Hindawi
Journal of Sensors
Volume 2018, Article ID 8602103, 13 pages
https://doi.org/10.1155/2018/8602103

Research Article

Convolution Neural Network Based on Two-Dimensional Spectrum for Hyperspectral Image Classification

Hongmin Gao,[1] **Shuo Lin**,[1] **Yao Yang**,[1] **Chenming Li**,[1] **and Mingxiang Yang**[2]

[1]*College of Computer and Information Engineering, Hohai University, Nanjing 211100, China*
[2]*State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, China Institute of Water Resources and Hydropower Research, Beijing 100038, China*

Correspondence should be addressed to Chenming Li; lichenming55@163.com

Received 11 May 2018; Accepted 24 July 2018; Published 28 August 2018

Academic Editor: Stelios M. Potirakis

Inherent spectral characteristics of hyperspectral image (HSI) data are determined and need to be deeply mined. A convolution neural network (CNN) model of two-dimensional spectrum (2D spectrum) is proposed based on the advantages of deep learning to extract feature and classify HSI. First of all, the traditional data processing methods which use small area pixel block or one-dimensional spectral vector as input unit bring many heterogeneous noises. The 2D-spectrum image method is proposed to solve the problem and make full use of spectral value and spatial information. Furthermore, a batch normalization algorithm (BN) is introduced to address internal covariate shifts caused by changes in the distribution of input data and expedite the training of the network. Finally, Softmax loss models are proposed to induce competition among the outputs and improve the performance of the CNN model. The HSI datasets of experiments include Indian Pines, Salinas, Kennedy Space Center (KSC), and Botswana. Experimental results show that the overall accuracies of the 2D-spectrum CNN model can reach 98.26%, 97.28%, 96.22%, and 93.64%. These results are higher than the accuracies of other traditional methods described in this paper. The proposed model can achieve high target classification accuracy and efficiency.

1. Introduction

Hyperspectral images (HSIs) are typically composed of hundreds of spectral data channels in the same scene. HSIs can provide continuous data in space and spectrum through combined imaging and spectrum technology. Hyperspectral data are important in monitoring information of the Earth's surface because the spectral information provided by the hyperspectral sensor increases the accuracy of the resolution of target materials and thus improves classification accuracy [1].

At first, scholars mainly use artificial extraction of image features for object identification classification of remote sensing images using a local binary pattern, histogram of oriented gradient [2], and Gabor filter [3]. However, this method is ineffective in processing hyperspectral data with the increase in dimension. Thus, feature extraction and classifier are combined, thereby yielding a satisfactory classification effect. Methods for feature extraction include principal component analysis (PCA) [4], independent component analysis (ICA) [5], and linear discriminant analysis (LDA) [6] and robust PCA. The classifier went through a process from fuzzy K-nearest neighbor algorithm [7], naive Bayes with deep feature weighting [8], and logistic regression [9] to support vector machine (SVM) [10]. SVM improves classification performance by extending classification kernel [11]. However, these combinatorial methods demonstrate the following significant limitations. (1) Feature extraction uses linear transformations to extract potentially useful features from the input data. Hyperspectral data are essentially nonlinear considering a complex light-scattering mechanism [12]. (2) Most traditional classification methods only consider single-layer processing, which reduces the capability of feature learning, and are unsuitable for high-dimensional data.

Neural networks (NNs) with multiple layers and hidden nodes are more suitable than shallow classifiers, such as

SVM, in building an HSI data model [13]. The NNs, including multilayer perceptron [14] and radial basis function [15], have been studied for classifying remote sensing data. Researchers have proposed a semisupervised NN framework for large-scale HSI classification [16]. Various deep NNs (DNN) have been developed according to system architecture and activation functions; these networks include deep belief network (DBN) [17], deep Boltzmann machine [18], and AutoEncoder (AE) [19]. In 2014, a stacked AE (SAE) was used for HSI classification [20]. An improved AE based on sparse constraints was then proposed [21]. The DBN is another DNN model that was proposed in 2015 [22]. The depth model can extract robust features and is superior to other methods in terms of classification accuracy.

Convolution NN (CNN) [23] uses local receptive fields in efficiently extracting spatial information and sharing weights to significantly reduce the number of parameters. CNNs are used to extract the spatial spectral features of hyperspectral images for classification [24], and their performance was better than that of traditional classifiers such as SVM. In addition, a method using a virtual sample enhanced to limited labeled samples was proposed in [25]. A previous study proposed the use of a greedy layer unsupervised pretraining to form a CNN model [26]. However, the application technology of CNN in hyperspectral classification remains imperfect, and several shortcomings, such as easy saturation of the training gradient, low classification accuracy, and poor model generalization, should be addressed.

The spectral values of the HSIs in the third dimension are approximately continuous, and the curves of each feature possess a unique spectral plot that is different from those of other classes. In the traditional classification methods, one-dimensional spectral vectors are used as the final form of input data [27, 28] or neighboring pixels are used to form small regional pixel blocks as input data [29, 30]. Although the former simplifies the complexity of deep learning network training, it omits spatial dimension information of spectral values at the same time. The latter combines multiple pixels into one sample, which introduces heterogeneous noises and aggravates the problem of missing hyperspectral data.

Compared with the traditional CNN methods, this study designs a 2D-spectrum CNN model as follows:

(i) Hyperspectral pixels have rich spectral information. The traditional data processing methods which use small area pixel block or one-dimensional spectral vector as input unit bring many heterogeneous noises. In this paper, we convert the spectral value vector to 2D-spectrum image, so that the optimization of all CNN model parameters (including the BN parameters) is based on the spectral values of the pixel points and spectral space information. The target of fully extracting spectral spatial information can be achieved while heterogeneous noises are also avoided. In addition, a multilevel BN algorithm is achieved for the first time, and the effect of network acceleration is obvious.

(ii) A BN algorithm is introduced to reduce the vanishing gradient problem and dynamically accelerate the training speed of the DNN by reducing the scaling and initialization of the dependent parameters. A small area pixel block was selected as the input unit. Liu et al. [30] used the BN algorithm to the CNN for the HIS. However, the introducing heterogeneous noises and wasting scarce samples will weaken the BN algorithm's role in network regularization and accelerated training.

(iii) Softmax loss models are used instead of combining Softmax regression and multinomial logistic loss models; thus, the output of the last layer competes with one another to improve the classification accuracy. The experimental results show that the proposed CNN-based HSI classification model exhibits high accuracy and efficiency in the HSI dataset.

2. CNN-Based Classification Model

The researchers found that the human visual system can effectively solve the problem of classification, detection, and identification, with the rapid development of modern nervous systems. This development motivates researchers on biological visual systems to establish advanced data processing methods [31]. Cells in the cortex of a human visual system are only susceptible to small areas, and accepting cells in the field can exploit the local spatial correlation in the image.

The CNN architecture uses two special methods, namely, local receptive field and shared weights. The activation value of each convolution neuron is calculated by multiplying the local input with weight W, which is shared in the entire input space (Figure 1). Neurons that belong to the same layer share the same weight. The use of specific architectures, such as local receptive field and shared weights, reduces the total number of training parameters and facilitates the development of an efficient training model.

The complete CNN architecture consists of convolution and pooling layers. The convolution layer alternates with the pooling layer, thereby mimicking the properties of complex and simple cells in the mammalian visual cortex [32]. In the CNN, the input data are a matrix or tensor with a 3D spatial structure, where (H, W), (H', W'), and (H'', W'') represent the size of the spatial dimension of input data, convolution kernel, and output data, respectively. The number of convolution kernel feature channels is represented by D, and D'' represents the 3D data.

$$\begin{aligned} x &\in \mathbb{R}^{H\times W\times D}, \\ f &\in \mathbb{R}^{H'\times W'\times D\times D''}, \\ y &\in \mathbb{R}^{H''\times W''\times D''}, \end{aligned} \tag{1}$$

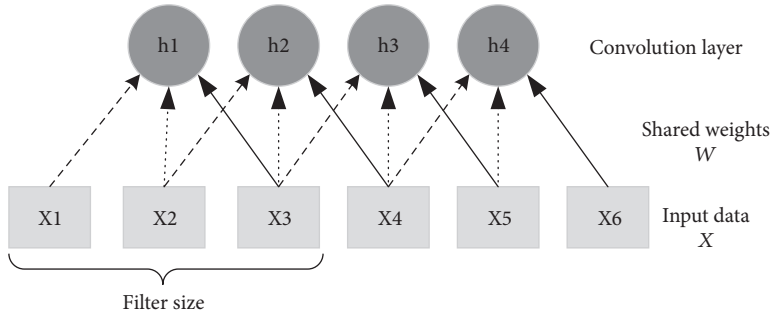


Figure 1: Local receptive field and shared weights.

where x is the input data, f is the convolution filter, and y is the output data. The 1D signal x is convoluted by filter f to calculate signal y as follows:

$$y_{i''j''d''} = b_{d''} + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d'=1}^{D} f_{i'j'd} \times x_{i''+i'-1,j''+j'-1,d',d''}, \quad (2)$$

where $b_{d''}$ is the neuron offset and $f_{i'j'd}$ is the convolution kernel matrix of the dth $i' \times j'$.

The pooling layer is typically obtained after the convolution layer. The most common pooling function is max pooling. This function calculates the maximum response of each feature channel in the $H' \times W'$ region. The feature map becomes robust to the distortion of the data and achieves a high invariance through the pooling. The pooling layer can also decrease the size of the feature map, thereby reducing computational burden.

$$y_{i''j''d''} = \max_{1 \le i' \le H', 1 \le j' \le W'} x_{i''+i'-1,j''+j'-1,d}. \quad (3)$$

The traditional CNN processing flow is designed in accordance with the data structure characteristics of the HSI classification task (Figure 2). The training data are obtained through forward propagation to determine the actual output and then compared with the real data tag competition. Stochastic gradient descent (SGD) is used to modify the synapses and parameters of the network structure, and several iterative trainings are conducted to form a network model. Test data are inputted into the network model. The output data obtained by feature extraction are matched with the actual data, and the result is classified by the competitive output model.

The 3D data of the HSIs reach hundreds, and a numerical approximation is considered continuous. The curve of each pixel possesses a unique spectral plot that differs from those of other categories, and this plot is difficult to distinguish by the human eye. However, the CNN exhibits a better performance than several human visual aspects. Therefore, this study uses spectral labels to improve the classification performance of the CNN models on the HSIs.

3. Proposed CNN-Based HSI Classification Model

3.1. BN. The SGD is used to train NN in CNN training. This method is simple and effective, but the model parameters should be carefully adjusted. In particular, the learning rate and initialization parameters of the model are added in the optimization. This step significantly reduces the speed of the CNN training. The entire network should conform to the new data distribution when the distribution of the input data at the network layer changes, thereby resulting in the decreased training speed of the network and saturated gradient. If the nonlinear input data distribution is ensured to be stable, then the probability of nonlinear saturation problem will be minimal and can accelerate the training of the network. Therefore, the BN algorithm is proposed to eliminate this phenomenon and expedite the training of the network.

In the BN algorithm, for each hidden layer of neurons, the input distribution, the value interval of which is closer to the limit saturation region through nonlinear function mapping, is forced back to the normal distribution of the comparison standard, with a mean of 0 and a variance of 1. Thus, the input value of the nonlinear transformation function falls into the region, which is sensitive to the input data, to avoid a gradient disappearance problem. The most mature technique in the early DNN normalization operation is whitening; however, whitening the input of each layer would result in excessive computational costs and computational time and not all differential. Thus, BN is used in two simplified ways.

The first step indicates that zero mean and variance normalization for scalar features are used instead of whitening. Simultaneously, the input and output of the layer are normalized. The following formula is used for NNs with d-dimensional input $x = (x^{(1)}, \ldots, x^{(d)})$ to normalize each dimension, where the predicted value and variance can be calculated from the corresponding batches.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - E\left[x^{(k)}\right]}{\sqrt{\text{var}\left[x^{(k)}\right]}}. \quad (4)$$

The primitive activation value x that corresponds to the neuron is converted by subtracting the corresponding

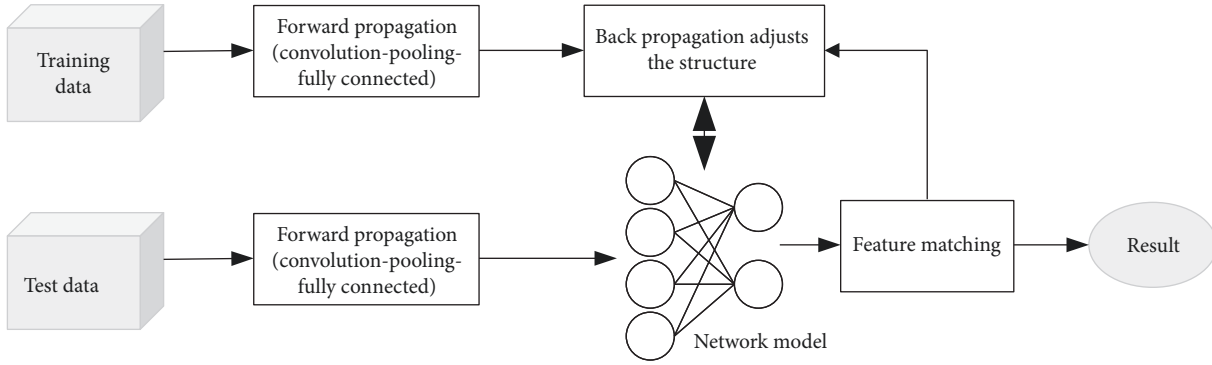


Figure 2: Traditional CNN processing flow.

batch mean $E(x)$ and dividing it by the variance $\mathrm{Var}(x)$. If the normalization for input of NN layer is rather simple, then the characterization capability of the layer is reduced. Thus, the BN introduces a pair of parameters $\gamma^{(k)}$ and $\beta^{(k)}$ for each activation value $x^{(k)}$, which can scale and translate the normalized input as

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}. \tag{5}$$

The two parameters are similar to the network parameters and are trained and modified in the same way. The characterization capability of the model is then restored. When $\gamma^{(k)} = \sqrt{\mathrm{var}\,[x^{(k)}]}, \beta^{(k)} = E[x^{(k)}]$, it can obtain the original activation value based on the output and restore representation ability. That is, the network can restore the feature distribution to be learned by the original network.

The second step denotes that NN training is based on the entire training dataset. Theoretically, the entire training dataset can be used to normalize the activation value. However, it cannot be applied to the SGD because of a large amount of calculation of the dataset. The BN introduces min-batch in the SGD and calculates the corresponding mean and variance by using the minimum batch data. If a minimum batch data B exists, then the corresponding size is m. Only one of the activation values $x^{(k)}$ is considered because each dimension of the activation value is normalized, while the variable k is then ignored. Each B has m activation values. The BN algorithm can be obtained by Formula (5).

$$\mathrm{Algorithm}_{\gamma,\beta} : x_{1\ldots m} \longrightarrow \widehat{x}_{1\ldots m} \longrightarrow y_{1\ldots m}. \tag{6}$$

The mean μ_B and variance σ_B^2 of the minimum batch data are defined as follows:

$$\begin{aligned} \mu_B &\longleftarrow \frac{1}{m}\sum_{i=1}^{m} x_i, \\ \sigma_B^2 &\longleftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2, \end{aligned} \tag{7}$$

in which $\widehat{x}$ and y can be obtained after the normalization.

$$\begin{aligned} \widehat{x}_i &\longleftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 - \varepsilon}}, \\ y_i &\longleftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i), \end{aligned} \tag{8}$$

where ε is a constant (tends to be 0) to ensure the stability of the calculation of variance.

The data computation becomes large and complex in applying the BN algorithm to the CNN if each neuron is normalized in every layer. Thus, this algorithm is based on the idea of weight sharing, and the mean and variance of the activation value are obtained for the entire map. Input x, output y, parameter W, and b are defined as

$$\begin{aligned} x, y &\in \mathbb{R}^{H\times W\times K\times T}, \\ \mathrm{W} &\in \mathbb{R}^{K}, \\ b &\in \mathbb{R}^{K}, \end{aligned} \tag{9}$$

where H, W, K, and T represent the length and width of the input and output data, number of characteristic channels, and min-batch size, respectively. We explicitly define the input and output arrays as 4D data to process the feature mappings by batch. The output characteristic map is expressed by the following formula:

$$\begin{aligned} y_{ijkt} &= W_k \frac{x_{ijkt} - \mu_k}{\sqrt{\sigma_k^2 + \varepsilon}} + b_k, \\ \mu_k &= \frac{1}{HWT}\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{t=1}^{T} x_{ijkt}, \\ \sigma_k^2 &= \frac{1}{HWT}\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{t=1}^{T}\left(x_{ijkt} - \mu_k\right). \end{aligned} \tag{10}$$

The BN is performed to the activation value of each hidden layer of neurons, which can be regarded as an add-on operation layer. It is located after the activation

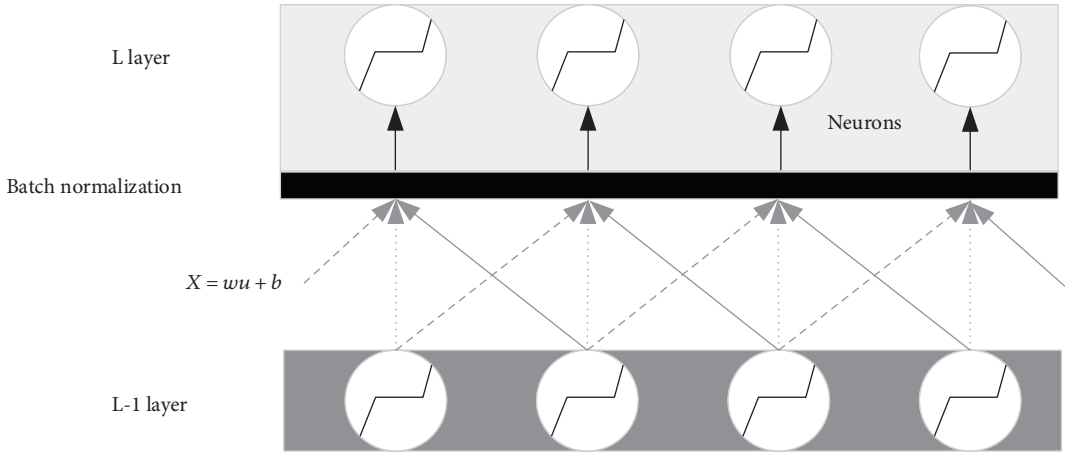


Figure 3: Operation layer of BN.

value $x = wu + b$ obtained and before the nonlinear activation function, as depicted in Figure 3.

The proposed BN algorithm possesses the following characteristics after the theoretical analysis and experimental verification. (1) A large initial learning rate can be selected so as to improve training speed. (2) Dropout can be removed and L2 weight attenuation coefficient can be reduced. (3) Local response normalization (LRN) can be replaced. (4) The training data can be completely disrupted (i.e., a sample is not frequently selected in each batch of training).

3.2. Softmax Loss Models. The Softmax regression model is a spread of logistic regression model in solving multiclass problems. Following the data structure of Section 4.1, it can obtain the following output of the regression function.

$$y_{ijk} = \frac{e^{x_{ijk}}}{\sum_{t=1}^{D} e^{x_{ijt}}}. \tag{11}$$

This formula is not limited by the number of feature channels and is applied to all spatial positions in a convoluted manner to translate linear predictions into categorical probabilities. The linear prediction results x_{ijk} of the kth category serve as input to the regression model, and D probability values (likelihood), which represent the possibility of the data belonging to different categories, are obtained. The Softmax regression model can be regarded as the combination of the exponential form of activation function and normalized operators. The classification loss function $\ell(x, c)$ is aimed at comparing the prediction x with the real class label c. The classification loss is defined as follows:

$$\ell(x, c) = \sum_{ijn} \omega_{ij1n} \ell\left(x_{ij:n}, c_{ij:n}\right), \tag{12}$$

where $x \in \mathbb{R}^{H \times W \times C \times N}$, $c \in \{1, \ldots, C\}^{H \times W \times 1 \times N}$, and N is the size of the 3D data. The 1D vector $x_{ij:n}$ represents class C fraction, and $c_{ij:n}$ represents a real class label. Logarithmic loss function or logarithmic likelihood loss function is commonly used in logistic regression, which is based on the maximum likelihood principle. Thus, vector x represents the posterior probability $p(k) = x_k, k = 1, \ldots, C$ of the different classes. The output of the loss function is the negative logarithmic probability of the real label.

$$\begin{aligned} &\ell(x, c) = -\log x_c \cdot x \geq 0, \\ &\sum_k x_k = 1, \end{aligned} \tag{13}$$

where x is the output of the Softmax regression model and is numerically unstable. On the one hand, the score x_c should compete with other scores $x_k (k \neq c)$ to obtain a meaningful logarithmic loss. If it is not the case, then the minimization of Formula (13) can be achieved by maximizing all x_k, but the real prediction effect is that x_c is larger than $x_k (k \neq c)$. On the other hand, the Softmax regression model allows the score x to compete through the normalization factor. This study proposes the Softmax loss models, which combines the calculation module of the regression model and calculation module of logarithmic loss into a single one.

$$\ell(x, c) = -\log \frac{e^{x_c}}{\sum_{k=1}^{C} e^{x_k}} = -x_c + \log \sum_{k=1}^{C} e^{x_k}. \tag{14}$$

The combined modules result in a stable value of the output fraction x. By combining the logarithmic loss with Softmax, the loss model automatically makes the score compete $\ell(bx, c) \approx 0$, when $x_c \gg \sum_{k \neq c} x_k$. Although this model is similar to the final output result of the logarithmic loss function, the experimental results show that the Softmax loss model has the following advantages. (1) The calculation steps are minimal, while the calculated amount is small. (2) Numerical gradients are relatively stable. (3) Competitive output can improve the classification accuracy.

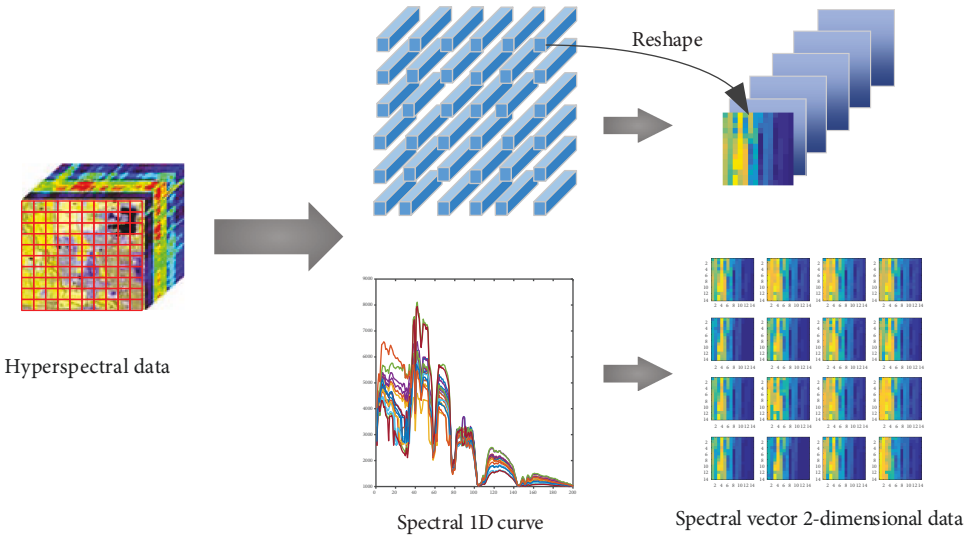


Figure 4: Hyperspectral image spectral vector preprocessing. Compared with the one-dimensional curve, the 2D spectrum has more obvious distinguishing features between different categories.

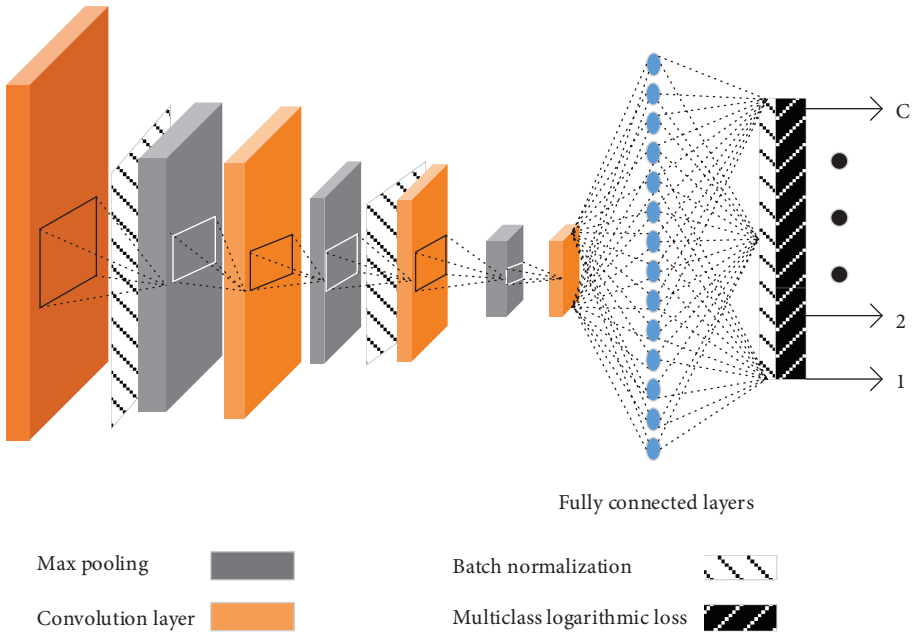


Figure 5: 2D-spectrum CNN structure.

3.3. Two-Dimensional Spectrum CNN Model. The hyperspectral data has a small total number of pixels, while a single pixel has rich spectral values. According to this, the 2D-spectrum method is proposed in the paper, which converts the pixel spectral vectors into two-dimensional spectral images as the input data of CNN. The convolution network can fully utilize the spatial position information between the different spectral values to improve the classification accuracy, as shown in Figure 4. The network adopts multiple convolution and max pooling layers alternately. BN layers, which regulate the data distribution and accelerate network training, are insert among the one, four, and seven layers. The Softmax loss model is then used to control the output. The 2D-spectrum CNN structure is shown in Figure 5.

4. Experimental Results

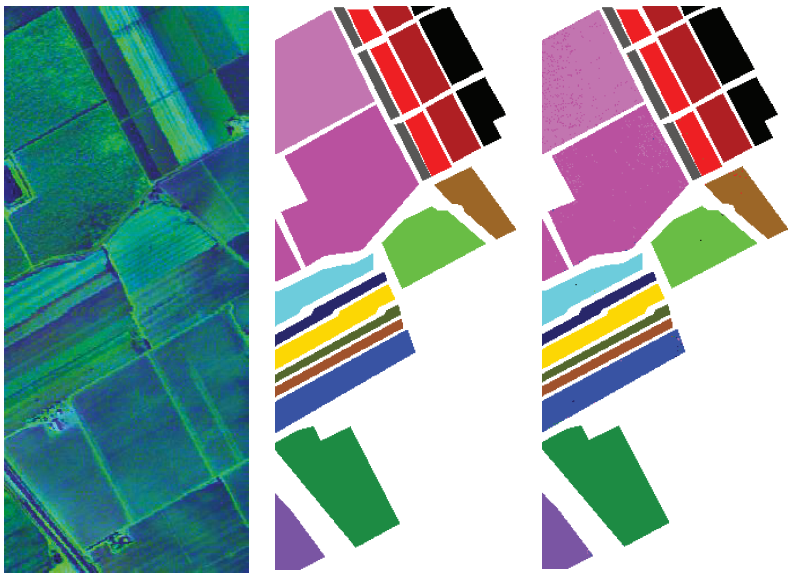
In this chapter, we first describe the dataset used in the experiment. Furthermore, the BN algorithm and Softmax loss model are used to verify the HSI classification performance of the CNN model. Finally, the proposed method is compared with other similar methods to determine the advantages and disadvantages of the 2D-spectrum CNN

Table 1: Hyperspectral data set features.

Dataset	Size	Spectral bands (original)	Labels	Instrument
Indian Pines	145×145	200 (224)	16	AVIRIS
Salinas	512×217	204 (224)	16	AVIRIS
KSC	512×614	176 (224)	13	NASA AVIRIS
Botswana	1476×256	145 (242)	14	NASA EO-1

(a) (b) (c)

Figure 6: (a) AVIRIS Infrared image, (b) Indian Pines ground-truth map, and (c) classification map.



(a) (b) (c)

Figure 7: (a) RGB color image from random three bands, (b) Salinas ground-truth map, and (c) Salinas classification map.

model. The overall accuracy (OA) and Kappa coefficients are used as the performance measurements for each type of classification accuracy (percentage). The mathematical relationship between error and OA is

$$\mathrm{OA} = (1 - \mathrm{error}) \times 100\%. \tag{15}$$

Each classification result is averaged over 10 runs to avoid any deviation caused by random sampling. All of the experiments are tested on a desktop with NVIDIA GeForce GTX 1070 8G GPU, 16 GB memory, and 64-bit Windows 7 OS using MATLAB 2014b.

4.1. Hyperspectral Datasets. In the experiment, the hyperspectral data of the network model include Indian Pines, Salinas, and Kennedy Space Center (KSC) datasets and are applied to the new dataset, Botswana. Table 1 summarizes the detailed information of the four datasets.

Figures 6–8 exhibit the characteristics of the four datasets. Tables 2–5 list the number of samples trained and tested

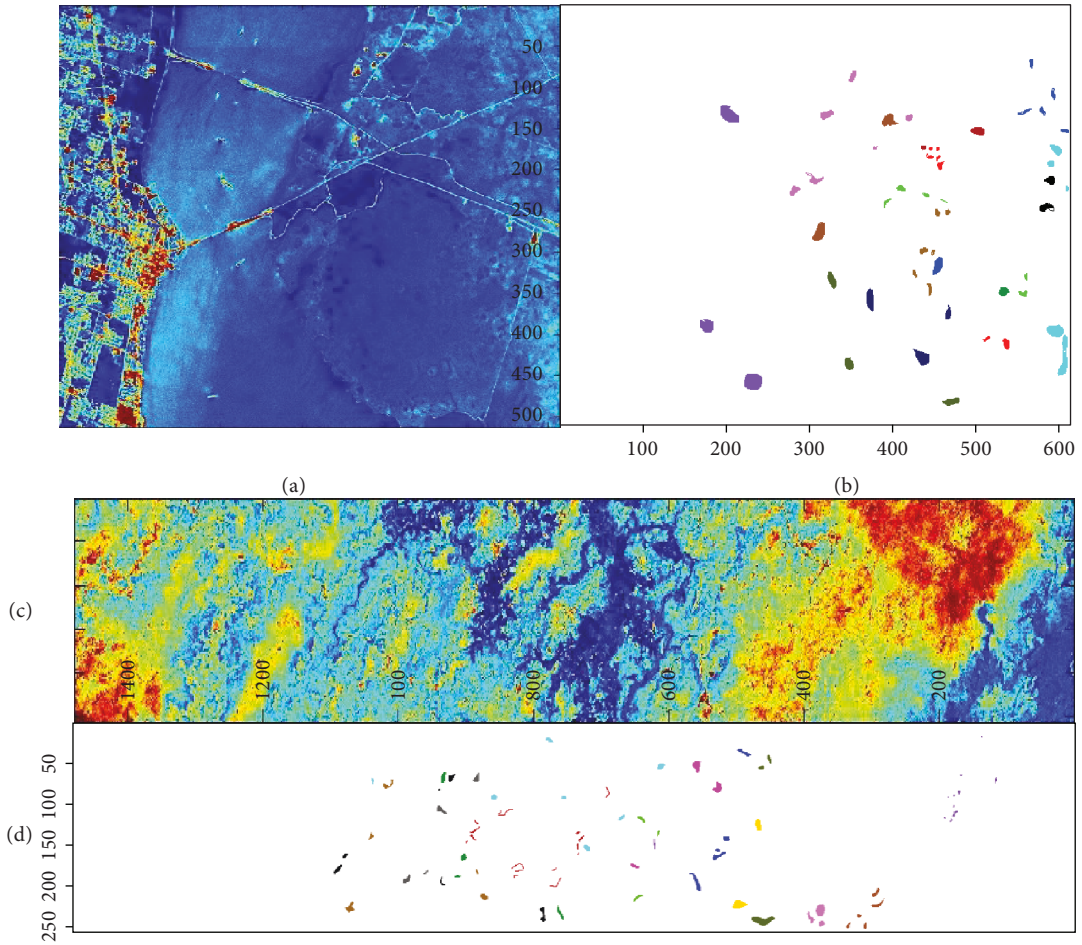


Figure 8: (a) KSC third band, (b) KSC ground-truth map, (c) Botswana fourth band, and (d) Botswana ground-truth map.

Table 2: Number of labeled training samples and testing samples of Indian Pines.

Class number	Class name	Training samples	Testing samples
1	Alfalfa	**35**	**11**
2	Corn-no till	357	1071
3	Corn-min till	207	623
4	Corn	59	178
5	Grass-pasture	120	363
6	Grass-trees	182	548
7	Grass-pasture-mowed	**21**	**7**
8	Hay-windrowed	119	359
9	Oats	**15**	**5**
10	Soybean-no till	243	729
11	Soybean-min till	613	1842
12	Soybean-clean	148	445
13	Wheat	51	154
14	Woods	316	949
15	Buildings-grass-trees-drives	96	290
16	Stone-steel-towers	**70**	**23**
Total		2652	7597

Table 3: Number of labeled training samples and testing samples of Salinas.

Class number	Class name	Training samples	Testing samples
1	Brocoli_green_weeds_1	503	1506
2	Brocoli_green_weeds_2	932	2794
3	Fallow	494	1482
4	Fallow_rough_plow	349	1045
5	Fallow_smooth	670	2008
6	Stubble	990	2969
7	Celery	895	2684
8	Grapes_untrained	2818	8453
9	Soil_vinyard_develop	1550	4653
10	Corn_senesced_green_weeds	820	2458
11	Lettuce_romaine_4wk	267	801
12	Lettuce_romaine_5wk	482	1445
13	Lettuce_romaine_6wk	229	687
14	Lettuce_romaine_7wk	268	802
15	Vinyard_untrained	1817	5451
16	Vinyard_vertical_trellis	452	1355
Total		13536	40593

Table 4: Number of labeled training samples and testing samples of KSC.

Class number	Class name	Training samples	Testing samples
1	Scrub	190	571
2	Willow swamp	60	183
3	CP hammock	64	192
4	Slash pine	63	189
5	Oak/broadleaf	40	121
6	Hardwood	58	171
7	Swamp	27	78
8	Graminoid marsh	108	323
9	Spartina marsh	130	390
10	Cattail marsh	101	303
11	Salt marsh	105	314
12	Mud flats	126	377
13	Water	232	695
Total		1304	3907

Table 5: Number of labeled training samples and testing samples of Botswana.

Class number	Class name	Training samples	Testing samples
1	Water	67	203
2	Hippo grass	25	76
3	Floodplain grasses 1	62	189
4	Floodplain grasses 2	53	162
5	Reeds	67	202
6	Riparian	67	202
7	Firescar	64	195
8	Island interior	50	153
9	Acacia woodlands	78	236
10	Acacia shrublands	62	186
11	Acacia grasslands	76	229
12	Short mopane	45	136
13	Mixed mopane	67	201
14	Exposed soils	23	72
Total		806	2442

for the corresponding dataset and are assigned according to (train) 1 : (test) 3. According to previous experiments, a considerably minimal training data will produce an underfitting phenomenon. The labels 1, 7, 9, and 16 of the Indian Pines datasets are insufficient. The above labels allocate the total amount of samples according to 3 (train) : 1 (test) to avoid underfitting and to ensure that the training sample is sufficient.

The learning rate of the CNN network model is [1e-03 5e-04 1e-05], the corresponding number of training iterations is 50, 30, and 20, and the number of samples per batch is 100.

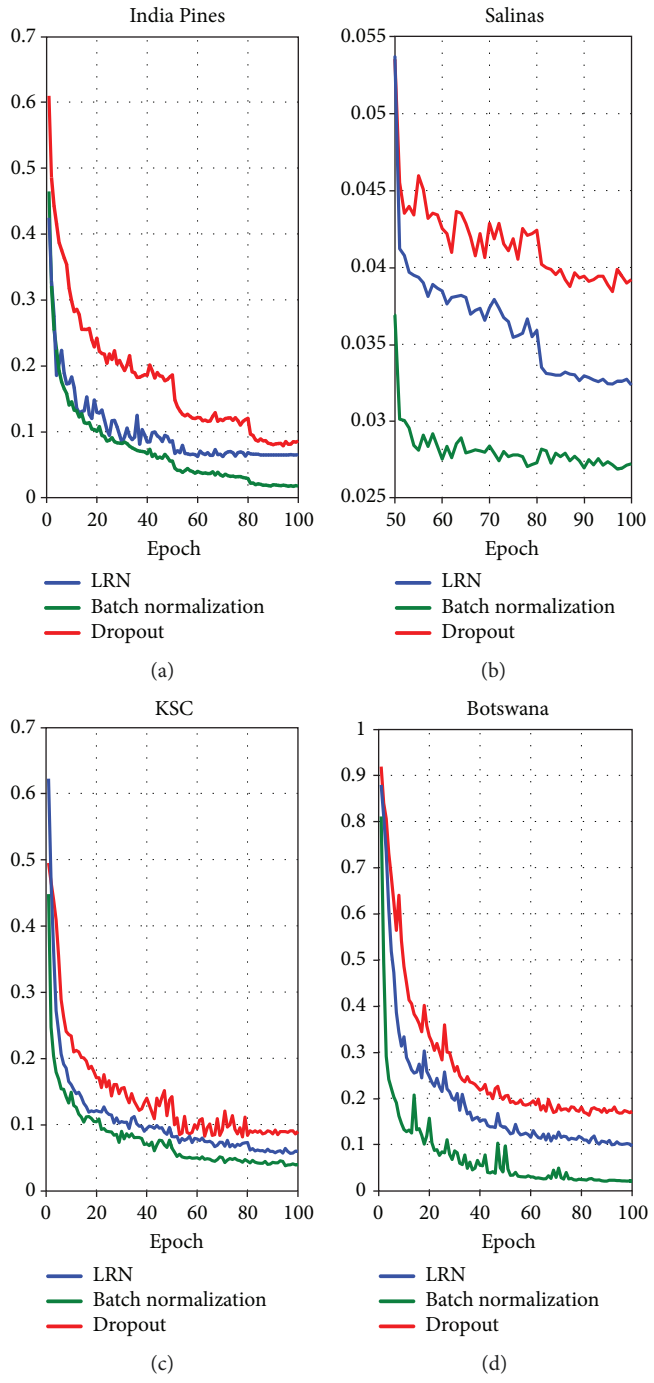


Figure 9: BN performance verification curve: y-axis is error.

4.2. BN Performance Verification. The BN operation layer is added in the CNN to solve the problem of modified distribution of the internal nodes caused by the change in the input data of the traditional CNN. The addition of this layer can solve the problem of training saturation and gradient disappearance, significantly increase the training speed, and improve the classification performance. According to Section 4.1, the BN operator layer can replace the traditional dropout layer and the local response normalization layer (LRN) and achieve a more satisfactory result, as presented in Figure 9 and Table 6.

Table 6: Performance comparison of dropout, LRN, and BN.

Dataset	Operating layer	Unit time of batch/s*	Error	Kappa
Indian pines	Dropout	0.940	0.0860	0.905
	LRN	0.898	0.0629	0.930
	B N	0.742	**0.0174**	0.978
Salinas	Dropout	0.671	0.0392	0.945
	LRN	0.360	0.0323	0.939
	B N	0.342	**0.0272**	0.972
KSC	Dropout	0.115	0.0893	0.892
	LRN	0.114	0.0592	0.922
	B N	0.099	**0.0378**	0.956
Botswana	Dropout	0.362	0.1709	0.829
	LRN	0.323	0.0970	0.886
	B N	0.283	**0.0207**	0.963

*Owing to the number of samples between the datasets and the specific parameters of the training, this entry only applies to the comparison within the same dataset.

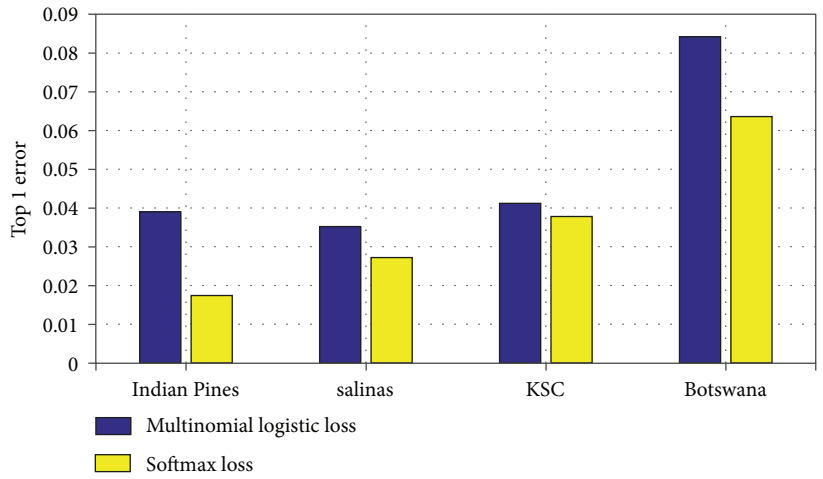


Figure 10: Classification accuracy of the different output models.

Table 7: Classification accuracy of different outputs model (error).

Output model	Performance	Indian Pines	Salinas	KSC	Botswana
Multinomial logistic loss	OA	0.0390	0.0352	0.0412	0.0842
	Kappa	0.952	0.948	0.952	0.914
Softmax loss	OA	**0.0174**	**0.0272**	**0.0378**	**0.0207**
	Kappa	0.978	0.972	0.956	0.963

Table 6 indicates that dropout and LRN are trained on the network to accelerate and prevent the overfitting phenomenon. However, the BN operation layer in training acceleration and classification accuracy are better than the two areas by data comparison, and the experimental results depict obvious differences. Figure 10 denotes the test results for error in Table 6.

4.3. Softmax Loss Models. The CNN in this study uses the Softmax loss models in the output to replace the traditional Softmax regression and multinomial logistic loss models. Table 7 compares the classification accuracy among the three models.

Figure 10 is a more intuitive histogram. The results of four hyperspectral data experiments show that the Softmax loss model has the highest classification accuracy, compared with the multinomial logistic loss model.

4.4. Comparison with Other Approaches. The proposed method in this study is compared with recent works on

Table 8: Comparison of CNNs with other algorithms in Indian Pines, Salinas, KSC, and Botswana.

Dataset	Performance	CNN (vector)	PPF (vector)	BASS (blocks)	2D-spectrum CNN
Indian Pines	OA	86.44	94.34	96.77	**98.26**
	Kappa	0.856	0.922	0.955	0.978
Salinas	OA	89.28	94.80	95.36	**97.28**
	Kappa	0.878	0.931	0.928	0.962
KSC	OA	88.38	93.18	94.53	**96.22**
	Kappa	0.870	0.912	0.939	**0.956**
Botswana	OA	89.45	91.57	95.42	**97.93**
	Kappa	0.885	0.908	0.939	**0.963**

Vector: one-dimensional spectral vectors. Blocks: small-area pixel division blocks.

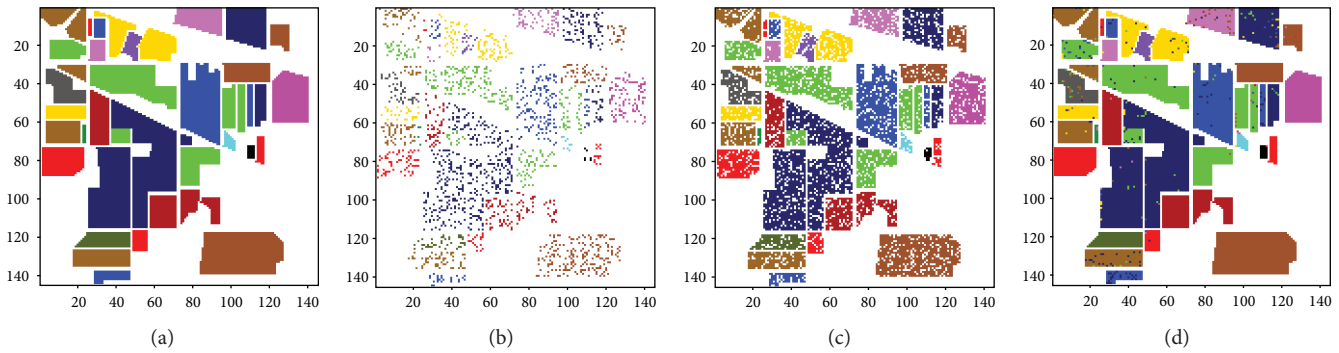


Figure 11: Indian Pines dataset. (a) The ground truth, (b) training samples, (c) test samples, and (d) classification maps.

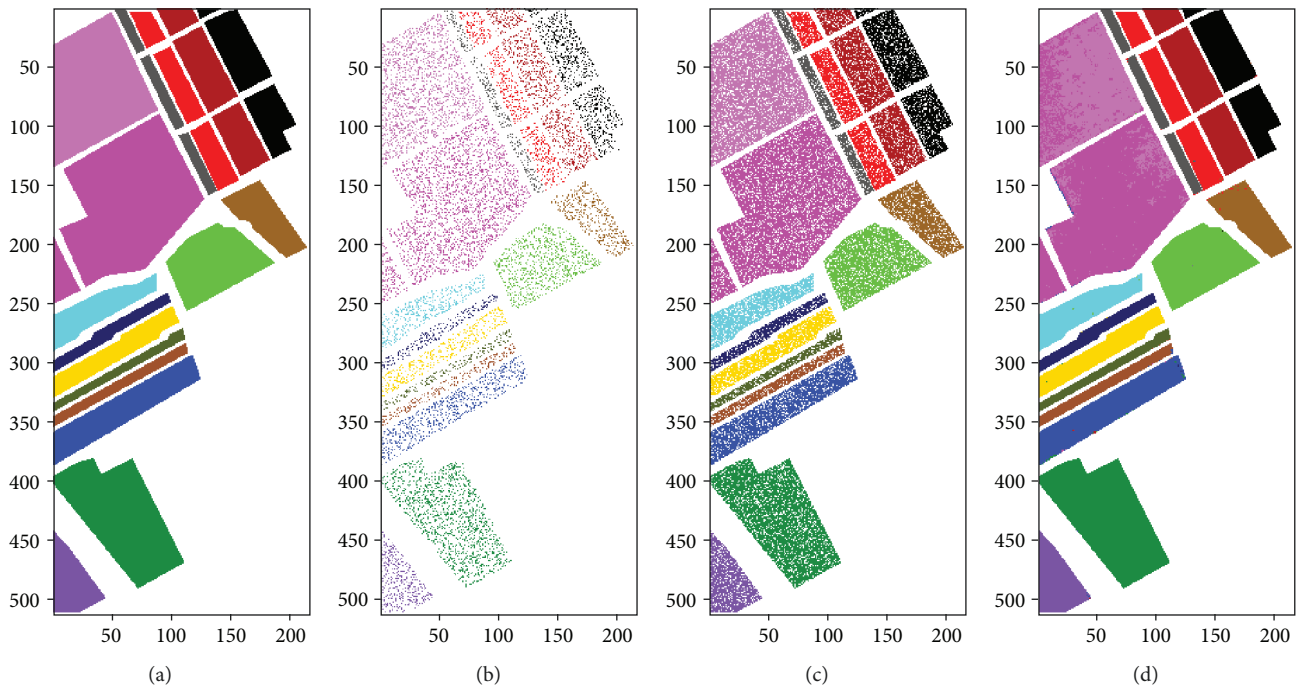


Figure 12: Salinas data set. (a) The ground truth, (b) training samples, (c) test samples, and (d) classification maps.

the HSI classification studies. The training accuracy evaluation indicators, namely, OA and Kappa coefficients, are important criteria for assessing network model classification performance. In Table 8, the proposed 2D-spectrum CNN model is compared with other advanced deep learning methods. The comparative data is derived from the work of

Hu et al. [27], and the CNN data based on PPFs is from the research of Li et al. [28]. The data of Band-Adaptive Spectral-Spatial (BASS) net architecture is from the paper of Santara et al. [29].

Figures 11 and 12 display the ground truth, training set, test set, and classification maps of Indian Pines and Salinas. It can be seen from the comparison of the performance that the classification performance of the 2D-spectrum CNN model is superior to other methods in different hyperspectral data in terms of overall accuracy and Kappa coefficient.

5. Conclusion

This paper proposes the 2D-spectrum CNN model that adds multilevel BN operating layers for HSI classification. The output uses a Softmax loss model for classification. In addition, many of the relevant factors include the number of iterations, learning rate, size of the input data, and size of the filter and feature graph. All of them will affect the final classification performance. The parameters obtained the optimal solution and reached the ideal effect in the experiments. The experiment results show that the CNN model proposed in this paper provides excellent performance. The training process converges quickly, thereby indicating that this method can be applied to large datasets. Furthermore, high classification accuracy can be achieved by applying sufficient iterations and number of datasets.

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.

Authors' Contributions

Hongmin Gao and Shuo Lin contributed equally to this work and should be considered co-first authors.

Acknowledgments

This study is supported by the National Natural Science Foundation of China (no. 61701166), Projects in the National Science & Technology Pillar Program during the Twelfth Five-Year Plan Period (no. 2015BAB07B01), the Fundamental Research Funds for the Central Universities (no. 2018B16314), the China Postdoctoral Science Foundation (no. 2018M632215), National Science Foundation for Young Scientists of China (no. 51709271), and Young Elite Scientists Sponsorship Program by China Association for Science and Technology (no. 2017QNRC001).

References

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

[2] C. J. Zhang, Y. N. Wang, X. Lu, K. N. Wang, and G. F. Wang, "Front-vehicle detection algorithm based on hypothesis and verification of improved HOG feature," *Journal of Electronic Measurement and Instrumentation*, vol. 29, no. 2, pp. 165–171, 2015.

[3] G. G. Zhang and X. H. Xu, "SAR image target recognition algorithms based on weighted texture features," *Foreign Electronic Measurement Technology*, vol. 34, no. 9, pp. 22–25, 2015.

[4] 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 Geoscience and Remote Sensing Letters*, vol. 9, no. 3, pp. 447–451, 2012.

[5] A. Villa, J. A. Benediktsson, J. Chanussot, and C. Jutten, "Hyperspectral image classification with independent component discriminant analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 12, pp. 4865–4876, 2011.

[6] T. V. Bandos, L. Bruzzone, and G. Camps-Valls, "Classification of hyperspectral images with regularized linear discriminant analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 3, pp. 862–873, 2009.

[7] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy K-nearest neighbor algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 4, pp. 580–585, 1985.

[8] L. Jiang, C. Li, S. Wang, and L. Zhang, "Deep feature weighting for naive Bayes and its application to text classification," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 26–39, 2016.

[9] S.-B. Bai, J. Wang, G.-N. Lü, P.-G. Zhou, S.-S. Hou, and S.-N. Xu, "GIS-based logistic regression for landslide susceptibility mapping of the Zhongxian segment in the Three Gorges area, China," *Geomorphology*, vol. 115, no. 1-2, pp. 23–31, 2010.

[10] M. F. Yan, G. F. Zhao, Z. F. Liu, Q. Zhang, Z. X. Wang, and Z. Y. Du, *L-Fuzzy Sets and L-Fuzzy Support Vector Machine*, Journal of Tangshan Normal University, 2015.

[11] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral-spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 844–856, 2013.

[12] 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 Processing Magazine*, vol. 31, no. 1, pp. 55–66, 2014.

[13] Q. Gao, W. Yang, and Q. Li, "Research on deep belief network layer tendency and its application into identifying fault images of aerial images," *Chinese Journal of Scientific Instrument*, vol. 36, no. 6, pp. 1267–1274, 2015.

[14] P. M. Atkinson and A. R. L. Tatnall, "Introduction neural networks in remote sensing," *International Journal of Remote Sensing*, vol. 18, no. 4, pp. 699–709, 1997.

[15] L. Bruzzone and D. F. Prieto, "A technique for the selection of kernel-function parameters in RBF neural networks for classification of remote-sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 2, pp. 1179–1184, 1999.

[16] F. Ratle, G. Camps-Valls, and J. Weston, "Semi-supervised neural networks for efficient hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 5, pp. 2271–2282, 2010.

[17] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[18] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann machines," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 448–455, Clearwater Beach, FL, USA, 2009.

[19] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer wise training of deep networks," in *Proceedings of Neural Information Processing Systems*, pp. 153–160, Cambridge, MA, USA, 2007.

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

[21] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectral–spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2438–2442, 2015.

[22] Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2381–2392, 2015.

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

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

[25] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 10, pp. 6232–6251, 2016.

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

[27] 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.

[28] W. Li, G. Wu, F. Zhang, and Q. Du, "Hyperspectral image classification using deep pixel-pair features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 844–853, 2017.

[29] A. Santara, K. Mani, P. Hatwar et al., "BASS net: band-adaptive spectral-spatial feature learning neural network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 9, pp. 5293–5301, 2017.

[30] B. Liu, X. Yu, P. Zhang, X. Tan, A. Yu, and Z. Xue, "A semi-supervised convolutional neural network for hyperspectral image classification," *Remote Sensing Letters*, vol. 8, no. 9, pp. 839–848, 2017.

[31] N. Kruger, P. Janssen, S. Kalkan et al., "Deep hierarchies in the primate visual cortex: what can we learn for computer vision?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1847–1871, 2013.

[32] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.