Journal of Applied Remote Sensing

RemoteSensing.SPIEDigitalLibrary.org

Three-dimensional densely connected convolutional network for hyperspectral remote sensing image classification

Chunju Zhang
Guandong Li
Shihong Du
Wuzhou Tan
Fei Gao

SPIE.

Chunju Zhang, Guandong Li, Shihong Du, Wuzhou Tan, Fei Gao, "Three-dimensional densely connected convolutional network for hyperspectral remote sensing image classification," *J. Appl. Remote Sens.* **13**(1), 016519 (2019), doi: 10.1117/1.JRS.13.016519.

Three-dimensional densely connected convolutional network for hyperspectral remote sensing image classification

Chunju Zhang,[a,*] Guandong Li,[a,*] Shihong Du,[b] Wuzhou Tan,[c] and Fei Gao[a]

[a]Hefei University of Technology, School of Civil Engineering, Hefei, China
[b]Peking University, Institute of Remote Sensing and Geographic Information System, Beijing, China
[c]Anhui Bureau of Surveying and Mapping, Hefei, China

Abstract. Hyperspectral remote sensing images (HSIs) are rich in spatial and spectral information, thus they help to enhance the ability to distinguish geographic objects. In recent years, great progress have been made in image classification using deep learning (such as 2D-CNN and 3D-CNN). Compared with traditional machine learning methods, deep learning methods can automatically extract the abstract features from low to high levels and convert the images into more easily recognizable features. Most HSI classification tasks focus on spectral information but often ignore the rich spatial structures in HSIs, leading to a low classification accuracy. Moreover, most supervised learning methods use shallow structures in HSI classifications and hence exhibit weak performance in finding sparse geographic objects. We proposed to use the three-dimensional (3-D) structure to extract spectral–spatial information to build a deep neural network for HSI classifications. Based on DenseNet, the 3D densely connected convolutional network was improved to learn spectral-spatial features of HSIs. The densely connected structure can enhance feature transmission, support feature reuse, improve information flow in the network, and make deeper networks easier to train. The 3D-DenseNet has a deeper structure than 3D-CNN, thus it can learn more robust spectral–spatial features from HSIs. In fact, the deeper network structure has a regularized effect, which can effectively reduce overfitting on small sample datasets. The network uses HSIs instead of feature engineering as input data and is trained in an end-to-end manner. The experimental results of this model on the Indian Pines datasets and the Pavia University datasets show that deeper neural networks further improve the classification of complex objects, especially in the areas where geographic objects are sparse. It effectively improves the classification accuracy of HSIs. [DOI: 10.1117/1.JRS.13.016519]

Keywords: Three-dimensional-DenseNet; feature fusion; spectral–spatial features; hyperspectral remote sensing image classification; DenseNet.

Paper 180588 received Jul. 11, 2018; accepted for publication Dec. 20, 2018; published online Feb. 23, 2019.

1 Introduction

Hyperspectral remote sensing images (HSIs) take the form of a data cube containing spectral and spatial information. Unlike multispectral remote sensing data, HSIs can obtain information on hundreds of contiguous spectrum segments of geographic objects. They provide rich spectral information to enhance the ability to distinguish geographic objects.[1] However, due to the high-dimensional characteristics of the signal, information redundancy, and multiple uncertainties, the data structure of HSIs is highly nonlinear. Some classification models based on statistical patterns do not perform well in classifying and recognizing original hyperspectral

*Address all correspondence to Chunju Zhang, E-mail: zcjtwz@sina.com; Guandong Li, E-mail: leeguandon@gmail.com

1931-3195/2019/$25.00 © 2019 SPIE

data.[2] The amount of available training data for supervised learning is very limited.[3] With such a limited number of training samples, there will be a Hughes phenomenon.[4] That is, the classification accuracy will decrease as feature dimension increases. In actual classification tasks, most HSI classification methods focus on spectral information, but largely ignore the rich spatial structure of HSIs, leading to a low classification accuracy. The situation may be improved by exploiting the spatial structure information in the HSI classification.[5]

Traditional pixel-level HSI classification methods are based on a framework of feature engineering and classifiers.[6] The purpose of feature engineering is first to extract discriminative features from HSIs and then use the extracted features to train classifiers. However, feature extraction using manually engineering is essentially limited and incomprehensive. There is a need to introduce more effective and robust methods for HSI classifications. Recently, considerable progress has been made in image classification using deep learning.[7–10] Compared with the manually designed features, deep learning methods can automatically extract abstract features at multiple semantic levels and convert the images into more easily recognizable features. The classifier's objective is to generate image-to-label mapping. For example, stacked autoencoder (SAE)[8] and deep brief network (DBN)[11] use unsupervised methods to learn HSI features. For spectral–spatial feature extraction and classification, SAE and DBN can extract deep features using a process of hierarchical training. However, training samples must be flattened to one-dimensional vectors to meet the requirements of the model inputs, but in doing so, the flattened training samples will lose the spatial information of the original image. In addition, DBN and SAE are unsupervised learning methods and cannot fully utilize label information. Therefore, it was suggested to use two-dimensional convolutional neural networks (2D-CNN) to extract spatial features and use principal component analysis to reduce the dimension of the input image.[12–14] However, the 2D-CNN-based methods separate the spatial features from the spectral features, do not make full use of the combined spatial and spectral information, and require complex preprocessing. To make full use of the spectral–spatial information in HSIs, the study[15] uses three-dimensional convolutional neural networks (3D-CNN) to extract deep features and provides a good classification performance. Based on 3D-CNN, the authors of Ref. 16 proposed to use a spectral–spatial residual network, as well as spectral and spatial residual blocks to consecutively learn discriminative features from the rich spectral signatures and spatial contexts of HSIs. The two residual blocks are continuous in structure, but the spatial and spectral dimensions are sampled sequentially by changing the sampling dimension. However, the approach does not make full use of the advantages that 3D-CNN can achieve in simultaneous spatial and spectral features. Thus, most deep learning models use relatively shallow networks with only a few convolutional layers[15] and cannot effectively extract deep features in HSIs. As a consequence, they do not exhibit high performance in classifying complex objects, especially in small areas where the objects are sparse.

Here we explore the potential of the 3-D structure of 3D-CNN to extract spectral–spatial information with a view to creating a deep neural network for HSI classifications. Based on DenseNet, the 3-D densely connected convolutional network (3D-DenseNet) was improved for HSI classifications. Through the densely connected structure, it enhances feature transmission, encourages feature reuse, and improves information flow in the network. We used the characteristics of the 3D-CNN to extract spectral–spatial information, which is combined with the dense connectivity of a deep network built in DenseNet to creat 3D-DenseNet. When compared with an existing 3D-CNN, the 3D-DenseNet is deeper in structure and can learn more robust spectral–spatial features from HSIs. Moreover, the deeper network structure has a regularized effect and can effectively reduce overfitting on small sample datasets.[14] In addition, the deeper neural networks will further enhance the ability to learn the classification of complex objects, especially in small areas where the objects are sparse but the spectral dimensions are still high.

The remaining parts of this paper are organized as follows. Section 2 presents a 3-D densely connected convolutional network for HSI classifications, Sec. 3 illustrates experimental datasets, and Sec. 4 discusses experimental parameters. Section 5 shows experimental results, and in Sec. 6, we summarize the conclusions and discuss future research directions.

2 3-D Densely Connected Convolutional Network for HSI Classification

2.1 *DenseNet*

DenseNet[17] is a CNN with more than 100 layers for image recognition. The core idea of DenseNet is to establish the connection relationships between different layers, which help to make full use of the features, effectively reducing the problem of gradient disappearance and making the network deeper. In addition, adopting the bottleneck layer, the translation layer, and the smaller growth rate make the network narrower, which reduces the model parameters, effectively suppresses overfitting and saves computational power. On the benchmark datasets such as CIFAR-10,[18] CIFAR-100,[18] SVHN,[19] and ImageNet,[20] DenseNet has significantly outperformed most classification methods in accuracy.

Based on the characteristics of DenseNet, 3D-DenseNet was developed by adding spectral dimensions to DenseNet's convolution kernel and pooling layers. DenseNet is based on a 2D-CNN, whereas the 3D-DenseNet is based on a 3D-CNN.[21] 3D-CNN uses a 3-D convolution kernel to perform convolution operations. Although there are many ways to combine a 2D-CNN with separately extracted spectral information, these methods often require complex preprocessing and postprocessing. The resulting using a 2D-CNN is weaker than a 3D-CNN in generalization.[22] 3D-DenseNet[21] retains the dense connectivity in the DenseNet architecture, enabling deep feature extraction. Furthermore, it makes full use of the spectral and spatial information provided by HSIs simultaneously. For 3D-DenseNet, the dense block is a primary structure composed of densely connected composite functions, which consist of three consecutive operations: BN,[23] ReLU,[24] and a 3-D convolutional layer. The transition layers between different dense blocks are 3-D convolution and 3-D pooling operations. In the experiment, adding a 3-D pooling layer after the 3-D convolution layer is a normal operation in 3D-CNN because the pooling operation can keep the spatial characteristics unchanged. It can effectively reduce the size of spectral–spatial features, and the number of the model parameters. Also, it can save computing power and control overfitting problems. There is a global average pooling 3-D after the last dense block. Using ReLU as an activation function in 3D-DenseNet can effectively prevent the model from overfitting and increase the nonlinearity of the model. In general, ReLU is faster and more accurate than other activation functions and is widely used in neural networks.

2.2 *3-D Densely Convolution Network*

2.2.1 *Dense connectivity*

Dense connectivity implies that a layer directly connects to all of its subsequent layers. Layer l sends its feature map y_l to all subsequent layers, such as $l+1, l+2, \ldots, L$, where L is the number of layers. Correspondingly, layer l will receive the feature maps of all previous layers as input:[21]

$$y_l = F_l([y_0, y_1, \ldots, y_{l-1}]), \tag{1}$$

where $[y_0, y_1, \ldots, y_{l-1}]$ is the concatenation of composite function outputs from layer 0 to $l-1$. Dense connectivity is the core idea of DenseNet. The 3D-DenseNet[21] inherits this idea for HSI classifications (Fig. 1).

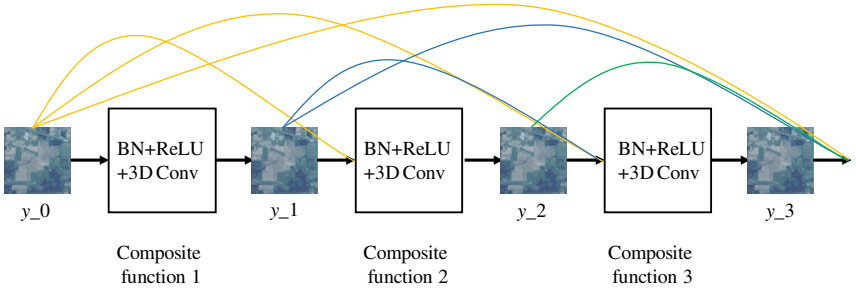

Fig. 1 Dense connectivity diagram.

Since there is no need for a large number of repetitive learning feature maps, a densely connected mode generally requires fewer parameters than a traditional convolutional network. In addition, dense connection further improves the flow of information between different layers.

2.2.2 *Structure of 3D-DenseNet*

Composite function. The composite function consists of BN, ReLU, and 3D convolutional layers, with the BN layer being placed before the ReLU layer. It aims at normalizing the input data so that the input is not too large or too small within the input field of the activation function. It smoothes the optimization landscape. This helps to accelerate the model training process and increase accuracy. We designed two kinds of composite functions, one is connected without a bottleneck layer and the other includes a bottleneck layer, which can effectively reduce the parameters (Fig. 2). We set up three dense blocks, each of which contains multiple composite functions in the form of a feedforward neural network.

Growth rate. It assumes that each composite function in the layer l produces k feature maps and the input to the layer l has $k \times (l-1) + k_0$ feature maps, where k_0 is the number of channels in the current composite function. As a result, parameter k is the growth rate of the network and determines the size of the network, thus k should not be set too large in order to prevent neural networks growing too quickly. In the experiment, we show that as k increases, OA is also increasing. Relatively small growth rates are not sufficient for feature acquisition. One explanation for this is that each layer has access to all the preceding feature-maps in its block. We can view the feature-maps as the global state of the network. Each layer adds k feature-maps of its own to this state. The growth rate regulates how much new information that each layer contributes to the global state. The global state can be accessed from everywhere within the network. We chose $4k$[17] as our parameter value.

3-D bottleneck layer. Even if each layer only outputs k feature maps, the model parameters are still large. A $1 \times 1 \times 1$ 3-D convolutional layer was used as a 3-D bottleneck layer to reduce the input $4k$ feature map[17] before the $3 \times 3 \times 3$ 3-D convolution, where k is the growth rate. The operation mentioned in the work of Refs. 25 and 26 can significantly reduce training parameters and improve computational efficiency. Generally, the design of BN-ReLU-Conv $(1 \times 1 \times 1)$-BN-ReLU-Conv$(3 \times 3 \times 3)$ is very effective for the 3D-DenseNet.

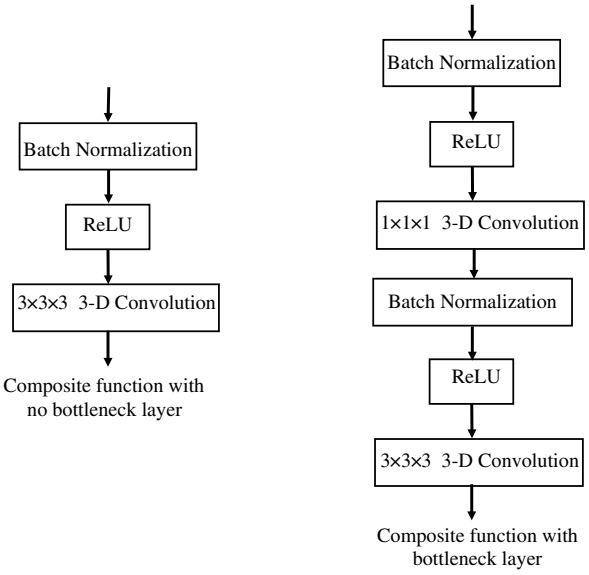

Fig. 2 Two kinds of composite functions (the right includes a bottleneck layer, and the left does not).

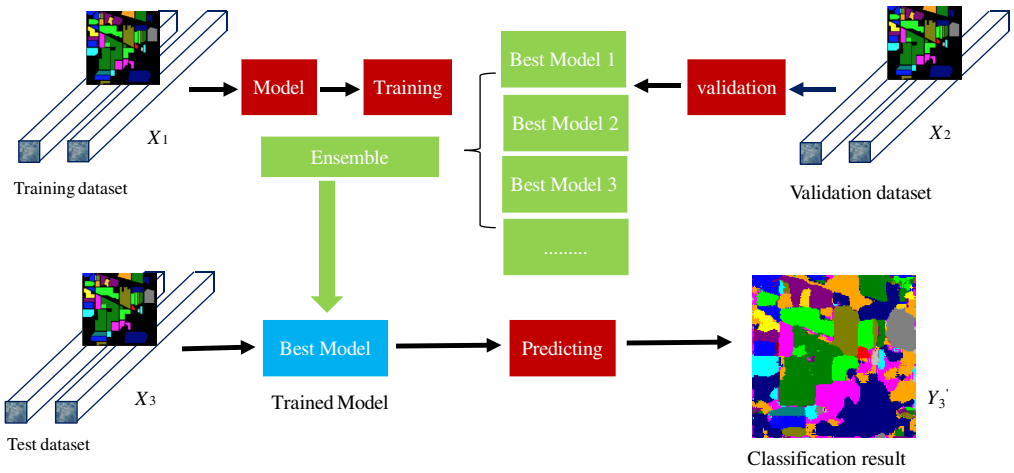

Fig. 3 Feature extraction framework.

2.3 *Framework of 3D-DenseNet for HSI Classification*

The end-to-end deep learning framework for HSI classification used in this paper is shown in Fig. 3. In this framework, all sample data are divided into three groups: training dataset, validation dataset, and test dataset. The dataset contains N labeled pixels, such as P_1, P_2, and P_3, which correspond to the C_1, C_2, and C_3 tag vectors after one-hot encoding.[27] To make full use of the spectral–spatial information provided by HSIs, the model uses neighboring pixel blocks of size $M \times N \times L$ from the raw data as input. $M \times N$ is the spatial sample size, and L is the spectral dimension of the input image. The goal of the training is to update the parameters of the model until the model can predict the category Y_3' for the given data X_3 with high accuracy.

After determining the structure and depth of the model, the training dataset X_1 are used to train the model. The validation dataset X_2 monitors the training process by testing the classification accuracy and the loss of the temporary model generated during training. The network with the highest classification accuracy is selected from the validation process, and the corresponding weight parameters are retained. During the test, we integrate the multiple best retained models and calculate the average and standard deviation of the multiple datasets of overall accuracies, average accuracies, and kappa coefficients. In the experiment, we reserved the optimal set of parameters according to loss and accuracy. We then averaged the values of the final three sets

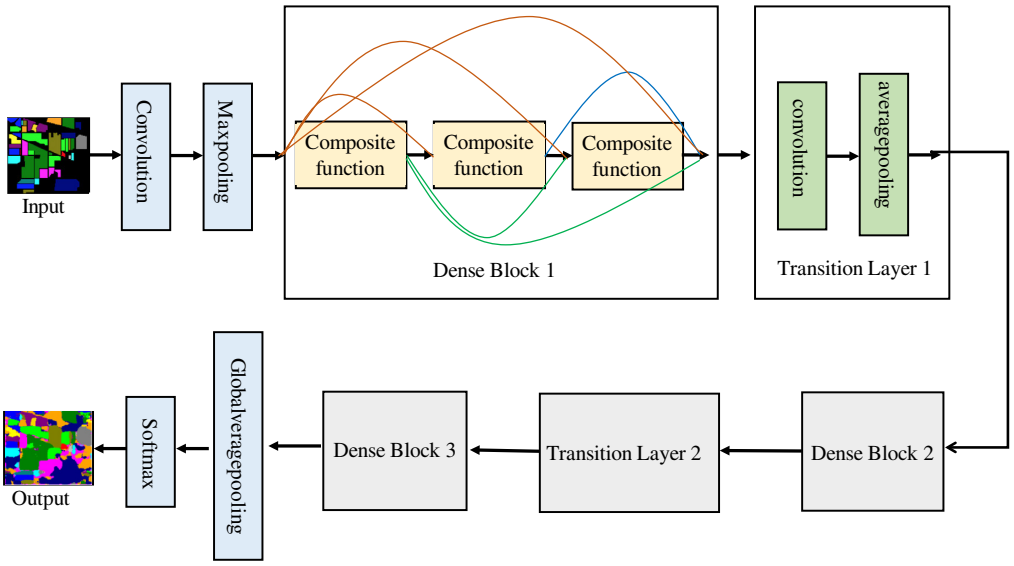

Fig. 4 The 3D-DenseNet model flowchart.

of best models and calculated the standard deviation to obtain more reliable results for evaluating the stability of the model. Finally, the test dataset X_3 was used to evaluate the classification performance of the trained model.

The model uses neighboring pixel block sampled in the HSI dataset as input data, which refines the sampling space of the input to obtain high accuracy. In the experiment, we create two different versions of the 3D-DenseNet model: 3D-DenseNet and 3D-DenseNet-BC (Fig. 4). For each model, there are three blocks and two transition layers. In each block, we investigated the optimal number of composite functions and finally chose to set 3 or 12 composite functions in each layer. The translation layer is placed between two dense blocks, including a BN, a ReLU, a $1 \times 1 \times 1$ convolutional layer, and an averagepooling3D operation. For the 3D-DenseNet (Table 1), the model does not contain the bottleneck layer, and the compression is set to 0.5. For the 3D-DenseNet-BC (Table 2), the model contains a bottleneck layer, and

Table 1 The 3D-DenseNet architecture (e.g., $M = 11$, $N = 11$, and $L = 200$).

Layers	Output size	3D-DenseNet ($k = 32$)	3D-DenseNet($k = 32$)
3-D convolution	$11 \times 11 \times 200$	$3 \times 3 \times 3$ conv with stride 1	
3-D maxpooling	$5 \times 5 \times 99$	$3 \times 3 \times 3$ maxpool with stride 2	
Dense block1	$5 \times 5 \times 99$	$[3 \times 3 \times 3$ conv$] \times 3$	$[3 \times 3 \times 3$ conv$] \times 12$
Transition layer1	$5 \times 5 \times 99$	$1 \times 1 \times 1$ conv	
	$3 \times 3 \times 50$	Average pool with stride 2	
Dense block2	$3 \times 3 \times 50$	$[3 \times 3 \times 3$ conv$] \times 3$	$[3 \times 3 \times 3$ conv$] \times 12$
Transition layer2	$3 \times 3 \times 50$	$1 \times 1 \times 1$ conv	
	$2 \times 2 \times 25$	$1 \times 1 \times 1$ average pool with stride 2	
Dense block3	$2 \times 2 \times 25$	$[3 \times 3 \times 3$ conv$] \times 3$	$[3 \times 3 \times 3$ conv$] \times 12$
Classification layer		Globalaveragepooling3D	
		Classification and softmax	

Table 2 The 3D-DenseNet-BC architecture (e.g., $M = 11$, $N = 11$, $L = 200$).

Layers	Output size	3D-DenseNet-BC ($k = 32$)	3D-DenseNet-BC ($k = 32$)
3-D convolution	$11 \times 11 \times 200$	$3 \times 3 \times 3$ conv with stride 1	
3-D maxpooling	$5 \times 5 \times 99$	$3 \times 3 \times 3$ maxpool with stride 2	
Dense block1	$5 \times 5 \times 99$	$\begin{bmatrix} 1 \times 1 \times 1 \\ 3 \times 3 \times 3 \end{bmatrix}$ conv $\times 3$	$\begin{bmatrix} 1 \times 1 \times 1 \\ 3 \times 3 \times 3 \end{bmatrix}$ conv $\times 12$
Transition layer1	$5 \times 5 \times 99$	$1 \times 1 \times 1$ conv	
	$3 \times 3 \times 50$	Average pool with stride 2	
Dense block2	$3 \times 3 \times 50$	$\begin{bmatrix} 1 \times 1 \times 1 \\ 3 \times 3 \times 3 \end{bmatrix}$ conv $\times 3$	$\begin{bmatrix} 1 \times 1 \times 1 \\ 3 \times 3 \times 3 \end{bmatrix}$ conv $\times 12$
Transition layer2	$3 \times 3 \times 50$	$1 \times 1 \times 1$ conv	
	$2 \times 2 \times 25$	Average pool with stride 2	
Dense block3	$2 \times 2 \times 25$	$\begin{bmatrix} 1 \times 1 \times 1 \\ 3 \times 3 \times 3 \end{bmatrix}$ conv $\times 3$	$\begin{bmatrix} 1 \times 1 \times 1 \\ 3 \times 3 \times 3 \end{bmatrix}$ conv $\times 12$
Classification layer		Globalaveragepooling3D	
		Classification and softmax	

the compression is 0.5. Compared with 3D-DenseNet, the bottleneck layer is more conducive to avoiding overfitting and reducing training parameters. The last layer is a softmax classifier for mapping the extracted features to the corresponding category probability values.

3 Experimental Datasets

To evaluate the effect of the presented 3D-DenseNet, the two most representative HSI datasets were used: the Indian pines dataset and the Pavia University dataset. Classification assessment uses overall accuracy (OA), average accuracy (AA), and the kappa coefficient. OA is the number of the correctly classified test samples divided by the total number of test samples, AA is the average rate of correctness for each class, whereas kappa (K) combines the diagonal and off-diagonal terms of the confusion matrix and is a robust measure of consistency. For the results of 3D-DenseNet-BC, we also discussed the recall and precision.

3.1 *Datasets*

3.1.1 *Indian Pines dataset*

The Indian Pines dataset was collected in June 1992 by the AVIRIS spectral imager at the Indiana Pine Forest Experimental Area in Northwest Indiana. Its size is 145×145 pixels with spatial resolution of 20 m, and there are 220 bands in the wavelength range of 0.4 to 2.5 μm. This experiment removed 20 water vapor absorption and low signal-to-noise ratio bands, and retained the other 200 bands for experiments. The dataset includes 16 types of geographic objects, such as grassland, architecture, and crops. Its false color image and ground-truth labels are shown in Fig. 5.

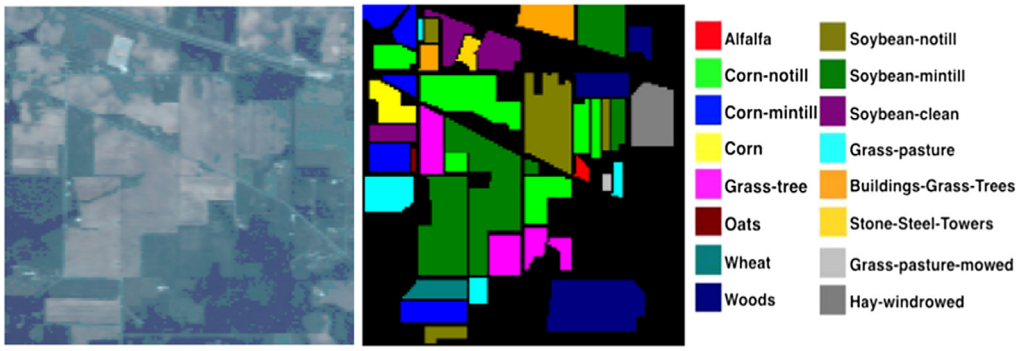

Fig. 5 False color image and ground-truth labels of Indian Pines.

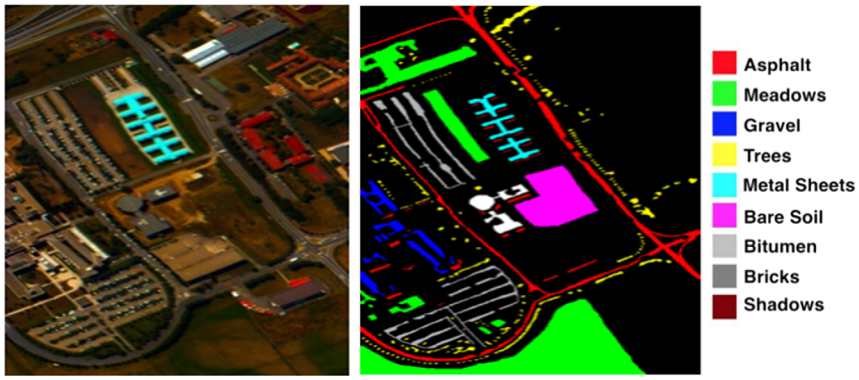

Fig. 6 False color image and ground-truth labels of Pavia University dataset.

3.1.2 *Pavia University dataset*

The Pavia University dataset was collected by the ROSIS spectral imager in 2001 in the Pavia region of northern Italy. The image contains 610×340 pixels with a spatial resolution of 1.3 m and 115 bands with the wavelength range of 0.43 to 0.86 μm. The experiment removed 12 bands containing strong noise and water vapor absorption, leaving the other 103 bands for our experiments. The dataset includes nine types of geographic objects, such as roads, trees, and roofs. Its false color image and ground-truth labels are shown in Fig. 6.

3.2 *Data Preprocessing*

For the Indian Pines and Pavia University dataset, each type of geographic objects is randomly shuffled to ensure the distribution of the samples. The two hyperspectral image datasets were normalized to values with unit variance, which reduces the computational effort of the model and hence speeds up model calculations. For the two dataset, the results (Fig. 7) of noise processing methods, including mean filtering, Gaussian filtering, and median filtering were compared. Intuitively, median filtering is the best to deal with noise. Taking the Indian Pines dataset for example, we used 3D-DenseNet-BC as the base model, where the number of composite functions $c = 3$, the growth rate $k = 32$, neighboring pixel block size is 11, and training dataset ratio is 2:1:7.[19] However, from the figure of the accuracy and loss of training and validation (Fig. 8), the Gaussian filter produced a smoother curve and higher classification accuracy.

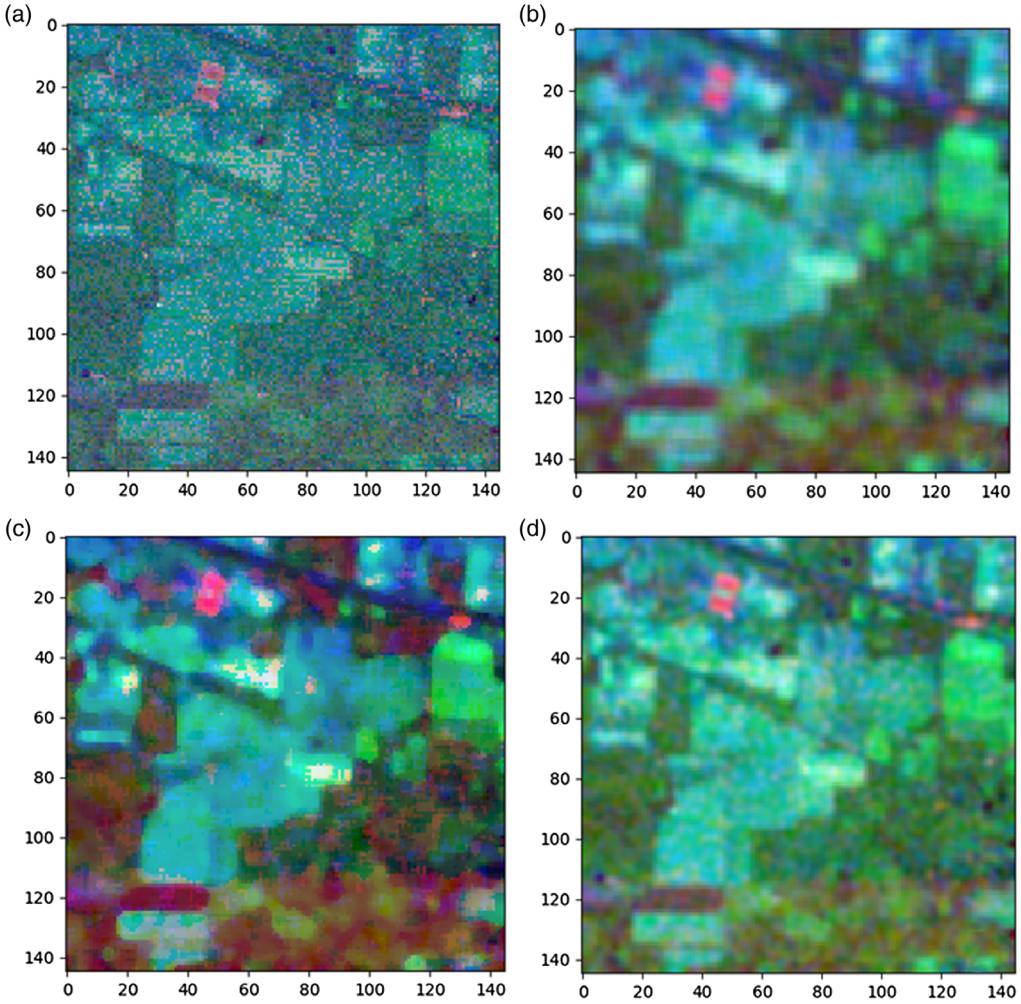

Fig. 7 The results of noise process methods: (a) origin filtering, (b) mean filtering, (c) Gaussian filtering, and (d) median filtering.

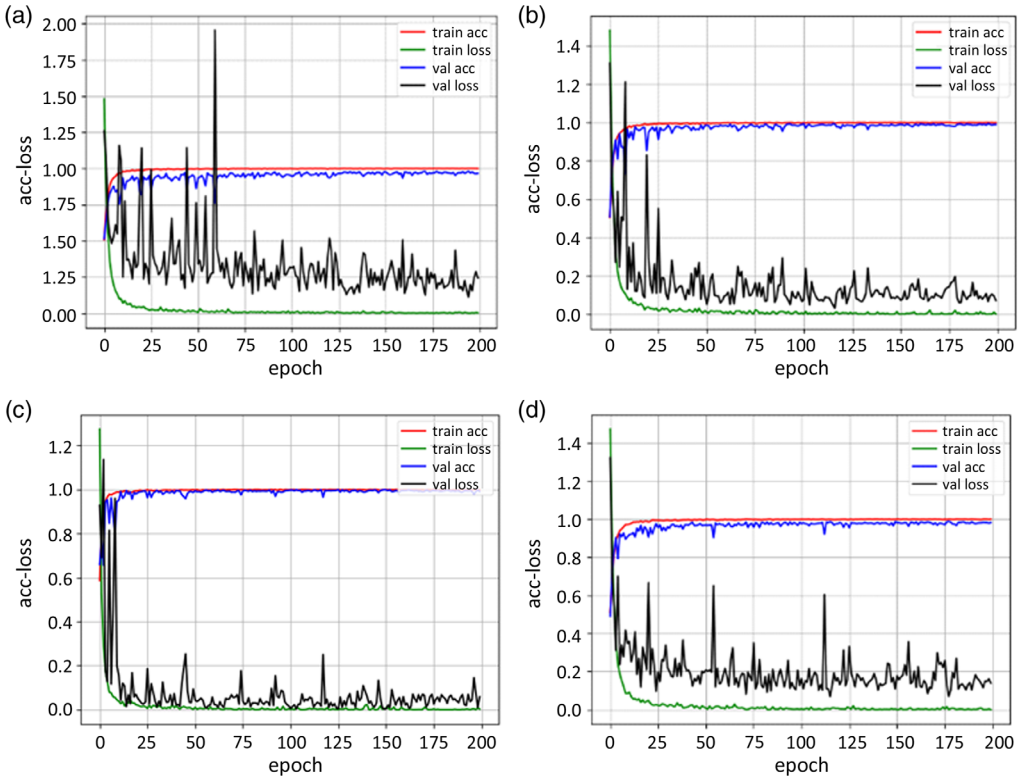

Fig. 8 Loss and accuracy changes on training and validation datasets on Indian Pines dataset: (a) origin, (b) mean filtering, (c) Gaussian filtering, and (d) median filtering.

4 Experimental Parameter Discussion

For the 3D-DenseNet for HSI classifications, the training and weight parameters need to be configured and updated. This section will focus on the four factors that control the training process and influence the classification results, including the neighboring pixel block size, the number of composite functions, the growth rate, and the ratio of the training dataset. Using a batch_size of 16, the RMSprop optimizer optimizes the loss function. During the training process, the best model with the highest classification performance on the validation dataset is retained, and the test dataset is evaluated by the optimal model. The experimental hardware platform uses Intel i5-8500k for the CPU, GTX1080Ti for the GPU, and 16g for the memory. In Secs. 4.1 and 4.2, we refer to Ref. 19 for identifying the ratio of training dataset, validation dataset, and test dataset to evaluate the effects of neighboring pixel block size, the number of composite functions, and the growth rate. For the Indian Pines dataset, the ratios are 2:1:7, whereas for the Pavia University dataset, the ratios are 1:1:8.

4.1 *Effect of Neighboring Pixel Block Size*

The network fills the edge of the input $145 \times 145 \times 200$ image into $155 \times 155 \times 200$ images (Indian Pines as an example). A neighboring pixel block of $M \times N \times L$ was sequentially selected on the $155 \times 155 \times 200$ image, where $M \times N$ is the spatial sampling size, and L is the full-dimensional spectrum. If the original image is too large, the running speed will be slow, the short-term memory consumption will rise, and the hardware platform will be very demanding. Therefore, the mechanism of inputting neighboring pixel blocks is adopted. Generally, if the range of neighboring pixel blocks is too small, the receptive field for the feature extraction is likely to be insufficient, and a good effect is not obtained locally.

3D-DenseNet-BC ($c = 3$, $k = 32$) is effective in reducing parameters, so it is used as a base model to evaluate the effect of neighboring pixel block size. On the Indian Pines dataset (Table 3), the pixel block size ranges from 9 to 15, the accuracy is significantly improved,

Table 3 Training time, test time, and overall accuracy for different neighboring pixel block sizes on the Indian Pines dataset.

Neighboring pixel block ($M = N$)	Training time	Test time	OA
9	1913.79	6.95	98.27
11	2062.07	11.08	98.68
13	2594.27	13.18	98.82
15	3421.97	19.76	99.44
17	4186.47	23.43	99.12
19	5445.92	29.91	99.27

Table 4 Training time, test time, and overall accuracy for different neighboring pixel block sizes on the Pavia University dataset.

Neighboring pixel block ($M = N$)	Training time	Test time	OA
9	2466.68	21.43	98.99
11	3090.87	28.64	99.35
13	3782.06	36.77	99.68
15	4807.01	47.62	99.44
17	5753.11	63.08	99.82
19	7067.33	91.06	99.87

with a shock from 15 to 19. Training and test time grow as the range of neighboring pixel blocks grows, so the neighboring pixel block size of 15 on the Indian Pines dataset is appropriate for training time, test time, and overall accuracy. This is also evident on the Pavia University dataset (Table 4). As the range of pixel blocks increases, the OA grows small and small, and there are also obvious threshold effects. 19 was selected as the neighboring pixel block size on the Pavia University dataset.

4.2 *Effect of Composite Functions and Different Growth Rates*

For the 3D-DenseNet, the model parameters mainly depend on the depth of the model. The deeper the model, the greater the possibility of high accuracy,[28] but the increase of the parameter quantity leads to an increase in both the training time and the required computing power. Therefore, we evaluated the 3D-DenseNet's accuracy, parameter quantities, training time, and test time. The most direct indicator of the depth and parameter quantities in the model is the number of composite functions and the growth rates. In the experiment, the composite function was chosen to be 3, 6, and 12 to test the effects of the model depth, respectively. Considering the lack of samples in HSI dataset, when using 12 composite functions in each dense block, the model is already deep. In the choice of growth rate, considering the three groups of 12, 24, and 32, k determines the size of the network. If its value is too large, the model grows too fast. The neighboring pixel block size of 15 was selected on the Indian Pines.

We evaluated the performance of the 3D-DenseNet and the 3D-DenseNet-BC with different composite functions and different growth rates in the Indian Pines dataset. We experimented with $k = 32$ and the composite functions of 3, 6, and 12 on 3D-DenseNet and 3D-DenseNet-BC (Table 5). The compression factor is 0.5. The experiment found that the OA on 3D-DenseNet-BC is generally higher than that on 3D-DenseNet, whereas the parameter quantity, training time, and test time are lower than 3D-DenseNet. In terms of accuracy, although the

Table 5 Number of composite functions, params, training time, test time, and OA values for 3D-DenseNet on Indian Pines dataset.

Method	Number of composite functions	Growth rate (k)	Params	Training time	Test Time	OA
3D-DenseNet	3	32	898,288	2423.70	12.97	98.69
3D-DenseNet	6	32	3,176,848	5317.92	26.71	99.08
3D-DenseNet	12	32	11,977,936	15071.37	74.25	99.27
3D-DenseNet-BC	3	32	1,164,208	3421.97	19.76	99.44
3D-DenseNet-BC	6	32	2,562,448	6538.60	33.25	99.30
3D-DenseNet-BC	12	32	6,076,048	14418.09	71.77	99.61

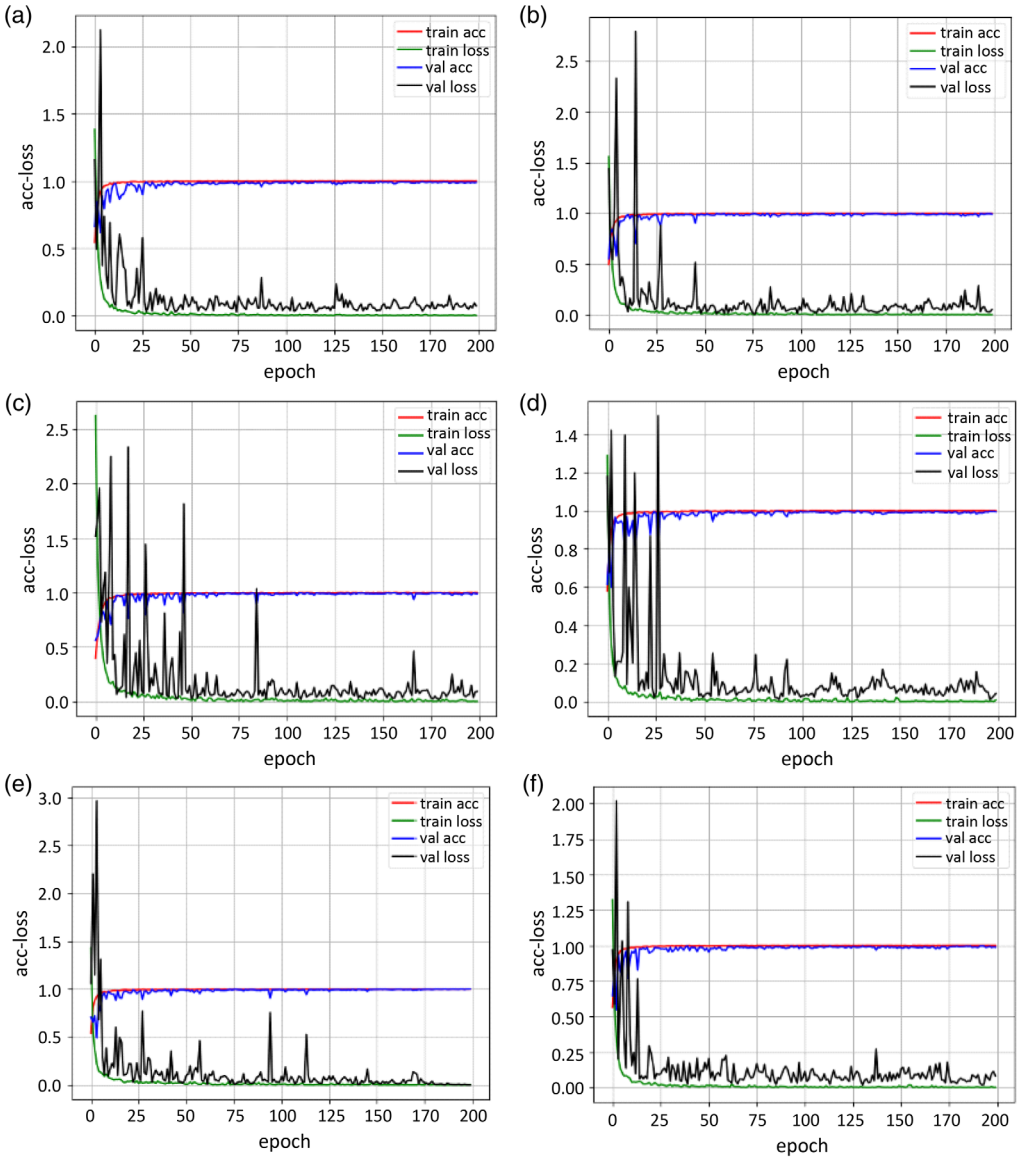

Fig. 9 Loss and accuracy changes on training and validation datasets on Indian Pines dataset: (a) 3D-DenseNet ($c = 3$, $k = 32$), (b) 3D-DenseNet ($c = 6$, $k = 32$), (c) 3D-DenseNet ($c = 12$, $k = 32$), (d) 3D-DenseNet-BC ($c = 3$, $k = 32$), (e) 3D-DenseNet-BC ($c = 6$, $k = 32$), and (f) 3D-DenseNet-BC ($c = 12$, $k = 32$).

Table 6 Growth rate (k), params, training time, test time, and OA values for 3D-DenseNet on Indian Pines dataset.

Method	Number of composite functions	Growth rate (k)	Params	Training time	Test time	OA
3D-DenseNet	3	12	202,426	1986.16	10.99	99.06
3D-DenseNet	3	24	560,176	2248.55	12.23	99.27
3D-DenseNet	3	32	888,496	2423.70	12.97	98.69
3D-DenseNet	12	12	1,988,944	7904.31	40.45	98.97
3D-DenseNet	12	24	6,974,224	12,106.17	61.68	99.11
3D-DenseNet	12	32	11,968,144	15,071.37	74.25	99.27
3D-DenseNet-BC	3	12	184,318	2301.51	12.04	98.71
3D-DenseNet-BC	3	24	669,760	2976.34	15.27	98.72
3D-DenseNet-BC	3	32	1,164,208	3421.97	19.76	99.44
3D-DenseNet-BC	12	12	932,848	7781.09	39.03	99.23
3D-DenseNet-BC	12	24	3,477,904	11,717.97	58.50	99.53
3D-DenseNet-BC	12	32	6,076,048	14,418.09	71.77	99.61

models with parameters $c = 12$ and $k = 32$ all achieved the highest OA, the training, and test time increased rapidly with the increase of the parameter amount. Therefore, the model with $c = 3$, although losing some accuracy, achieves faster training speed and saves computational power, whereas the model with $c = 12$ achieves higher precision. It has a good performance in different application scenarios. The loss and accuracy changes on the training and validation datasets are shown in Fig. 9.

In Table 6, we chosen two cases, $c = 3$ and $c = 12$, to discuss the effect of k value. As k increases, the OA rises. In 3D-DenseNet, when $c = 3$, $k = 24$, and $c = 12$, $k = 32$ is the highest OA that reaches 99.27%. However, $c = 3$, $k = 24$ has a smaller amount of parameters and less training and testing time. From the accuracy and loss changes of training and validation (Figs. 10 and 11), $c = 3$, $k = 24$ are also more stable. In 3D-DenseNet-BC, with the increase of k, the accuracy improvement is more obvious. When $c = 12$, $k = 32$, the number of network parameters is the largest, but the accuracy is also the highest. Compared with $c = 12$, $k = 32$, when $c = 3$, $k = 32$, the number of the network parameter is only $1/6$ of the former, but the accuracy is lost only by 0.17, and the training and test time is shorter. It can be seen from the accuracy and loss changes of training and validation, the latter is more stable and less fluctuating.

4.3 *Effect of Different Ratios of Training Datasets*

In order to consider the impact of small data volumes on model training, we discussed the ratios of the training, validation, and test dataset. We know that the 3D-DenseNet-BC is effective in reducing parameters and improving OA, when $c = 3$, $k = 32$, neighboring pixel block sizes are 15 and 19 for the Indian Pines and Pavia University dataset, respectively. Therefore, we used this model for experiments. The training time, test time, and accuracy changes are shown in Tables 7 and 8 for different training dataset ratios. The loss and accuracy changes on the training and validation datasets are shown in Figs. 12 and 13.

The Indian Pines dataset is small, a larger training dataset scale was used to improve accuracy, and the training process is more stable. For the Indian Pines dataset, we finally chose the ratio of 5:1:4. The Pavia University dataset is a relatively large dataset, thus the training dataset need not to be too large for achieving high accuracy. Increasing the number of training datasets

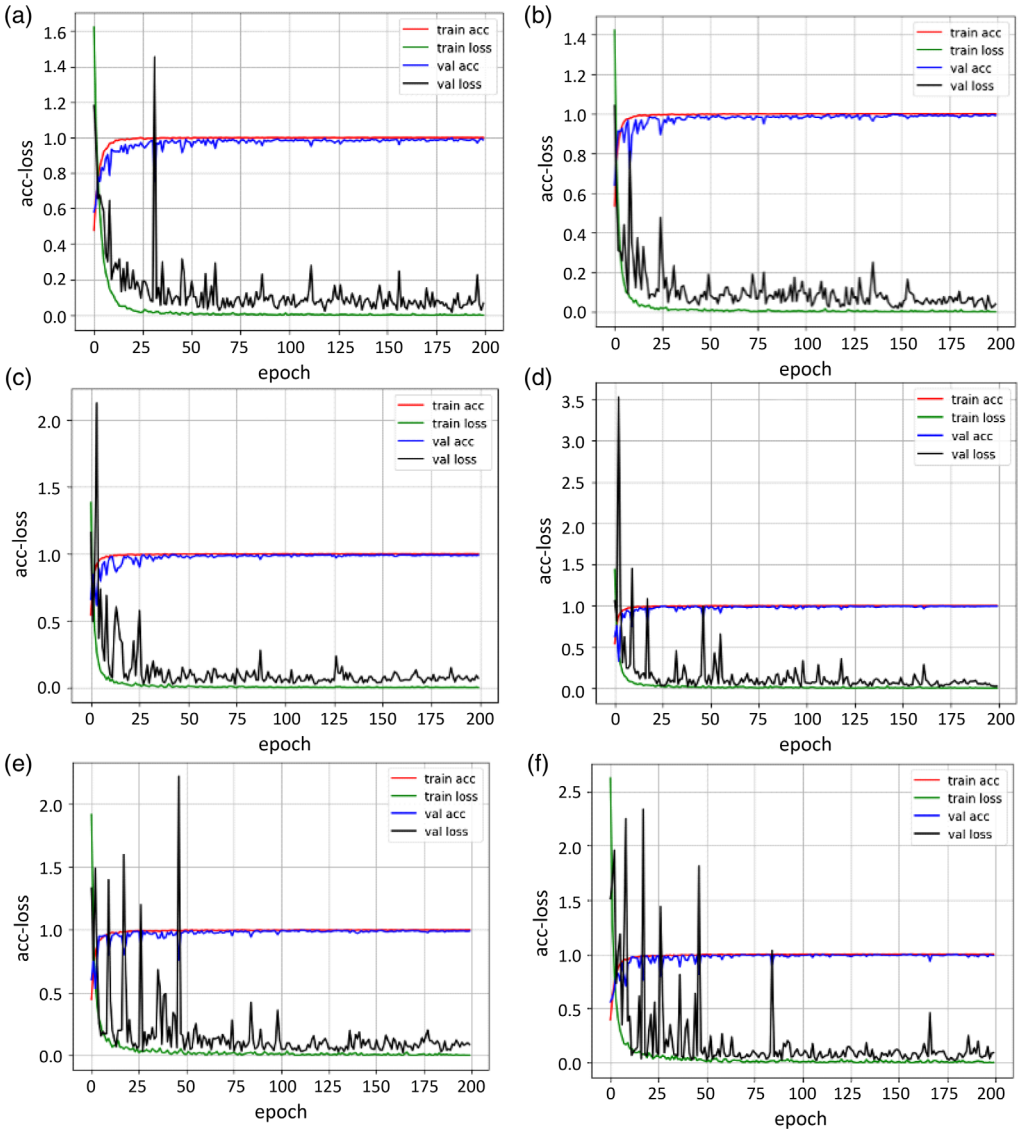

Fig. 10 Loss and accuracy changes on training and validation datasets on Indian Pines dataset: (a) 3D-DenseNet ($c = 3$, $k = 12$), (b) 3D-DenseNet ($c = 3$, $k = 24$), (c) 3D-DenseNet ($c = 3$, $k = 32$), (d) 3D-DenseNet ($c = 12$, $k = 12$), (e) 3D-DenseNet ($c = 12$, $k = 24$), and (f) 3D-DenseNet ($c = 12$, $k = 32$).

has little effects on the overall accuracy, the training and test time will rise rapidly, and the running cost will increase. For the Pavia University dataset, we finally chose a ratio of 3:1:6.

5 Results

5.1 *Results of the Indian Pines Dataset*

In Indian Pines dataset, we chose the 3D-DenseNet-BC ($c = 3$, $k = 32$) and the 3D-DenseNet-BC ($c = 12$, $k = 32$) for comparison. The accuracy of the 3D-DenseNet-BC ($c = 3$, $k = 32$) is lower than that of 3D-DenseNet-BC ($c = 12$, $k = 32$), but the actual accuracy of multigroup average is not very different (Table 5). The training and test time and parameter amount of 3D-DenseNet-BC ($c = 3$, $k = 32$) are much lower than 3D-DenseNet-BC ($c = 12$, $k = 32$). Therefore, from the perspective of the training time and accuracy of the model, we have chosen these two 3D-DenseNet-BC models. To evaluate the model, we compared the SAE,[8] 3D-CNN,[15] and SSRN[16] models, respectively (Table 9). SAE is an unsupervised learning method and excels

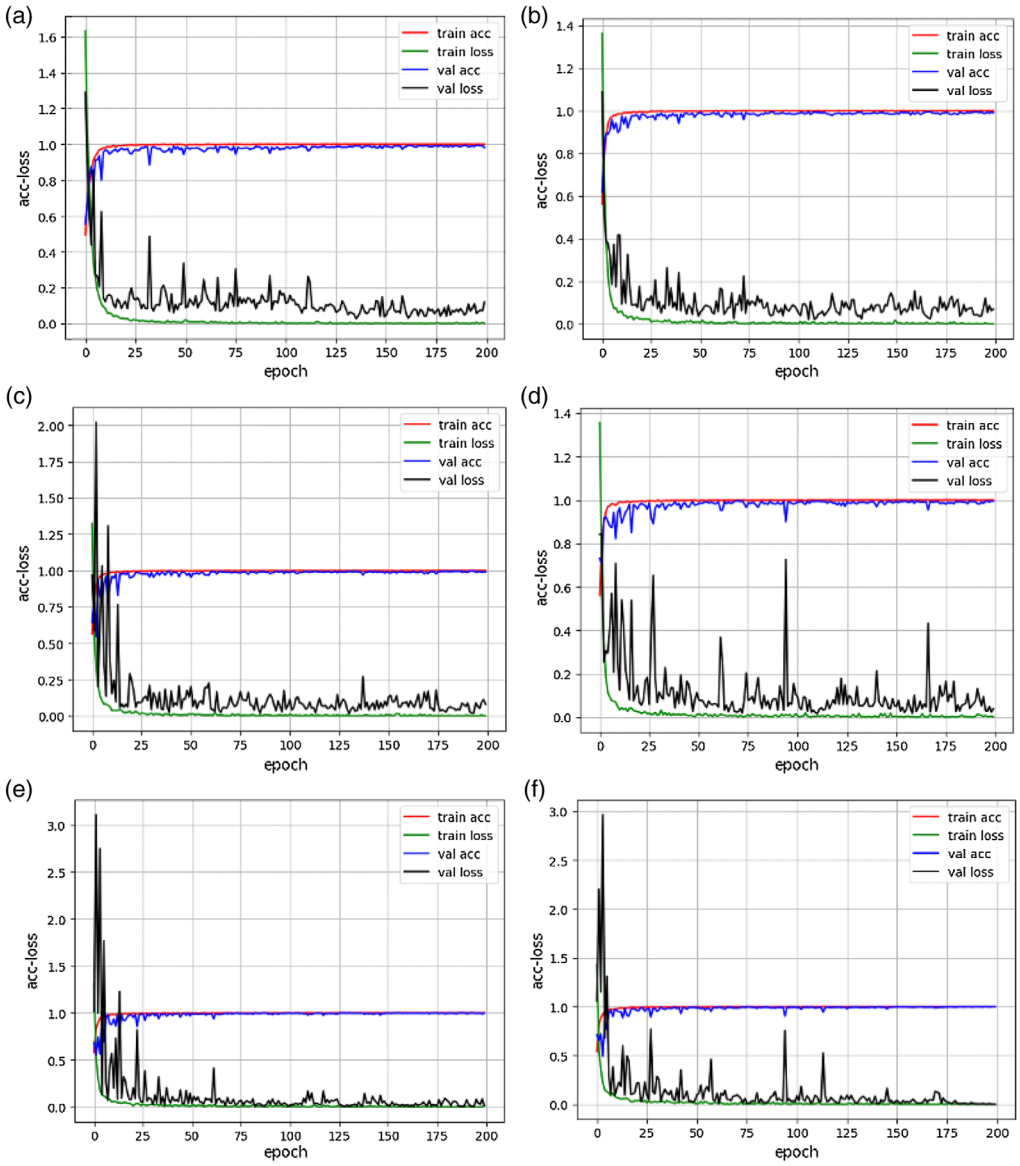

Fig. 11 Loss and accuracy changes on training and validation datasets on Indian Pines dataset: (a) 3D-DenseNet-BC ($c = 3$, $k = 12$), (b) 3D-DenseNet-BC ($c = 3$, $k = 24$), (c) 3D-DenseNet-BC ($c = 3$, $k = 32$), (d) 3D-DenseNet-BC ($c = 12$, $k = 12$), (e) 3D-DenseNet-BC ($c = 12$, $k = 24$), and (f) 3D-DenseNet-BC ($c = 12$, $k = 32$).

Table 7 Training time, test time, and OA under different training dataset ratios on the Indian Pines dataset.

Ratios	Training time	Test time	OA
1:1:8	1971.96	20.07	96.20
2:1:7	3421.97	19.76	99.44
3:1:6	4989.28	23.41	99.81
4:1:5	6284.01	12.95	99.80
5:1:4	7743.69	10.36	99.92

Table 8 Training time, test time, and OA under different training dataset ratios on Pavia University dataset.

Ratios	Training time	Test time	OA
1:1:8	7067.33	91.06	99.87
2:1:7	12,380.97	80.75	99.88
3:1:6	17,638.48	67.74	99.89
4:1:5	22,903.57	57.49	99.91
5:1:4	28,169.03	66.22	99.93

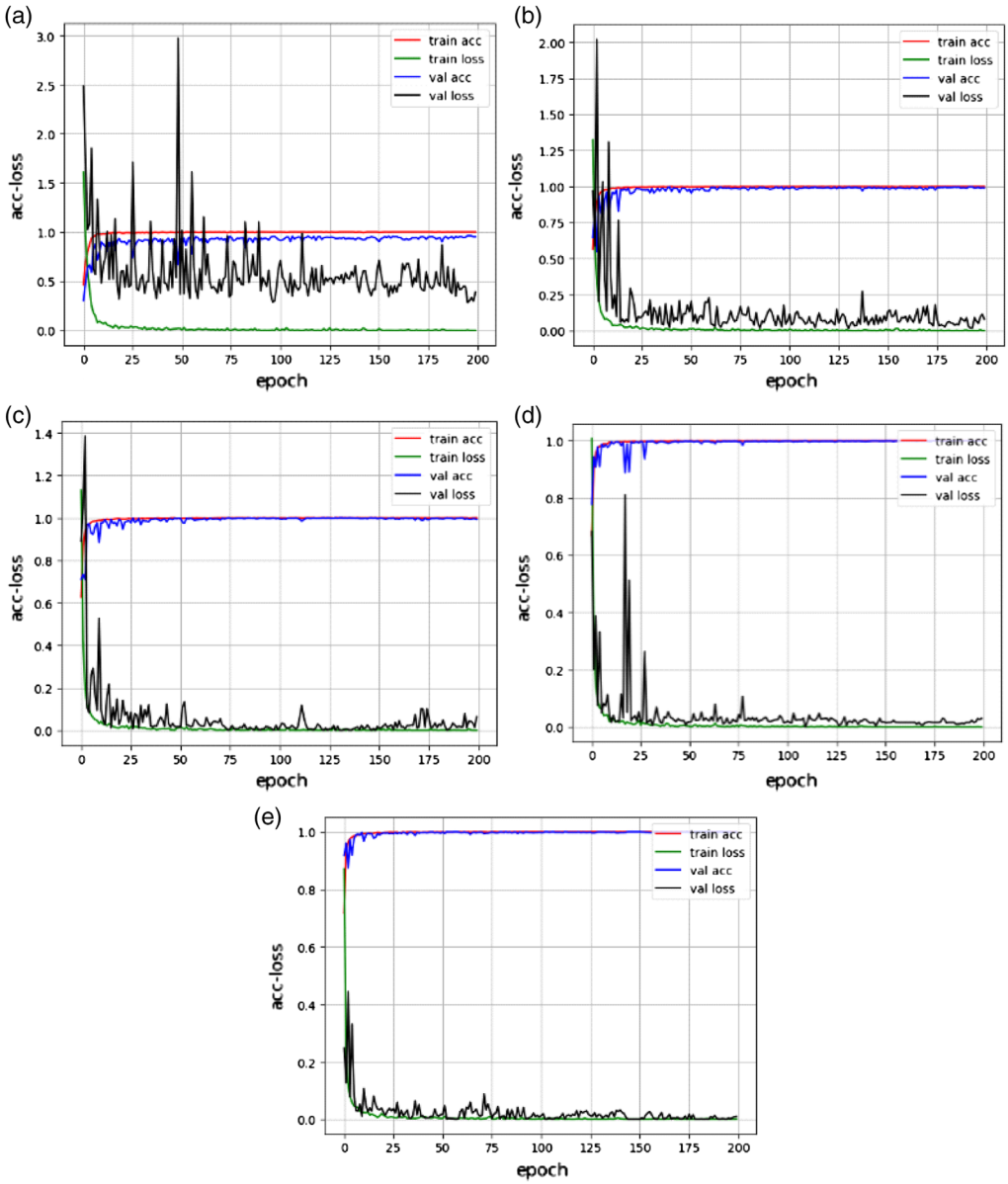

Fig. 12 Loss and accuracy changes on training and validation datasets under different training dataset ratios on Indian Pines dataset: (a) 1:1:8, (b) 2:1:7, (c) 3:1:6, (d) 4:1:5, and (e) 5:1:4.

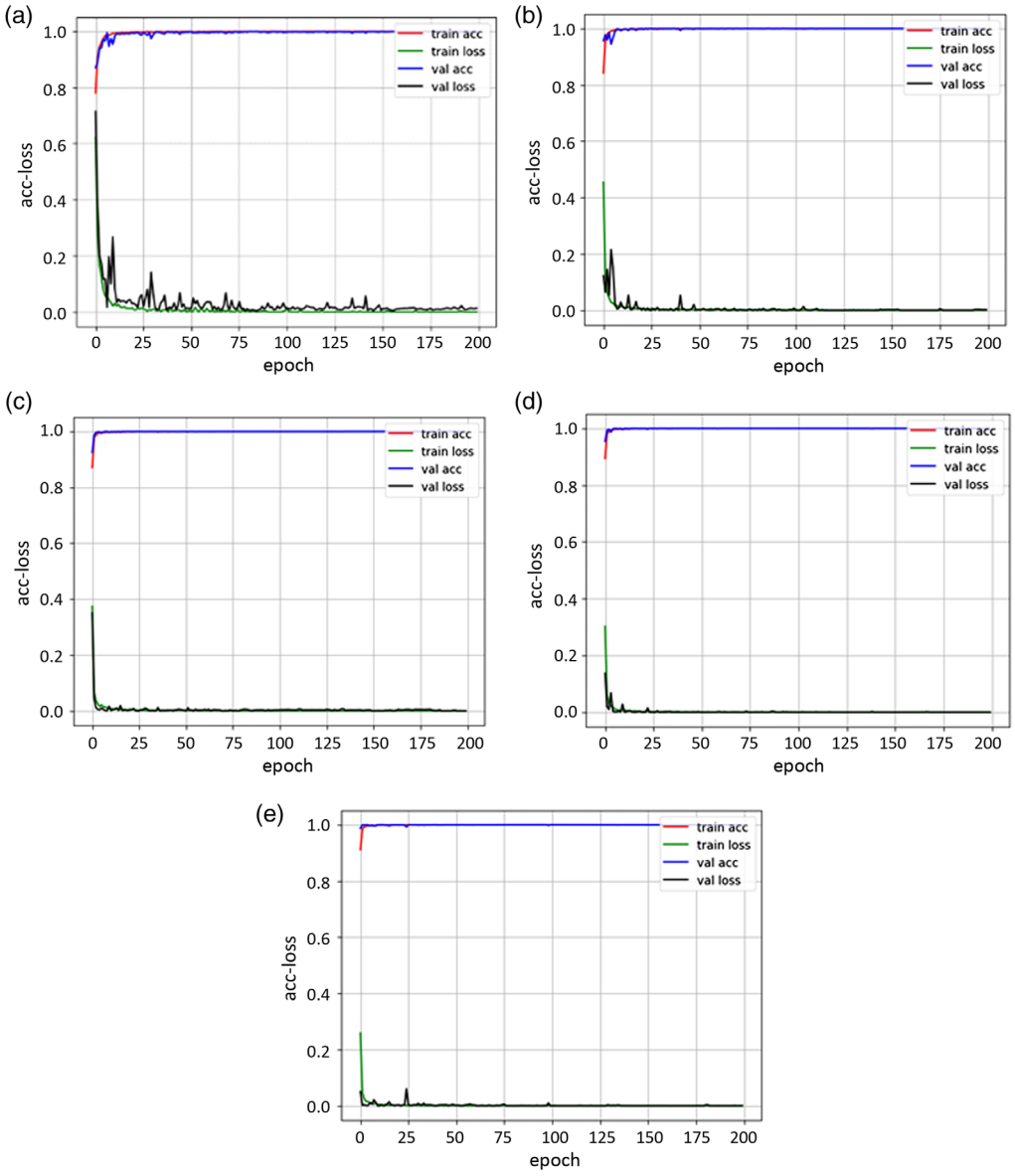

Fig. 13 Loss and accuracy changes on training and validation datasets under different train dataset ratios on Pavia University dataset: (a) 1:1:8, (b) 2:1:7, (c) 3:1:6, (d) 4:1:5, and (e) 5:1:4.

in processing small-scale HSIs, and the 3D-CNN is the current state-of-the-art method, which uses the same input and processing with the 3D-DenseNet-BC. Based on 3D-CNN, SSRN is based on the supervised spatial–spectral residual network and uses spatial and spectral residual module to extract spatial and spectral information, respectively, which is a powerful extension of the residual structure on 3D-CNN.

The OA, AA, and kappa of the 3D-DenseNet-BC ($c = 3$, $k = 32$) and 3D-DenseNet-BC ($c = 12$, $k = 32$) are higher than that of the above three methods, and the OA accuracy is 99.81% and 99.85%, respectively. The recall and precision of 3D-DenseNet-BC ($c = 3$, $k = 32$) are both 99.94%, and the recall and precision of 3D-DenseNet-BC ($c = 12$, $k = 32$) are 99.92% and 99.91%. The classification accuracy of SSRN on class 9 oats is 0, but it is 100% on 3D-DenseNet-BC ($c = 3$, $k = 32$) and 3D-DenseNet-BC ($c = 12$, $k = 32$). The training dataset of oats is extremely small, and SSRN is unstable in HSI classification. Especially, it is not reliable for the identification of small HSI samples. Figure 14 shows the classification results of the five methods for the Indian Pines dataset. The classification results based on SAE

Table 9 Comparison of classification accuracy (%) of different methods on the Indian Pines dataset.

No	SAE	3D-CNN	SSRN	3-D-DenseNet-BC ($c = 3$, $k = 32$)	3-D-DenseNet-BC ($c = 12$, $k = 32$)
1	81.82	96.88	100	100	100
2	82.16	98.02	99.85	99.44	99.40
3	77.54	97.74	99.83	99.77	99.48
4	68.11	96.89	100	99.17	100
5	94.36	99.12	99.78	99.58	100
6	94.45	99.41	99.81	100	100
7	94.70	88.89	100	100	100
8	94.36	100	100	100	100
9	82.56	100	0	100	100
10	81.28	100	100	100	99.85
11	84.47	99.33	99.62	100	99.53
12	83.77	97.67	99.17	99.32	98.58
13	96.42	99.64	100	100	100
14	92.27	99.65	98.87	100	99.89
15	80.63	96.34	100	100	99.64
16	81.82	97.92	98.51	97.78	97.10
OA	85.47 ± 0.58	98.13 ± 0.12	99.62 ± 0.00	99.81 ± 0.01	99.85 ± 0.04
AA	86.31 ± 1.14	98.80 ± 0.11	93.46 ± 0.50	99.69 ± 0.08	99.71 ± 0.25
Kappa	83.42 ± 0.66	97.96 ± 0.53	99.57 ± 0.00	99.78 ± 0.01	99.84 ± 0.04

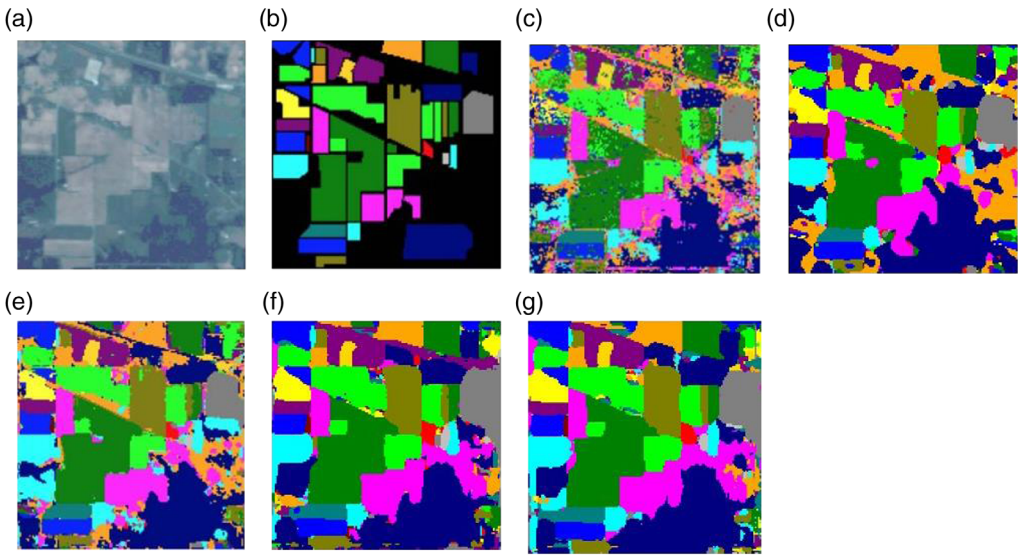

Fig. 14 Classification results of the best models for the Indian Pines dataset: (a) false color image, (b) ground-truth labels, (c)–(g) classification results of SAE, 3D-CNN, SSRN, 3D-DenseNet-BC ($c = 3$, $k = 32$), and 3D-DenseNet-BC ($c = 12$, $k = 32$), respectively.

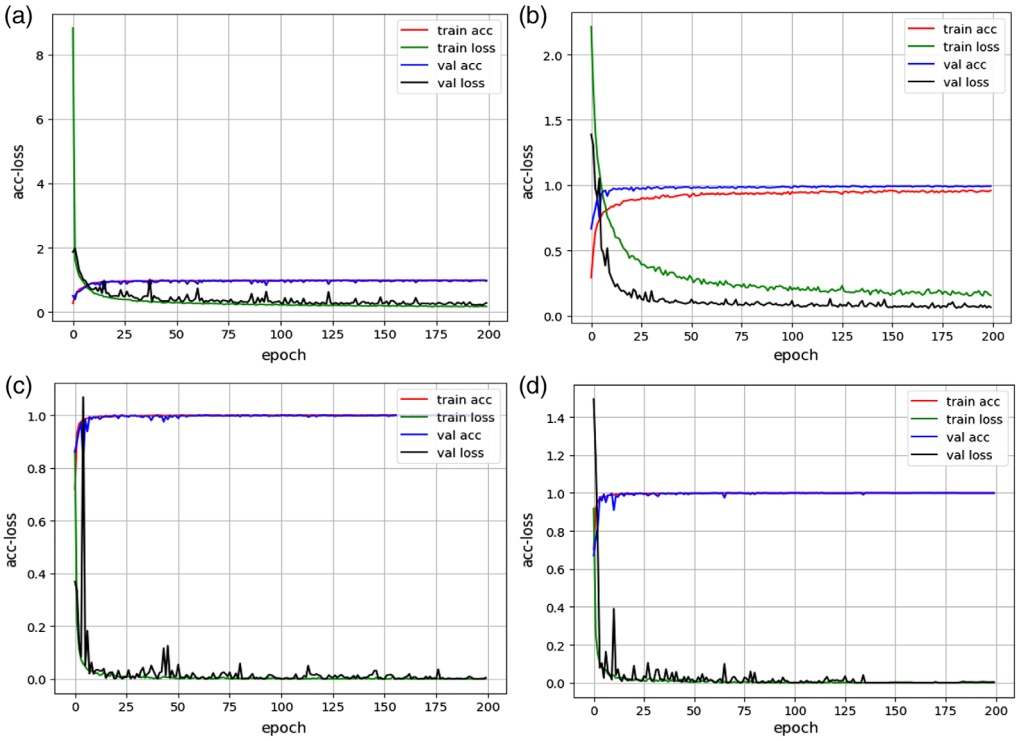

Fig. 15 Loss and accuracy changes on training and validation datasets on Indian Pines dataset: (a) 3D-CNN, (b) SSRN, (c) 3D-DenseNet-BC ($c = 3$, $k = 32$), and (d) 3D-DenseNet-BC ($c = 12$, $k = 32$).

and SSRN have obvious discrete isolated points, which greatly reduce the classification accuracy. 3D-CNN and 3D-DenseNet-BC can effectively remove these misclassification noises. Especially, in the classification results of 3D-DenseNet-BC, the edge contours of geographic objects are clearer.

We compared the loss and accuracy changes of the 3D-CNN, SSRN, 3D-DenseNet-BC ($c = 3$, $k = 32$), and 3D-DenseNet-BC ($c = 12$, $k = 32$) during training and validation (Fig. 15). The 3D-CNN used in this study consists of three CNN layers, two maxpooling layers, one full connected layer, and one softmax classifier. The network is not deep, but the number of parameters is very large. The initial loss of the model is also high. Compared with the 3D-CNN, the 3D-DenseNet-BC greatly reduces the number of parameters but provides a deeper network. The loss and accuracy curves of SSRN during training and validation are not good enough. In comparison, the 3D-DenseNet-BC has the best convergence, the training and validation loss and accuracy curves are almost coincident. The model has little fluctuation and has high stability. 3D-DenseNet-BC means fewer parameters, faster convergence, and deeper networks. It can reach the desired accuracy faster, save computing power and time, and improve the model efficiency.

5.2 *Results of the Pavia University Dataset*

In the Pavia University dataset, the input size is $19 \times 19 \times 103$ and the learning rate is 0.0003. The experiment also selected the SAE, 3D-CNN, SSRN, 3D-DenseNet-BC ($c = 3$, $k = 32$), 3D-DenseNet-BC ($c = 12$, $k = 32$) for comparison (Table 10). The 3D-DenseNet-BC ($c = 3$, $k = 32$) performed better on Pavia University dataset. Compared to the Indian Pines dataset, the Pavia University dataset has larger sample sizes and fewer classification categories. For deep feature extraction, samples are relatively abundant. The model may not to be particularly deep to achieve good classification accuracy. On the contrary, if the model is too deep, problems such as overfitting may occur.

On the Pavia University dataset, the training and validation loss and accuracy curves are shown in Fig. 16 for 3D-CNN, SSRN, 3D-DenseNet-BC ($c = 3$, $k = 32$), and 3D-

Table 10 Comparison of classification accuracy (%) of different methods on the Pavia University dataset.

No	SAE	3-D-CNN	SSRN	3-D-DenseNet-BC ($c = 3$, $k = 32$)	3-D-DenseNet-BC ($c = 12$, $k = 32$)
1	87.24	98.56	99.96	99.81	99.97
2	89.93	99.77	99.99	99.99	99.98
3	86.48	99.05	99.64	99.82	99.95
4	99.95	99.98	99.83	99.96	99.87
5	95.78	99.90	99.81	100	100
6	97.69	99.03	99.98	100	100
7	95.44	99.71	97.97	99.62	100
8	84.40	97.53	99.56	99.82	99.86
9	100	99.86	100	100	100
OA	90.58 ± 0.18	98.47 ± 0.41	99.79 ± 0.01	99.93 ± 0.002	99.94 ± 0.002
AA	92.99 ± 0.39	99.28 ± 0.31	99.75 ± 0.15	99.90 ± 0.004	99.95 ± 0.003
Kappa	87.21 ± 0.25	98.97 ± 0.21	99.87 ± 0.27	99.92 ± 0.003	99.95 ± 0.002

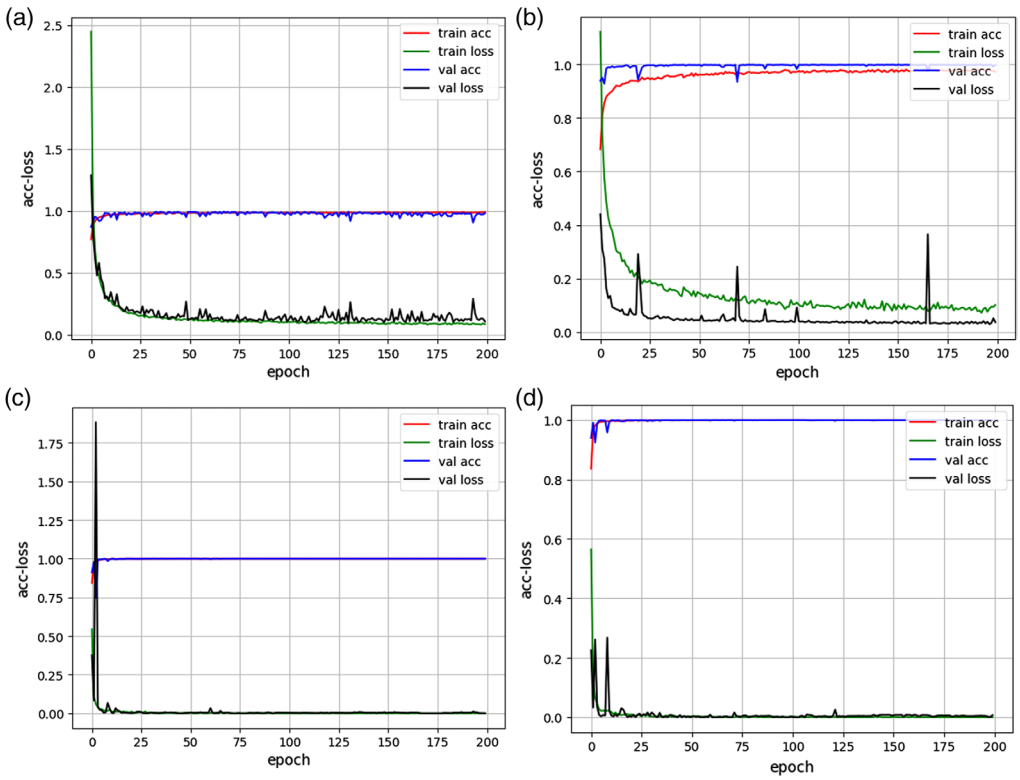

Fig. 16 Loss and accuracy changes on training and validation datasets on Pavia University dataset: (a) 3D-CNN, (b) SSRN, (c) 3D-DenseNet-BC ($c = 3$, $k = 32$), and (d) 3D-DenseNet-BC ($c = 12$, $k = 32$).

DenseNet-BC ($c = 12$, $k = 32$). SSRN convergence is incomplete and fluctuates up and down. The initial loss of 3D-DenseNet-BC ($c = 3$, $k = 32$) is large, but the model quickly converges and the model changes very stably. For 3D-DenseNet-BC ($c = 12$, $k = 32$), although the initial loss and accuracy are slightly fluctuating, the follow-up is still very stable.

5.3 *Discussion*

The experiments verify the effectiveness of the 3D-DenseNet in the HSI classification. Compared with traditional methods, deep learning can effectively simulate the human visual system through a hierarchical structure and automatical feature extraction. The end-to-end model framework can avoid the complex preprocessing and postprocessing of traditional methods for HSI classifications. Compared with other deep learning networks, the 3D-DenseNet achieves dense connection based on the dense blocks, increases information flow, and enhances feature reuse. The bottleneck layer and compression operation effectively reduce the number of parameters. This model can use the features effectively while completing a deeper network structure. It also makes the model easier to train. In remote sensing image classification, the 3D-DenseNet explores the potential of deeper neural networks. In a limited HSI dataset, we did not use data enhancement to further promote the 3D-DenseNet performance, even if the training dataset is not large. The 3D-DenseNet achieves high classification accuracy through deep feature extraction in HSIs. Deep networks have the effect of regularization and can effectively avoid overfitting. This is of great significance in the absence of HSI sample data.

6 Conclusion

This paper used the characteristics of the 3D-CNN to extract spatial and spectral information, combined with the dense connectivity of the deep network built in DenseNet. Compared with existing 3D-CNN, the 3D-DenseNet is deeper in structure and can learn more robust spectral–spatial features from HSIs. Through the densely connected structure, it enhances feature transmission, encourages feature reuse, and improves information flow in the network. Moreover, the deeper network structure has a regularized effect, which can effectively reduce overfitting on small sample datasets of HSIs. Experiment results show that the 3D-DenseNet performs well on the two benchmark datasets: the Indian Pines datasets and Pavia University datasets.

The 3D-DenseNet can be easily extended to other remote sensing image datasets because of its unusual structural design and deep feature learning capabilities. Deep learning model can automatically extract features from input data without feature engineering. The model needs a reasonable structural design. The configuration of hyperparameters depends on the number of training samples and the spatial size of each sample input. In the case of limited samples, the description dimension of the features may be limited, which will be overfitted. This paper overcomed this difficulty by discussing depth, enhancing feature reuse, and considering a large number of spectral signals and spatial semantics. The model uses end-to-end training and does not require complex preprocessing.

The future will still focus on the 3-D densely connected convolutional network to discuss the application of deep network in remote sensing image classification. From the perspective of feature extraction effectiveness, we can design a deep feature extraction model that combines effective feature aggregation methods such as dense connectivity and residual learning to deepen the 3D-CNN structure and accelerate information flow.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China under Grant Nos. 41671456 and 41401451.

References

1. C. I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*, Plenum Publishing Co., New York (2003).

2. P. J. Du et al., "Review of hyperspectral remote sensing image classification," *J. Remote Sens.* **20**(2), 236–256 (2016).
3. L. He et al., "Recent advances on spectral-spatial hyperspectral image classification: an overview and new guidelines," *IEEE Trans. Geosci. Remote Sens.* **56**(3), 1579–1597 (2018).
4. G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Trans. Inf. Theory* **14**(1), 55–63 (1968).
5. M. Fauvel et al., "Advances in spectral-spatial classification of hyperspectral images," *Proc. IEEE* **101**(3), 652–675 (2013).
6. P. Du, K. Tan, and X. Xing, "A novel binary tree support vector machine for hyperspectral remote sensing image classification," *Opt. Commun.* **285**(13), 3054–3060 (2012).
7. J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks* **61**, 85–117 (2015).
8. L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data a technical tutorial on the state of the art," *IEEE Geosci. Remote Sens. Mag.* **4**(2), 22–40 (2016).
9. Y. Xu et al., "Spectral–spatial unified networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.* **56**(10), 5893–5909 (2018).
10. Y. Xu et al., "Hyperspectral image classification via a random patches network," *ISPRS J. Photogramm. Remote Sens.* **142**, 344–357 (2018).
11. Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **8**(6), 2381–2392 (2015).
12. W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: a dimension reduction and deep learning approach," *IEEE Trans. Geosci. Remote Sens.* **54**(8), 4544–4554 (2016).
13. J. Yue et al., "Spectral–spatial classification of hyperspectral images using deep convolutional neural networks," *Remote Sens. Lett.* **6**(6), 468–477 (2015).
14. K. Makantasis et al., "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *IEEE Int. Geosci. and Remote Sens. Symp. (IGARSS)*, pp. 4959–4962 (2015).
15. Y. Chen et al., "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.* **54**(10), 6232–6251 (2016).
16. Z. Zhong et al., "Spectral–spatial residual network for hyperspectral image classification: a 3-D deep learning framework," *IEEE Trans. Geosci. Remote Sens.* **56**(2), 847–858 (2018).
17. G. Huang et al., "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, Vol. 1, pp. 2261–2269 (2017).
18. A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master's Thesis, Department of Computer Science, University of Toronto (2009).
19. Y. Netzer et al., "Reading digits in natural images with unsupervised feature learning," *NIPS Workshop Deep Learn. and Unsupervised Feature Learn.*, pp. 5–15 (2011).
20. J. Deng et al., "ImageNet: a large-scale hierarchical image database," in *IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 248–255 (2009).
21. D. Gu, "3D densely connected convolutional network for the recognition of human shopping actions," Master Thesis, University of Ottawa, Ottawa, Ontario (2017).
22. W. Song et al., "Hyperspectral image classification with deep feature fusion network," *IEEE Trans. Geosci. Remote Sens.* **56**(6), 3173–3184 (2018).
23. S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *32nd Int. Conf. Mach. Learn.*, pp. 448–456 (2015).
24. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. and Stat.*, pp. 315–323 (2011).
25. K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 770–778 (2016).
26. C. Szegedy et al., "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 2818–2826 (2016).
27. M. G. Arnold, P. Vouzis, and J. H. Cho, "Bitstream efficiency of field programmable one-hot arrays," in *IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, pp. 436–441 (2010).

28. S. Xie et al., "Aggregated residual transformations for deep neural networks," in *IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 5987–5995 (2017).

Chunju Zhang received her PhD in cartography and geographic information system from the School of Geographical Science, Nanjing Normal University, Nanjing, China, in 2013. She is currently an associate professor at Hefei University of Technology, Hefei. Her research interests include high-resolution image processing, deep learning techniques, and computational intelligence in spatial data including GIS and remote sensing data.

Guandong Li is currently working toward a master's degree in the School of Civil Engineering, Hefei University of Technology, Hefei, China. His research interests include remote sensing data processing and machine learning.

Shihong Du is currently an associate professor with Peking University, Beijing. His research interests include qualitative knowledge representation, reasoning and its applications, and semantic understanding of spatial data including GIS and remote sensing data.

Biographies of the other authors are not available.