (e) 3D-CNN with Method A, and (f) 3D-CNN with Method B.

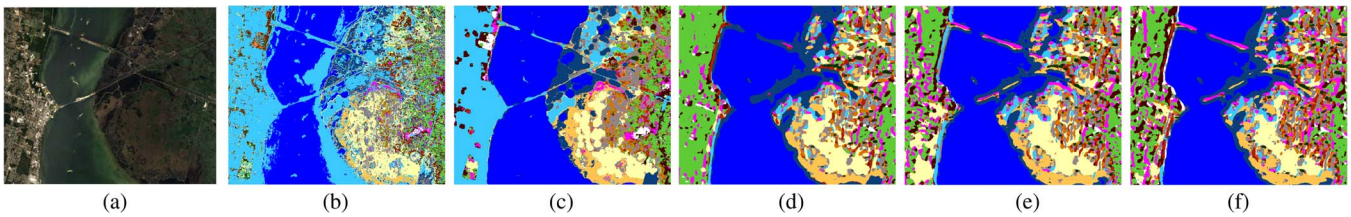


Fig. 22. KSC. (a) False color image. (b)–(f) Classification maps for different classifiers: (b) 1D-SVM, (c) 3D-EMP-SVM, (d) 3D-CNN, (e) 3D-CNN with Method A, and (f) 3D-CNN with Method B.

Under the condition of limited training samples, CNN with virtual samples outperformed EMP-based and original CNN methods in terms of OA, AA, and Kappa coefficient. This proves that CNN with virtual samples is a powerful tool for HSI classification.

In Tables XVIII–XX, in comparison with the original CNN, the OA improved by 0.97%, 0.12%, and 0.76% in the Indian Pines, University of Pavia, and KSC data sets, respectively. Moreover, the variances of OA are degraded too, which means that the CNNs with virtual samples are less influenced by different training samples.

It can be also found in experiments that CNN classifier will achieve a better performance in terms of classification accuracy if more virtual samples are created.

2) Classification Maps: At last, the classification accuracy values are examined to form a visual perspective. The 1D-SVM, 3D-EMP-SVM, 3D-CNN, and 3D-CNN with virtual samples are selected to classify the whole images. Figs. 20–22 are classification maps of different methods investigated in this paper for the three data sets. All parameters in these models are optimized. From the resulting images, we can figure out how the proposed FE method affects the classification results.

From Figs. 20–22, it is obvious that the spectral classification method (1D-SVM) always results in noisy scatter points in the images [see Figs. 20(a)–22(a)]. While the spectral–spatial methods correct this shortcoming, which eliminate noisy scattered points of misclassification. The CNN with virtual sample method gives more detailed classification maps [see Fig. 22(e) and (f)].

Obviously, both of the proposed virtual sample approaches can increase the classification accuracy of CNN significantly under insufficient training data.

VII. Discussion and Conclusion

In order to harvest the powerfulness of deep models for HSI FE and classification, in this paper, we have proposed deep CNN architectures to extract the spectral, spatial, and spectral-and-spatial-based deep features.

The design of proper deep CNN models is the first important issue we are facing. In the design of the spectral deep model, we use a small local reception field and three to five convolutional layers. For the spatial deep model, we use a small local reception field. For the spectral-and-spatial-based deep model, we use a special 3-D CNN model with a large reception field in the spectral domain and a small reception field in the spatial domain to extract the integrated features of HSI. The proper design will balance the capacity and complexity of the network, which is very important for further FE and classification.

In hyperspectral remote sensing cases, only limited training samples are available. To solve the problem of overfitting, we use L2 regularization for spectral CNN. When the input is a 3-D cube, overfitting becomes more serious. We then adopt a regularization entitled dropout. The proper regularization strategies play an important role for accurate classification of HSI.

Parameters affect the classification accuracy and computational complexity. In the realization of deep CNNs for HSI FE and classification, we gather some useful experience on parameter setting. The experimental results suggest that one or two layers often provide limited capacity in FE of HSI. Based on our experimental results, we suggest using a three-layer CNN with 4×4 or 5×5 convolution kernel and 2×2 pooling kernel in each layer for HSI FE.

By using proper architecture and powerful regularization, the proposed 3-D deep CNN has been demonstrated to provide excellent classification performance under the condition of limited training samples. The proposed deep model is promising with high potential, which opens a new window for further research.

In order to further improve the performance of CNN-based methods, a method entitled virtual sample is proposed. Virtual samples are generated by changing radiation and different mixture. Then, the training samples and the created virtual samples are used together in order to train a CNN.

In summary, to address the HSI FE and classification problem with limited training samples, we propose an idea of big network with strong constraints. The big feedforward DNN using deep 3-D CNN with virtual samples achieves by far the best results in terms of classification accuracy.

CNN is a hot topic in machine learning and computer vision. Various improvements have been made in recent years, and they can be also used in the proposed CNN architecture. The proposed model can be combined with post-classification processing to enhance mapping performance. It deserves to be investigated as a possible future work.

REFERENCES

[1] J. A. Benediktsson and P. Ghamisi, *Spectral–Spatial Classification of Hyperspectral Remote Sensing Images*. Boston, MA, USA: Artech House, 2015.

[2] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 1, pp. 55–63, Jan. 1968.

[3] J. B. Dias *et al.*, "Hyperspectral remote sensing data analysis and future challenges," *IEEE Geosci. Remote Sens. Mag.*, vol. 1, no. 2, pp. 6–36, Feb. 2013.

[4] X. Jia, B. Kuo, and M. M. Crawford, "Feature mining for hyperspectral image classification," *Proc. IEEE*, vol. 101, no. 3, pp. 676–679, Mar. 2013.

[5] G. Licciardi, P. R. Marpu, J. Chanussot, and J. A. Benediktsson, "Linear versus nonlinear PCA for the classification of hyperspectral data based on the extended morphological profiles," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 3, pp. 447–451, May 2011.

[6] A. Villa, J. A. Benediktsson, J. Chanussot, and C. Jutten, "Hyperspectral image classification with independent component discriminant analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 12, pp. 4865–4876, Dec. 2011.

[7] T. V. Bandos, L. Bruzzone, and G. Camps-Valls, "Classification of hyperspectral images with regularized linear discriminant analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 3, pp. 862–873, Mar. 2009.

[8] L. M. Bruce, C. H. Koger, and J. Li, "Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 10, pp. 2331–2338, Oct. 2002.

[9] L. O. Jimenez and D. A. Landgrebe, "Hyperspectral data analysis and supervised feature reduction via projection pursuit," *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 6, pp. 2653–2667, Nov. 1999.

[10] D. Lunga, S. Prasad, M. M. Crawford, and O. Ersoy, "Manifold-learning-based feature extraction for classification of hyperspectral data: A review of advances in manifold learning," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 55–66, Jan. 2014.

[11] T. Han, and D. Goodenough, "Investigation of nonlinearity in hyperspectral imagery using surrogate data methods," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 10, pp. 2840–2847, Oct. 2008.

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

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

[14] C. M. Bachmann, T. L. Ainsworth, and R. A. Fusina, "Improved manifold coordinate representations of large-scale hyperspectral scenes," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 10, pp. 2786–2803, Oct. 2006.

[15] B. Scholkopf and A. J. Smola, *Learning With Kernels*. Cambridge, MA, USA: MIT Press, 2002.

[16] B. C. Kuo, C. H. Li, and J. M. Yang, "Kernel nonparametric weighted feature extraction for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 4, pp. 1139–1155, Apr. 2009.

[17] A. Plaza, J. Plaza, and G. Martin, "Incorporation of spatial constraints into spectral mixture analysis of remotely sensed hyperspectral data," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, Grenoble, France, 2009, pp. 1–6.

[18] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Advances in spectral–spatial classification of hyperspectral images," *Proc. IEEE*, vol. 101, no. 3, pp. 652–675, Mar. 2013.

[19] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson, "SVM- and MRF-based method for accurate classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 4, pp. 736–740, Oct. 2010.

[20] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. 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.

[21] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral–spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 2, pp. 844–856, Feb. 2013.

[22] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3973–3985, Oct. 2011.

[23] B. Song, J. Li, J. M. Bioucas-Dias, and J. A. Benediktsson, "Remotely sensed image classification using sparse representations of morphological attribute profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 8, pp. 5122–5136, Aug. 2013.

[24] Y. Bengio, A. Courville, and P. Vincent, "Representation learning. A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[25] N. Kruger *et al.*, "Deep hierarchies in primate visual cortex what can we learn for computer vision?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1847–1871, Aug. 2013.

[26] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 1106–1114.

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

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[29] Y. Chen, Z. Lin, X. Zhao, and G. Wang, "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.

[30] Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 1–12, Jun. 2015.
[31] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1349–1362, Mar. 2016.
[32] Y. LeCun, C. Cortes, and C. Burges, The MNIST Database of Handwritten Digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[33] J. Deng and F. Li, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Miami, FL, USA, 2009, pp. 248–255.
[34] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
[35] N. LeRoux and Y. Bengio, "Deep belief networks are compact universal approximators," *Neural Comput.*, vol. 22, no. 8, pp. 2192–2207, Aug. 2010.
[36] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
[37] Z. Zuo *et al.*, "Learning contextual dependence with convolutional hierarchical recurrent neural networks," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 2983–2996, Jul. 2016.
[38] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE CVPR*, Boston, MA, USA, 2015, pp. 1–9.
[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.
[40] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE CVPR*, Columbus, OH, USA, 2014, pp. 581–587.
[41] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for non-orthogonal problems," *Technimetrics*, vol. 12, no. 1, pp. 55–67, Jan. 1970.
[42] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 807–814.
[43] G. E. Hinton *et al.*, "Improving neural networks by preventing co-adaptation of feature detectors," *Comput. Sci.*, vol. 3, no. 4, pp. 212–223, 2012.
[44] R. E. Edwards, H. Zhang, and L. E. Parker, "Approximate l-fold cross-validation with least squares SVM and kernel ridge regression," in *Proc. 12th ICMLA*, Miami, FL, USA, Dec. 2013, pp. 58–64.
[45] W. Hofmann, "Remote sensing: The quantitative approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 3, no. 6, pp. 713–714, Jun. 1981.
[46] D. J. Bartholomew, F. Steele, J. Galbraith, and I. Moustaki, "Analysis of multivariate social science data," *Struct. Equation Model. Multidisciplinary J.*, vol. 18, no. 4, pp. 686–693, Apr. 2011.
[47] H. Yang, F. Qin, and Y. Wang, "LLE-PLS nonlinear modeling method for near infrared spectroscopy and its application," *Spectrosc. Spectral Anal.*, vol. 27, no. 10, pp. 1955–1958, Oct. 2007.
[48] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Mar. 2011.
[49] L. G. Chova, D. Tuia, G. Moser, and G. C. Valls, "Multimodal classification of remote sensing images: A review and future directions," *Proc. IEEE*, vol. 103, no. 9, pp. 1560–1584, Nov. 2015.
[50] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectral–spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2438–2442, Dec. 2015.
[51] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, "Classification of hyperspectral data from urban areas based on extended morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 480–491, Mar. 2005.

Yushi Chen (M'11) received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2008.

Currently, he is an Associate Professor with the School of Electronics and Information Engineering, Harbin Institute of Technology. His research interests include remote sensing data processing and machine learning.

Hanlu Jiang received the Bachelor's degree in remote sensing science and technology in 2014 from the Harbin Institute of Technology, Harbin, China, where she is currently working toward the Master's degree in the School of Electronics and Information Engineering.

Her research area is in remote sensing image processing technologies.

Chunyang Li has been working toward the Master's degree in the Department of Information Engineering, School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin, China, since 2015.

Her research concerns remote sensing image processing based on deep learning methods.

Xiuping Jia (SM'03) received the B.Eng. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 1982 and the Ph.D. degree in electrical engineering from The University of New South Wales, Sydney, Australia, in 1996.

Since 1988, she has been with the School of Engineering and Information Technology, The University of New South Wales, Canberra, Australia, where she is currently a Senior Lecturer. She is also a Guest Professor with Harbin Engineering University, Harbin, China, and an Adjunct Researcher with the National Engineering Research Center for Information Technology in Agriculture, Beijing. She is the coauthor of the remote sensing textbook titled *Remote Sensing Digital Image Analysis* [Springer-Verlag, 3rd ed. (1999) and 4th ed. (2006)]. Her research interests include remote sensing and image data analysis.

Dr. Jia served as the inaugural Chair of the IEEE Australia Capital Territory and New South Wales Section GRSS Chapter from 2010 to 2013. She is an Associate Editor of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.

Pedram Ghamisi (S'12–M'15) received the B.Sc. degree in civil (survey) engineering from the Islamic Azad University, South Tehran Branch, Tehran, Iran; the M.Sc. degree (with first class honors) in remote sensing from Khajeh Nasir Toosi University of Technology, Tehran, in 2012; and the Ph.D. degree in electrical and computer engineering from the University of Iceland, Reykjavik, Iceland, in 2015.

He was a Postdoctoral Research Fellow with the University of Iceland. Since October 2015, he has been a Postdoctoral Research Fellow with Signal Processing in Earth Observation, Technical University of Munich, Munich, Germany, and a Researcher with the Remote Sensing Technology Institute (IMF), German Aerospace Center (DLR), Weßling, Germany. His research interests are in remote sensing and image analysis with special focus on spectral and spatial techniques for hyperspectral image classification and the integration of LiDAR and hyperspectral data for land-cover assessment.

In 2015, Dr. Ghamisi won the prestigious Alexander von Humboldt Fellowship.