

Research Article

FMRSS Net: Fast Matrix Representation-Based Spectral-Spatial Feature Learning Convolutional Neural Network for Hyperspectral Image Classification

Feifei Hou, Wentai Lei , Hong Li, and Jingchun Xi

School of Information Science and Engineering, Central South University, Changsha 410083, China

Correspondence should be addressed to Wentai Lei; leiwentai@csu.edu.cn

Received 6 February 2018; Revised 6 May 2018; Accepted 20 May 2018; Published 21 June 2018

Academic Editor: Paolo Addesso

Copyright © 2018 Feifei Hou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Convolutional Neural Network- (CNN-) based land cover classification algorithms have recently been applied in hyperspectral images (HSI) field. However, the large-scale training parameters bring huge computation burden to CNN and the spatial variability of spectral signatures leads to relative low classification accuracy. In this paper, we propose a CNN-based classification framework that extracts square matrix representation-based spectral-spatial features and performs land cover classification. Numerical results on popular datasets show that our framework outperforms sparsity-based approaches like basic thresholding classifier-weighted least squares (BTC-WLS) and other deep learning-based methods in terms of both classification accuracy and computational cost.

1. Introduction

Different from traditional images (e.g., RGB image), hyperspectral image (HSI) with hundreds of spectral channels has been widely applied in remote sensing [1]. Land cover classification is an important way to extract useful information from the HSI [2–4] where the task is to predict the type of land cover presented at the location of each pixel. There are several challenges associated with the predictive analysis of HSI classification: (1) huge computation resulting from large-scale training parameters and (2) large spatial variability of spectral signature.

Recently, supervised classification is probably the most active research area in hyperspectral data analysis. There is a vast literature on this topic using supervised machine-learning models, such as decision trees [5], random forests [6], and support vector machines (SVM) [7–9]. A random forest [10] is an ensemble learning approach that operates by constructing several decision trees in the training course and outputting the classes of the input hyperspectral pixels via integration of predictions of the individual trees. The SVM-based methods [8, 11], in general, follow a two-step approach. Firstly, complex handcrafted features are computed

from the raw data input, and secondly, the obtained features are used to learn classifiers. However, these approaches are treated as “shallow” models, and invariance and abstractness of the extracted features are limited compared to the “deep” ones. It is believed that, compared to the “shallow” models, deep learning architectures are able to extract high-level, hierarchical, and abstract features, which are generally more robust to the nonlinear processing.

Convolutional Neural Network (CNN) is regarded as an important branch of the deep learning family and especially good at image classification [12]. If designed properly, a CNN provides a hierarchical description of the input data in terms of relevant and easily interpreted features at every layer in image categorization tasks. For example, W. Hu et al. [13] trained a simple one-dimensional (1D) five-layer CNN that directly classifies hyperspectral images in spectral domain. D. Guidici et al. [14] attempted to carry out 1D CNN for classifying land cover from multi-seasonal hyperspectral imagery, followed by the features extracted from the spectral domain through the training of the network. To avoid overfitting, S. Mei et al. [15] suggested a spectral-spatial-feature-based classification framework, which jointly makes use of batch normalization, dropout, and parametric rectified linear

unit activation function and a 1D CNN. However, above-mentioned algorithms are all based on inputting 1D vector column corresponding to every pixel into CNN framework, followed by huge parameters when one needs to deal with hyperspectral data, which faces the difficulties of complex computation burden and much information redundancies to training framework.

To address the issue of imbalance between classification accuracy and computation, this paper establishes the framework of Fast Matrix Representation-Based Spectral-Spatial CNN (FMRSS Net), which uses a matrix representation of every pixel as input feature fed to the proposed deep model using format conversion. This approach reduces large-scale training parameters compared to that of vector column to decrease the computation burden. In addition, we also explore the spatial context in spectral domain to reduce the disturbance of interclass spectral variation. Furthermore, the learned feature can be transferred to different data or tasks [16].

The rest of this paper is organized as follows. An introduction to the existing methods is briefly given in Section 2. The details of the proposed FMRSS Net are described in Section 3. The datasets description, network analysis, experimental results, and a comparison with state-of-the-art algorithms are provided in Section 4. Finally, Section 5 concludes this paper.

2. Existing Methods

In this section, two kinds of state-of-the-art HSI classification methods were described in this work: one is the deep learning-based method and the other is the sparsity-based method.

2.1. HSI Classification via Simple CNN. At a broad level, a CNN is a deep-network topology that typically combines convolutional filter layers in conjunction with a classification network, which for this work is a fully connected neural network (NN). Through the standard back-propagation training process, convolutional filters are trained to capture salient structural features from the sequential input data. W. Hu et al. [13] mention these structural features as the “intra-class appearance and shape variation” within spectra and apply to HSI classification in the first time. The architecture of their proposed classifier contains five layers with weights, including the input layer, the convolutional layer, the max pooling layer, the full connection layer, and the output layer. They utilized single-season hyperspectral data and simple 1D CNN across the full spectral dimension to classify land cover with 90 to 93% overall accuracy, and CNN outperformed SVM by 1 to 3%. There are some drawbacks of this strategy. First, the proposed CNN is employed to classify HSI only in spectral domain which ignores the spatial information and leads to low accuracy. Second, the number of training parameters is large resulting in huge computation burden.

2.2. HSI Classification via BTC-WLS. The basic thresholding classifier (BTC) is a lightweight sparsity-based algorithm for HSI classification proposed by M. A. Toksöz et al. [17]. BTC is derived from the basic thresholding algorithm which could

be considered as one of the simplest techniques in compressed sensing theory [18, 19]. BTC is a pixelwise classifier which uses only the spectral features of a given test pixel. It performs the classification using a predetermined dictionary consisting of labeled training pixels. It then produces the class label and residual vector of the test pixel. In addition, their proposal is extended to a three-step spectral-spatial framework to improve classification accuracy. First, every pixel of a given HSI is classified using BTC. The resulting residual vectors form a cube which could be interpreted as a stack of images representing residual maps. Second, each residual map is filtered using an averaging filter. Finally, the class label of each test pixel is determined based on minimal residual. For the spectral-spatial proposal, BTC is also applied to the same filtering techniques in order to smooth the resulting residual maps and the version of it is called BTC-WLS (based on weighted least squares filtering [20]). The reason that this proposal includes the WLS filter is that it does not cause halo artifacts at the edges of an image as the degree of smoothing increases. The proposal outperforms well-known SVM-based methods and sparsity-based greedy approaches like simultaneous orthogonal matching pursuit in terms of both classification accuracy and computational cost. In the spectral-spatial case, although the BTC-WLS algorithm achieves the best results in terms of all metrics, it cannot distinguish between small targets in hyperspectral image scene for generally lacking training samples of desired class.

2.3. HSI Classification via SAE-LR. Chen et al. [21] employed deep learning method to handle HSI classification for the first time, where a stacked autoencoder (SAE) followed by logistic regression (LR) was adopted to extract the deep features in HSI. This paper optimizes using the mini-batch stochastic gradient descent method to derive the partial differentials of cost function. Then, the weight updating rule can be redefined. Both a representative spectral pixel vector and the corresponding spatial vector obtained from applying principle component analysis (PCA) to hyperspectral data over the spectral dimension are acquired separately from a local region and then jointly used as an input to the SAE. While SAE can extract deep features hierarchically in a layer-wise training fashion, the training samples composed of image patches have to be flattened to one dimension in order to meet the input requirement of such models. Unfortunately, SAE are unsupervised and do not directly make use of the label information when learning the features.

3. HSI Classification via Proposed FMRSS Net

As compared with the simple CNN and BTC-WLS algorithms, we can expect our algorithm to exhibit the following advantages:

- (1) It provides high classification accuracy.
- (2) It reduces the large number of training parameters and decreases computational cost.
- (3) It enables us to incorporate spatial information.

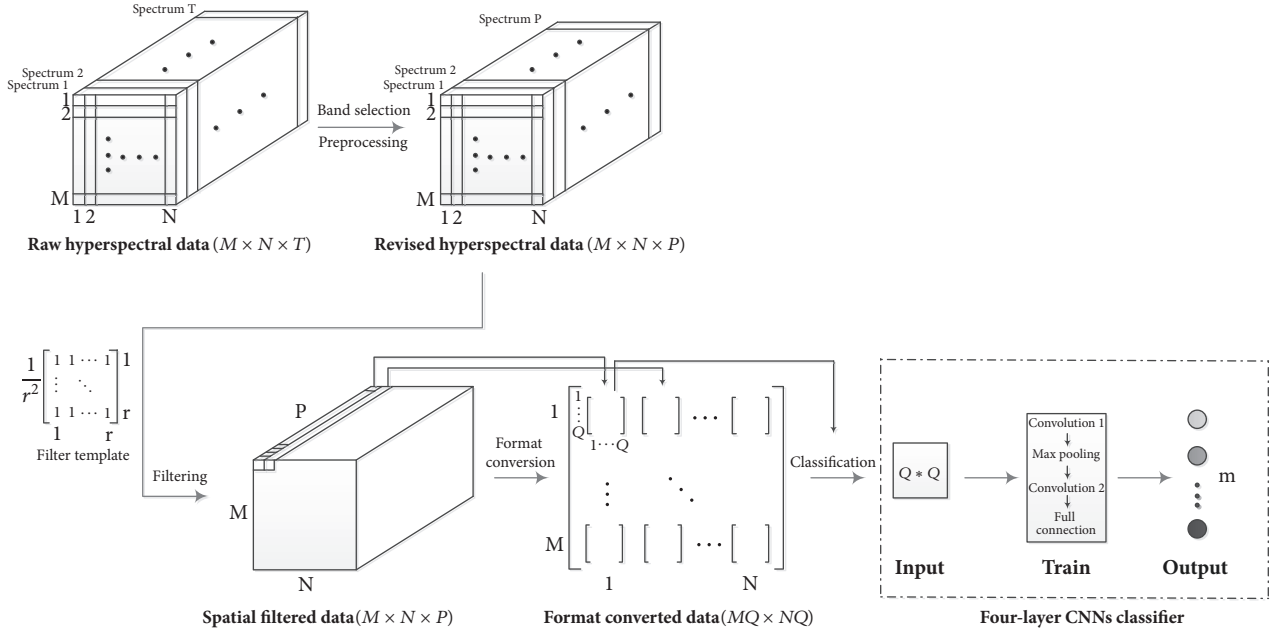


FIGURE 1: Flow chart of the proposed framework.

In this context, we propose the FMRSS Net framework for HSI classification which satisfies these properties and consists of four steps, as described briefly below.

- (1) HSI preprocessing: this step carries out band screening, normalization band sorting, and extraction.
- (2) Joint spatial-spectral information processing: mean filter is applied to each band data and the spectral information per individual pixel is correlated with that of adjacent pixels.
- (3) Format conversion: in this step, a one-dimensional (1D) vector column is converted to a square matrix representation, which reduces the number of training parameters of CNN.
- (4) Classification by using a four-layer CNN: this classifier includes four learning layers: convolution layer 1, max pooling layer, convolution layer 2, and fully connected layer.

The flow chart of our FMRSS Net framework is shown in Figure 1.

3.1. Preprocessing for HSI. Suppose the original HSI includes T frequency bands, and each band corresponds to a two-dimensional (2D) image with size $M \times N$. The noise bands and water absorption bands need to be removed from the T original bands in the beginning, resulting in U remaining bands.

After normalizing to $[-1, 1]$, the data of the remained U bands are sorted by the energy parameter of the 2D image corresponding to each band. In order to extract informative

bands, the top P bands with bigger energy values are extracted from U bands by using the following formula:

$$P = \lfloor \sqrt{U} \rfloor^2 \quad (1)$$

where $\lfloor \cdot \rfloor$ indicates rounding to the nearest integer towards minus infinity.

3.2. Joint Spectral-Spatial Processing. This processing requires a mean filter which is applied to each 2D image of P bands. The selection of filter template is related to the image size as well as the number of samples for every land cover type. For example, in the Indian Pines dataset, some classes have very few samples. Among these classes, if the filter template is too small, the spectral information of the neighborhood cannot be fully exploited. On the other hand, if bigger template is used, the image blur will be enlarged. Therefore, choosing a suitable filter template is of great importance. Here, we design the size of filter template to adapt experimental datasets.

For P bands of hyperspectral data, the collected data for each band is a 2D image with size $M \times N$. Let $(1/r^2)Br$ denote a filter template per band, where Br is a matrix with size $r \times r$ and every value in the matrix is 1. For different datasets, the values M and N represent the length and width of the 2D image corresponding to each band, and r is set as follows:

$$r = \begin{cases} 7, & (M > 300) \text{ and } (N > 300) \\ 5, & [(M \leq 300) \text{ or } (N \leq 300)] \text{ and } (\exists S(i) < 200) \\ 3, & [(M \leq 300) \text{ or } (N \leq 300)] \text{ and } (\forall S(i) \geq 200) \end{cases} \quad (2)$$

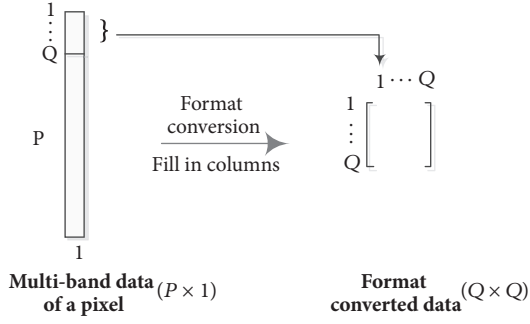


FIGURE 2: Format conversion of a pixel.

where $S(\cdot)$ indicates the number of samples for each class, $i = 1, 2, \dots, m$, and m is the number of land cover types.

3.3. Format Conversion. According to the parameter settings in following part, the number of training parameters is related to the input data size, kernel size, and feature map size. The input data with 1D vector column results in larger kernel and feature map which brings huge computational cost to CNN. We make a contribution to reduce the number of training parameters before the input data is fed into network, which is the format conversion using 2D matrix representation feature. Multiband data of each pixel perform format conversion, as shown in Figure 2. Format conversion, which represents 1D array ($P \times 1$) corresponding to a single pixel, is converted to the 2D square ($Q \times Q$) filled by column, the data of 1D array from 1 to Q is placed in first column of 2D square matrix, then the next data from $Q + 1$ to $2Q$ is placed in second column, and so on. From (1), it can be inferred that $Q = \lfloor \sqrt{U} \rfloor = \sqrt{P}$, so the 1D array of $P \times 1$ just fills in the square matrix of $Q \times Q$.

3.4. Architecture of the Proposed CNN. A four-learning-layer based CNN classifier is proposed to extract features and classify hyperspectral data, as shown in Figure 3.

The total spectral data to be processed has a total of $M \times N$ pixel vectors, which are divided into training samples and testing samples according to the proportion. To make more concrete, we randomly select the number of training samples according to the batch size (which is the number of training samples per training session) fed into the network and get parameters of the classifier by training these samples. Next, we could observe that the classification results are obtained by inputting the testing samples into the trained network. The parameter settings for each layer are given in Table 1.

4. Experimental Results and Discussion

All programs are implemented using the deep learning toolbox [22] based on MATLAB R2016a language, and the toolbox offers some deep learning templates that allow researchers in remote sensing to solve the issue of large-scale image classification. The experimental results are generated on a PC equipped with an Intel Core i7-7700 with 3.6GHz and Nvidia GTX 1060 6G graphics card.

4.1. Experimental Datasets Description and Parameter Settings. The experiments are performed on two popular HSI classification datasets: Indian Pines [23] and Salinas (http://www.ehu.es/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes). Both datasets were captured by an airborne visible/infrared imaging spectrometer (AVIRIS) sensor in wavelength range $0.4 \sim 2.5 \mu\text{m}$.

4.1.1. Indian Pines Dataset. The classification methods are firstly applied to the Indian Pines dataset. Only 200 ($U = 200$) bands are adopted after the removal of the spectral bands affected by atmospheric absorption. By sorting through the energy values, we finally select the 196 bands with larger energy for evaluation where Q takes 14. The dataset contains 145×145 ($M = N = 145$) pixels with a ground resolution of 20 m. Since the number of samples in the dataset is not evenly distributed based on the 16 land cover types ($m = 16$), we randomly select 10% of the land cover pixels as training samples and the remaining 90% as testing samples and set the minimum number of training samples to 10 for each class. Table 2 shows each class of experimental land cover information and the corresponding number of training and testing samples; we get a total of 10198 samples with 1000 training samples and 9198 testing samples. In addition, each pixel must be scaled to $[-1, 1]$. The layer parameters of this dataset in the proposed four-layer CNN classifier are given in Parameter Settings for Indian Pines/Salinas (Table 3).

4.1.2. Salinas Dataset. For the Salinas dataset, before the experiments, 20 noisy water absorption bands were discarded and only 204 ($U = 204$) bands remained for evaluation. By sorting through the energy values, we discard some bands and finally select the 196 bands with larger energy for evaluation where Q takes 14. The dataset contains 512×217 ($M = 512$ and $N = 217$) pixels with a ground resolution of 3.7 m and a total of 16 land cover types. We randomly pick up half of the all ground-truth samples for experiment since the number of all samples is so large to increase computation burden, then we randomly select 10% of them for training samples and the remaining 90% for testing sample s and set the minimum number of training samples to 10 for each class. Table 4 shows each class of the experimental land cover and the corresponding numbers of training and testing samples. A total of 2700 training samples and 24349 testing samples are selected as the original data for the experiment. In addition, each pixel is normalized to $[-1, 1]$. The layer parameters of this dataset in the proposed classifier are also given in Parameter Settings for Indian Pines/Salinas.

4.2. Comparison with Other Classification Algorithms. The test accuracies of the FMRSS Net framework (FMRSS) for the Indian Pines and Salinas datasets are compared with sparsity-based and deep learning-based algorithms in Table 5. All the classifiers are trained on the same train set and tested on the same test set for a fair comparison. According to (2), the filter template with size 5×5 and 3×3 are chosen for the Indian Pines and Salinas datasets. Among sparsity-based algorithms, SAE [22], SAE-LR [21], and BTC-WLS [17] are compared with our framework. A network structure of [196-100-16] is

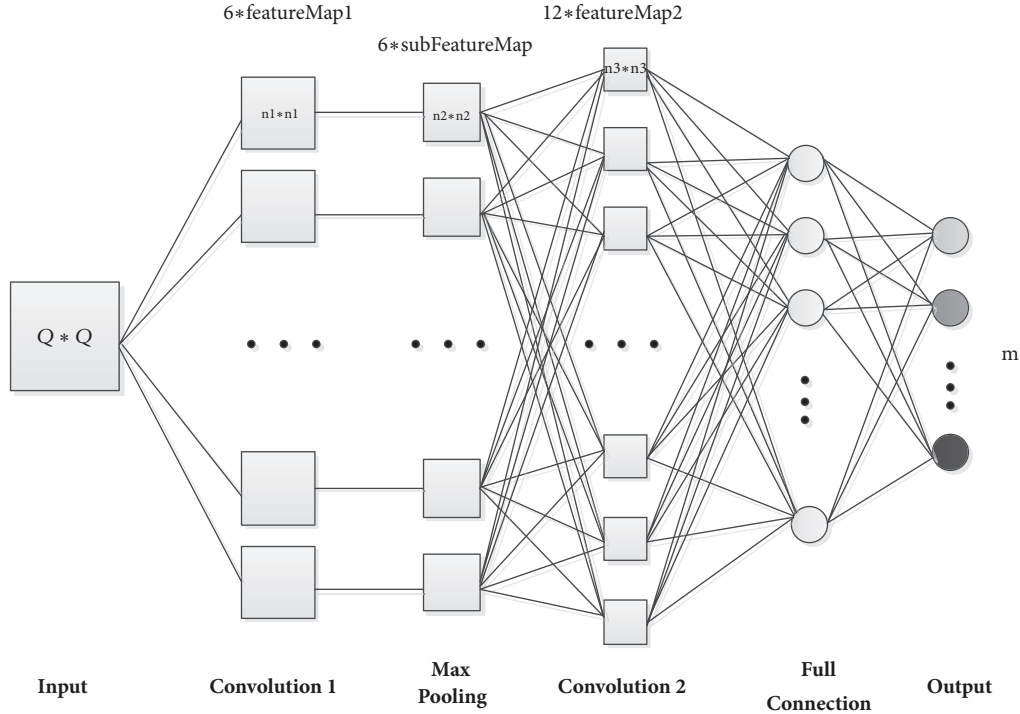


FIGURE 3: The proposed four-layer CNN classifier.

TABLE 1

Parameter Settings

INPUT:

(i) Input data: $Q \times Q$

CONVOLUTION 1:

- (i) 6 feature maps
- (ii) Kernel size: $k_1 \times k_1$, where $k_1 = \lceil Q/3 \rceil$, $\lceil \cdot \rceil$ rounds the elements to the nearest integers towards infinity.
- (iii) Feature map size: $n_1 \times n_1$ ($n_1 = Q - k_1 + 1$)
- (iv) Training parameters number: $(k_1 \times k_1 + 1) \times 6$
- (v) Activation function: Sigmoid

MAX POOLING:

- (i) 6 feature maps
- (ii) Kernel size: $k_2 \times k_2$, k_2 is set between 2 ~ 5
- (iii) Feature map size: $n_2 \times n_2$ ($n_2 = n_1/k_2$)
- (iv) Training parameters number: $6 \times k_2$
- (v) Pooling function: Max pooling sampling

CONVOLUTION 2:

- (i) 12 feature maps
- (ii) Kernel size: $k_3 = \lfloor n_2/2 \rfloor$.
- (iii) Feature map size: $n_3 \times n_3$ ($n_3 = n_2 - k_3 + 1$)
- (iv) Training parameters number: $6 \times k_3 \times k_3 \times 12 + 12$
- (v) Activation function: Sigmoid

















FULL CONNECTION:

- (i) Units number: $n_4 = 100$
- (ii) Training parameters number: $n_4 \times (12 \times n_3 \times n_3 + 1)$
- (iii) Activation function: Softmax

OUTPUT:

- (i) Units number m , also named classes number
- (ii) Training parameters number: $m \times (n_4 + 1)$

TABLE 2: Each class and the corresponding number of training and testing samples of Indian Pines dataset.

Color	No.	Class	Train Samples	Test Samples
	1	Alfalfa	10	36
	2	Corn-notill	143	1285
	3	Corn-mintill	83	747
	4	Corn	24	213
	5	Grass-pasture	49	434
	6	Grass-trees	73	657
	7	Grass-pasture-mowed	10	18
	8	Hay-windrowed	48	430
	9	Oats	10	10
	10	Soybean-notill	98	874
	11	Soybean-mintill	195	2209
	12	Soybean-clean	60	533
	13	Wheat	21	184
	14	Woods	127	1138
	15	Buildings-Grass-Trees-Drives	39	347
	16	Stone-Steel-Towers	10	83
	Sum		1000	9198

to carry out 500 times unsupervised training on each SAE network and 10,000 times supervised training on the entire classification network with the sigmoid as activation function [22]. Note that because deep learning-based methods may perform poor when training samples are not enough, in the comparison experiments the number of training samples in SAE-LR is set as 60% of all labeled pixels for Indian Pines dataset, while, for the Salinas dataset, we randomly pick up half of the all ground-truth samples for experiment, then we randomly select 60% of them for training samples, 20% for validation samples, and the remaining 20% for testing samples. The neural networks are constructed as [196-60-60-60-60-60-16] for the Indian Pines dataset and [196-30-30-30-30-30-16] for the Salinas dataset. The experiments considering joint spectral-spatial information are carried out with 3 principal components and 5000 epochs of pretraining and 100,000 epochs of fine-tuning [21]. Parameters of the learning algorithm of the BTC-WLS as well as the number of output classes are set equal to the values mentioned in [17] with the only exception that testing set size for Salinas is set to half of the original paper. Among deep learning-based classifiers, the simple 1D CNN architecture is implemented with the same architecture and hyperparameter values as mentioned in [13]. The proposed FMRSS and 1D CNN both are trained 10,000 times with sigmoid as activation function. Obviously, compared with several other algorithms, our proposal achieves a better performance on the two datasets in terms of overall accuracy (OA) of classification. In

addition, classification maps resulting from the Indian Pines and Salinas scenes using our framework are presented in Figures 4 and 5.

We present the confusion matrix of our framework for the Indian Pines in Table 6 where the index number (1, 2, . . . 16) represents the corresponding class in Table 2. The cell in the i th row and j th column means the percentage of the i th class samples (according to ground truth) which is classified to the j th class. For example, 99.77% of class 2 (Corn-notill) samples are classified correctly, but 0.16% of class 2 (Corn-notill) samples are wrongly classified to class 3 (Fallow). The percentages on diagonal line are the classification accuracies of corresponding classes. Furthermore, the performance verifies that the proposed framework has discriminative capability to extract subtle visual features, which is even superior to human vision for classifying complex curve shapes.

Compared with BTC-WLS algorithm, the proposed framework has higher classification accuracy not only for the overall dataset but also for certain specific classes (classes 2, 3, 5, 10, 11, and 15) on Indian Pines dataset, as shown in Figure 6. However, Figure 6 only shows accuracy in percent and ignores the number of testing samples. For clearer illustration of the advantage of the proposed algorithm, Figure 7 presents the number comparison of misclassification samples by FMRSS compared with BTC-WLS. Statistical results indicate that the total number of misclassified samples acquired by FMRSS (which is 208) is far less than BTC-WLS (which is 332).

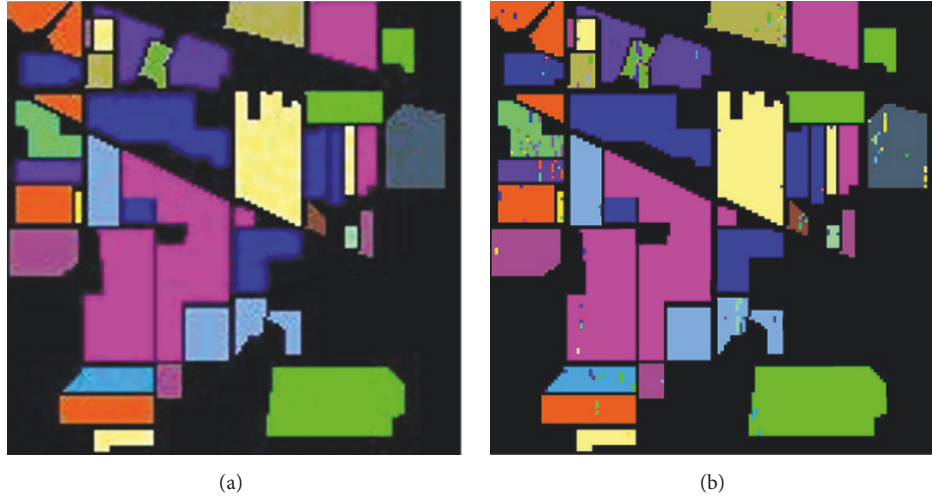


FIGURE 4: Classification maps on the Indian Pines dataset with 16 classes. (a) Ground truth; (b) our FMRSS Net.

TABLE 3

Parameter Settings for Indian Pines/Salinas
INPUT:
(i) Input data: 14×14 ($Q = 14$)
(ii) Learning rate: 1
(iii) Batch size: 100
CONVOLUTION 1:
(i) 6 feature maps
(ii) Kernel size: 5×5 ($k_1 = 5$)
(iii) Feature map size: 10×10 ($n_1 = 10$)
(iv) Training parameters number: 156
(v) Activation function: Sigmoid
MAX POOLING:
(i) 6 feature maps
(ii) Kernel size: $k_2 = 2$
(iii) Feature map size: 5×5 ($n_2 = 5$)
(iv) Training parameters number: 12
(v) Pooling function: Max pooling sampling
CONVOLUTION 2:
(i) 12 feature maps
(ii) Kernel size: $k_3 = 2$
(iii) Feature map size: 4×4 ($n_3 = 4$)
(iv) Training parameters number: 300
(v) Activation function: Sigmoid
FULL CONNECTION:
(i) Units number: $n_4 = 100$
(ii) Training parameters number: 19300
(iii) Activation function: Softmax
OUTPUT:
(i) Units number $m = 16$, named classes number
(ii) Training parameters number: 1616

4.3. *Comparison of Different Parameter Settings.* Table 7 shows the classification results under different filter templates

on two datasets. The results show that the filter templates of 5×5 and 3×3 are appropriate for Indian Pines and Salinas datasets, respectively. For the Indian Pines dataset, there exist some classes (1, 7, 9, 16) with low number of samples where the spectral information of neighborhood is not fully utilized when the filter template size is 3×3 , and the filter template with size 7×7 enlarges image blur, and the size of 5×5 is the most suitable filter template that utilizes spatial and spectral information properly. Based on the above analysis, this method can achieve the best classification accuracy of 97.74% and 99.29% on two datasets, respectively.

The depth of network is also an important factor affecting the network structure and determining the quality of the extracted data characteristics. Table 8 tests the effect of network depth on the classification results for different datasets. Two-learning-layer based network includes convolution layer and fully connected layer, while three-learning-layer based network includes convolution layer, max pooling layer, and fully connected layer. Four-learning-layer based network is the proposed classifier in this paper by referring to Section 3.4. The experiments show effectiveness of the proposed four-layer structure.

4.4. *Analysis of Computational Cost.* In addition, CNN share weights, which significantly decreases the number of parameters needed to be trained in comparison with other deep approaches. However, the number of parameters is still high when one needs to deal with hyperspectral data. The proposed format conversion addresses this issue which converts the 1D vector column into a matrix representation and thus reduces large-scale parameters and network complexity. The unified input bands and output classes are set to 196 and 16. From the ‘‘Parameter Settings for Indian Pines/Salinas’’, it can be obviously obtained that the total number of training parameters of the proposed framework is $156 + 12 + 300 + 19300 + 1616 = 21384$.

Different from the proposed FMRSS Net, each input data fed into CNN [13] is a 1D vector column. The input layer is

TABLE 4: Each class and the corresponding number of training and testing samples of Salinas dataset.

















Color	No.	Class	Train Samples	Test Samples
	1	Broccoli_green_weeds_1	102	912
	2	Broccoli_green_weeds_2	187	1674
	3	Fallow	100	898
	4	Fallow_rough_plow	71	635
	5	Fallow_smooth	135	1206
	6	Stubble	200	1792
	7	Celery	176	1583
	8	Grapes_untrained	552	5107
	9	Soil_vinyard_develop	302	2710
	10	Corn_senesced_green_weeds	163	1461
	11	Lettuce_romaine_4wk	55	488
	12	Lettuce_romaine_5wk	97	867
	13	Lettuce_romaine_6wk	45	405
	14	Lettuce_romaine_7wk	57	504
	15	Vinyard_untrained	365	3279
	16	Vinyard_vertical_trellis	93	828
	Sum		2700	24349

TABLE 5: The overall accuracy comparison with different methods on two datasets (in percent).

Algorithms	Datasets	
	Indian Pines	Salinas
SAE	74.95	85.80
CNN in [13]	90.44	92.75
SAE-LR	91.35	92.73
BTC-WLS	96.39	98.84
FMRSS Net	97.74	99.29

196×1 . The first hidden convolutional layer C1 filters the input data with 20 kernels of size 21×1 . Layer C1 contains $20 \times 176 \times 1$ nodes. The max pooling layer M2 is the second hidden layer, and the kernel size is 4×1 . Layer M2 contains $20 \times 40 \times 1$ nodes, and there is no parameter in this layer. The fully connected layer F3 has 100 nodes. The output layer has 16 nodes. Therefore, the total number of training parameters in [13] is $20 \times (21+1) + (20 \times 40 + 1) \times 100 + (100+1) \times 16 = 82156$.

Table 9 shows the comparison of classification and computational cost both on two datasets to certify effectiveness of format conversion for our frameworks: the framework with 1D vector column and 2D matrix representation input data. Compared with 1D vector column input data, 2D matrix representation reduces significantly 60772 parameters. In addition, compared to the 1D convolution kernel that only

extracts adjacent columns information of each pixel, the proposed CNN model employs the 2D convolution kernel which can utilize the neighborhood information of each pixel. In addition, the proposed CNN model with four learning layers can complete the layer-by-layer feature extraction of the image through multiple convolution layers. It can be obtained that the proposed framework can achieve an improved classification accuracy by 7.30% and 6.54% on Indian Pines and Salinas, respectively.

5. Conclusions

In this paper, a CNN-based classification framework has been proposed to directly address the problems of the training parameter burden and spatial variability of spectral signature pertaining to HSI classification. The matrix representation-based spectral-spatial feature learning and extensive parameter sharing in the neural network help achieve superior classification performance with fast speed than other popular methods on benchmark HSI classification datasets. Experiments results show that, compared with BTC-WLS algorithm, the proposed framework achieves a classification accuracy improvement of 1.35% on Indian Pines dataset and 0.45% on Salinas dataset. Likewise, compared with SAE-LR algorithm, this framework improves OA by 6.39% on Indian Pines dataset and 6.56% on Salinas dataset. Future work will focus on filtering parameters adaptive technique and semi-supervised algorithms combined with CNN.

TABLE 6: Confusion matrix of Indian Pines dataset (in percent).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	83.33	2.78	0	2.78	0	0	0	0	0	0	0	0	2.78	2.78	0	5.55
2	0	99.77	0.16	0.07	0	0	0	0	0	0	0	0	0	0	0	0
3	0.13	1.07	98.26	0.40	0	0	0	0	0	0	0.14	0	0	0	0	0
4	0.47	0	0.94	84.51	10.33	0	0	0	0.94	0.47	0	2.35	0	0	0	0
5	0	0	0	0	99.31	0.23	0	0	0.46	0	0	0	0	0	0	0
6	0	0	0	0	0.15	97.26	1.52	1.07	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	83.33	16.67	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0.70	1.16	94.88	2.09	0.47	0	0	0.23	0.23	0.23	0
9	0	0	0	10.0	0	0	0	10.0	80.0	0	0	0	0	0	0	0
10	0	0	0	0.11	0	0	0	0.23	0.11	99.31	0	0.24	0	0	0	0
11	0	0.18	0.05	0.09	0	0	0	0	0	0.09	99.59	0	0	0	0	0
12	0	0.75	2.06	0.75	0	0	0	0	0	0	0	95.87	0.57	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	2.72	92.39	4.89	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0.62	99.30	0.08	0
15	0	0	0	0	1.44	1.15	0.86	0.29	0.29	0	0	0	0	3.17	92.80	0
16	3.61	6.02	0	0	1.20	0	0	0	0	0	0	10.84	1.20	0	2.41	74.72

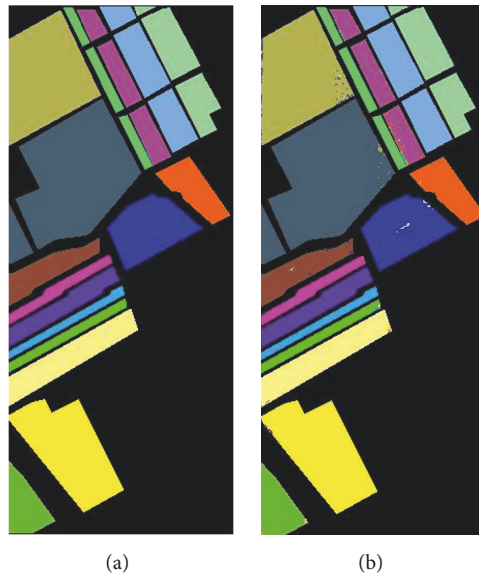


FIGURE 5: Classification maps on the Salinas dataset with 16 classes. (a) Ground truth; (b) our FMRSS Net.

TABLE 7: The classification results under different filter windows on two datasets (in percent).

Filter template	Datasets	
	Indian Pines	Salinas
3 × 3	96.28	99.29
5 × 5	97.74	99.15
7 × 7	96.31	99.03

TABLE 8: The classification accuracy under different depth of network on two datasets (in percent).

Num. of Layers	Datasets	
	Indian Pines	Salinas
2	70.27	83.95
3	92.44	96.15
4	97.74	99.29

TABLE 9: Computation cost with different neural networks on the two datasets.

Frameworks	Classification accuracy (%)		Parameters number	Processing time (s)	
	Indian Pines	Salinas		Indian Pines	Salinas
1D vector representation	90.44	92.75	82156	2700	9100
2D matrix representation	97.74	99.29	21384	900	3600

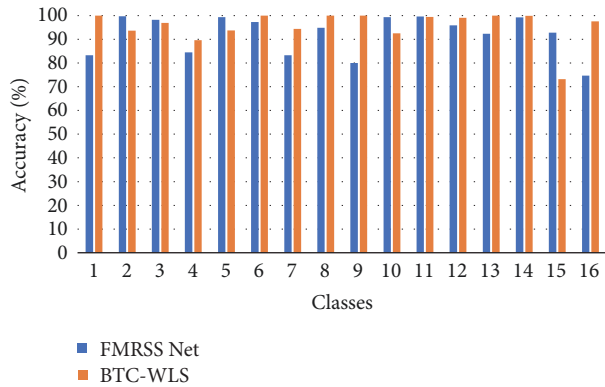


FIGURE 6: Classification accuracy comparison of each class on Indian Pines dataset.

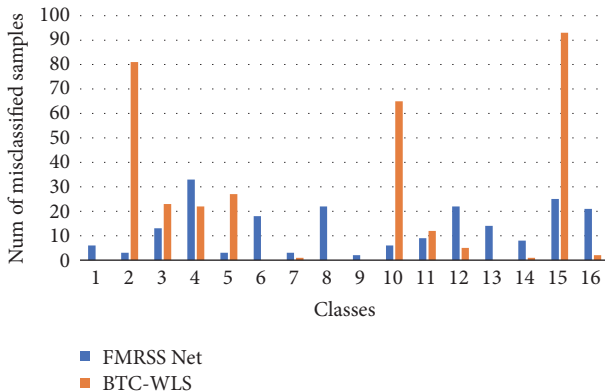


FIGURE 7: Number comparison of misclassified samples on Indian Pines dataset.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (nos. 61102139 and 61572524) and the Fundamental Research Funds for the Central Universities of Central South University (2018zzts181). The authors would like to thank Deliang Xiang at Beijing Academy of Military

Sciences (China) for greatly helping to improve the quality of this paper.

References

- [1] X. Zhu, D. Tuia, L. Mou et al., “deep learning in remote sensing,” *IEEE Geoscience and Remote Sensing magazine*, 2017.
- [2] Y. Chen, N. M. Nasrabadi, and T. D. Tran, “Hyperspectral image classification via kernel sparse representation,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 1, pp. 217–231, 2012.
- [3] W. Li, C. Chen, H. Su, and Q. Du, “Local Binary Patterns and Extreme Learning Machine for Hyperspectral Imagery Classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 7, pp. 3681–3693, 2015.
- [4] P. Wang, S. Xu, Y. Li, J. Wang, and S. Liu, “Hyperspectral image classification based on joint sparsity model with low-dimensional spectral–spatial features,” *Journal of Applied Remote Sensing*, vol. 11, no. 1, p. 015010, 2017.
- [5] S. Delalieux, B. Somers, B. Haest, T. Spanhove, J. Vanden Borre, and C. A. Múcher, “Heathland conservation status mapping through integration of hyperspectral mixture analysis and decision tree classifiers,” *Remote Sensing of Environment*, vol. 126, pp. 222–231, 2012.
- [6] J. Ham, Y. Chen, M. M. Crawford, and J. Ghosh, “Investigation of the random forest framework for classification of hyperspectral data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 3, pp. 492–501, 2005.
- [7] X. Kang, S. Li, and J. A. Benediktsson, “Spectral-spatial hyperspectral image classification with edge-preserving filtering,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 5, pp. 2666–2677, 2014.
- [8] F. Melgani and L. Bruzzone, “Classification of hyperspectral remote sensing images with support vector machines,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 8, pp. 1778–1790, 2004.
- [9] Y. Tarabalka, J. Chanussot, and J. A. Benediktsson, “Segmentation and classification of hyperspectral images using watershed transformation,” *Pattern Recognition*, vol. 43, no. 7, pp. 2367–2379, 2010.
- [10] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.
- [11] W. Li, E. W. Tramel, S. Prasad, and J. E. Fowler, “Nearest regularized subspace for hyperspectral classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 1, pp. 477–489, 2013.
- [12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

- [13] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep Convolutional Neural Networks for Hyperspectral Image Classification," *Journal of Sensors*, vol. 2015, Article ID 258619, 12 pages, 2015.
- [14] D. Guidici and M. L. Clark, "One-dimensional convolutional neural network land-cover classification of multi-seasonal hyperspectral imagery in the San Francisco Bay Area, California," *Remote Sensing*, vol. 9, no. 6, 2017.
- [15] S. Mei, J. Ji, Q. Bi, J. Hou, Q. Du, and W. Li, "Integrating spectral and spatial information into deep convolutional Neural Networks for hyperspectral classification," in *Proceedings of the 36th IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2016*, pp. 5067–5070, chn, July 2016.
- [16] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable Are Features in Deep Neural Networks?" *Advances in Neural Information Processing Systems*, vol. 27, pp. 3320–3328, 2014, <https://arxiv.org/abs/1411.1792>.
- [17] M. A. Toksöz and I. Ulusoy, "Hyperspectral image classification via basic thresholding classifier," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 7, pp. 4039–4051, 2016.
- [18] H. Rauhut, K. Schnass, and P. Vandergheynst, "Compressed sensing and redundant dictionaries," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 54, no. 5, pp. 2210–2219, 2008.
- [19] S. Foucart and H. Rauhut, "A Mathematical Introduction to Compressive Sensing," *A Mathematical Introduction to Compressive Sensing*, vol. 44, 2013.
- [20] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Transactions on Graphics*, vol. 27, no. 3, article 67, 2008.
- [21] Y. S. Chen, Z. H. Lin, X. Zhao, G. Wang, and Y. F. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.
- [22] R. B. Palm, "Prediction as a Candidate for Learning Deep Hierarchical Models of Data," Tech. Rep., Technical University of Denmark, 2012.
- [23] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3," 2015, <https://purr.purdue.edu/publications/1947/1>.

