Symmetric All Convolutional Neural-Network-Based Unsupervised Feature Extraction for Hyperspectral Images Classification

Mingyang Zhang, *Member, IEEE*, Maoguo Gong, *Senior Member, IEEE*, Haibo He, *Fellow, IEEE*, and Shengqi Zhu, *Member, IEEE*

***Abstract*—Recently, deep-learning-based feature extraction (FE) methods have shown great potential in hyperspectral image (HSI) processing. Unfortunately, it also brings a challenge that the training of the deep learning networks always requires large amounts of labeled samples, which is hardly available for HSI data. To address this issue, in this article, a novel unsupervised deep-learning-based FE method is proposed, which is trained in an end-to-end style. The proposed framework consists of an encoder subnetwork and a decoder subnetwork. The structure of the two subnetworks is symmetric for obtaining better downsampling and upsampling representation. Considering both spectral and spatial information, 3-D all convolution nets and deconvolution nets are used to structure the encoder subnetwork and decoder subnetwork, respectively. However, 3-D convolution and deconvolution kernels bring more parameters, which can deteriorate the quality of the obtained features. To alleviate this problem, a novel cost function with a sparse regular term is designed to obtain more robust feature representation. Experimental results on publicly available datasets indicate that the proposed method can obtain robust and effective features for subsequent classification tasks.**

***Index Terms*—3-D convolution neural networks, feature extraction (FE), hyperspectral images (HSIs), unsupervised training.**

I. Introduction

WITH the rapid development of the satellite sensing technology, hyperspectral images (HSIs) data can be easily obtained. Compared with conventional optical image data, HSI data have a very high resolution on the spectral domain, which can provide unique spectral features besides spatial information. The characteristic of jointly spectral–spatial imaging makes the content of HSI data distinguished in many areas [1], [2], such as geological surveys [3], vegetation research [4], atmospheric science research [5], marine research [6], agriculture [7], etc. However, detailed spectral information causes a significant increase in the spectral dimension of HSI, which will result in the Hughes phenomenon [8]. Moreover, adjacent spectral bands are always highly correlated, which contain redundancy information [9]. Thus, it is difficult to obtain useful information of interest directly from raw HSI data. To deal with raw HSI data, feature extraction (FE) has been considered to be a very effective way [10], which is one of the most potential researching areas and still an open challenge in HSI processing.

FE methods map the raw HSI data from original feature space to a low-dimension feature space, in which features are more distinguished and less correlated. Consequently, data redundancy and the Hughes phenomenon can be well alleviated [11]. Traditional FE methods are often divided into two types according to the usage of labeled samples. One is the supervised/semisupervised FE method [12]–[14], such as linear discriminant analysis (LDA) [15], local Fisher discriminant analysis (LFDA) [16], local discriminant embedding (LDE) [17], etc. LDA maximizes the interclass distance and minimizes the intraclass distance at the same time to obtain the optimal projection. Based on LDA, LFDA assigns weights according to the density between samples, and the closely connected samples will obtain a greater weight. LDE seeks the best projection by integrating the information of neighbor and class relations between samples. Labels play a very important role in these methods because it can guide the process of FE so that the features extracted from the data with the same label are similar while the differences between features extracted from data with different classes of labels are great [18]. The other one is the unsupervised FE method [19], such as principal component analysis (PCA) [20], independent component analysis (ICA) [21], locally linear embedding (LLE) [22], factor analysis (FA) [23], etc. PCA maps the input data to the feature domain by means of orthogonal transformations to extract better features. ICA is devoted to seek independent hidden factors or components in statistical data. LLE, one of the manifold learning methods, attempts to

Manuscript received October 23, 2019; revised May 12, 2020; accepted August 25, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61906147; in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2019JQ-471; and in part by the National Science Foundation under Grant ECCS 1917275. This article was recommended by Associate Editor M. Han.

Mingyang Zhang and Maoguo Gong are with the School of Electronic Engineering, Xidian University, Xi'an 710071, China, and also with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an 710071, China (e-mail: omegazhangmy@gmail.com; gong@ieee.org).

Haibo He is with the Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881 USA (e-mail: haibohe@uri.edu).

Shengqi Zhu is with the National Laboratory of Radar Signal Processing, Xidian University, Xi'an 710071, China (e-mail: sqzhu@xidian.edu.cn).

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

Digital Object Identifier 10.1109/TCYB.2020.3020540

extract features by revealing the inherent structure of nonlinear distributed data [24]. FA achieves the purpose of FE by extracting the intrinsic common factors between the samples. The difference between unsupervised and supervised methods is that labels are not available in the process of unsupervised FE [25]. Unsupervised FE methods use unlabeled data to infer the intrinsic features of raw data. When it comes to HSI data, it usually has a large number of pixels, and its label is pixel level. In practice, labels of HSI data are very insufficient, and manual annotation is time consuming and laborious. Therefore, unsupervised FE methods possess more practical values.

In early research, most FE methods, such as PCA and ICA, are designed to be a monolayer structure [26], which is also considered as a shallow model. In recent years, however, deep-learning-based deep models, which often consist of three or more layers, have become more popular. Compared with the traditional features extracted from shallow models, deep abstract features extracted from deep models have better representation performance and make subsequent tasks, such as classification more accurate [27]. Moreover, the deep model is considered to be more robust [28]. Therefore, for the past few years, a large number of deep learning FE methods have been introduced to HSI processing [29]. Stacked autoencoder with logistic regression (SAE-LR) [30] was presented to HSI FE and classification, where a stacked autoencoder is used to extract the initial features, and then a logical regression layer is added to obtain effective classification results. On the basis of SAE-LR, spatial updated deep autoencoders (SDAEs) [31] were put forward to make better use of deep features and suppress noise. Besides SAE, another deep network structure, deep belief network (DBN), was also applied in the process of FE [32], [33]. DBN with logistic regression (DBN-LR) [34] combines DBN and logistic regression to classify HSI. As one of the most representative deep learning models, the convolutional neural network (CNN) [35] achieves better performance in FE and classification [36]. A spectral-spatial-feature-based classification framework was presented to extract spectral and spatial features simultaneously through the balanced LDE algorithm and CNN [37]. In [38] and [39], 3-D convolution kernels were introduced into deep-learning-based FE methods. Zhang *et al.* [40] proposed a patch-to-patch CNN for classification of hyperspectral and LiDAR data.

Although better performance is achieved, supervised deep learning FE methods require large amounts of labeled samples to support the training of deep structural networks. Recently, researchers have focused more on the data augmentation and unsupervised training framework for deep-learning-based FE frameworks, which contain a more practical value for HSI processing. Li *et al.* [41] proposed the pixel-pair method based on CNN which can improve the quality of training samples for the CNN-based FE framework. In [42] and [43], two unsupervised training frameworks were proposed, which are based on generative adversarial nets and the autoencoder structure, respectively. However, how to design network structures to extract spatial–spectral features from 3-D raw HSI data more effectively and how to effectively train the proposed networks without supervision are still an open challenge.

To address this challenge, a symmetric all CNN (SACNN) is proposed in this article. SACNN is an end-to-end network and consists of two parts: 1) encoder subnetwork and 2) decoder subnetwork. The encoder network encodes inputs to a low-dimensional space to obtain feature representation. The decoder network is designed to restore the input by decoding the output of the encoder network. This form of network structure makes the entire FE process get rid of labeled samples, which adopts the reconstruction errors to train the proposed networks. As mentioned above, CNN has powerful FE capabilities to extract high-level abstract features and make classification more accurate. Moreover, in order to better jointly make full use of spatial information and spectral information of raw HSI data, 3-D convolution is introduced into the proposed model. We design a novel all 3-D convolutional network and a novel all 3-D deconvolutional network to constitute the proposed encoder subnetwork and decoder subnetwork, respectively. Since 3-D convolution kernel and 3-D deconvolution kernel contain large numbers of parameters, sparse representation is introduced into the design of the cost function. The main contributions of the proposed method are listed as follows.

1) We propose a novel FE method for HSIs which is a deep model consisting of 3-D convolution kernels. The proposed method can effectively obtain high-quality spatial–spectral features from raw HSIs which can significantly benefit subsequent classification tasks.
2) We propose a novel end-to-end training framework for the proposed feature extractor. Different from conventional deep-learning-based FE methods that rely on large amounts of labeled samples for their training heavily, the proposed framework is fully unsupervised by using the encoder–decoder architecture. Our proposed method mitigates the dependence of deep-learning-based FE methods on a large number of labeled samples.
3) We design a novel all convolutional and deconvolutional net architecture to build the encoder subnetwork and decoder network, respectively. The downsampling and upsampling processes are integrated into the network optimization process. Moreover, multiscale and multilevel features are also considered in the FE process, which can better improve classification performance.

The remainder of this article is as follows. Section II presents the background and motivation of this work. Section III gives a detailed description of our proposed approach. The experimental part is carried out in Section IV. Section V summarizes the entire article.

II. Background and Motivation

A. Convolution and Deconvolution

CNN, a kind of artificial neural network, has become a hot research field in deep learning due to its outstanding performance in image recognition tasks. In CNN, the role of convolution is designed to extract latent features of inputs, and deconvolution is used to restore inputs. In fact, the deconvolution here is not the inverse process of convolution but a

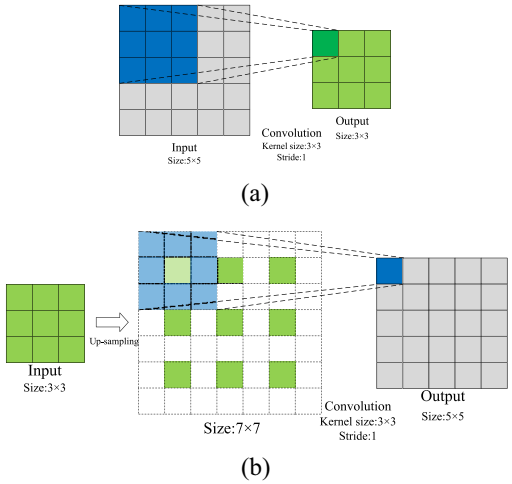


Fig. 1. Illustration of (a) convolution and (b) deconvolution. The implementation of deconvolution is divided into two steps: upsampling process and convolution process.

special convolution called the transposed convolution. Fig. 1 shows the differences between convolution and deconvolution.

There is a simple example of convolution in Fig. 1(a), a 3×3 convolution kernel slides on a 5×5 input, and with each sliding, the convolution kernel is multiplied by the area it covers. Eventually, a 3×3 feature map is obtained. In the convolution process, a certain connection exists between the sizes of inputs, outputs, and convolution kernels, which is given by

$$W_2 = (W_1 - F + 2P)/S + 1 \tag{1}$$

$$H_2 = (H_1 - F + 2P)/S + 1 \tag{2}$$

where W_2 is the width of the feature map after convolution and W_1 is the width of input. H_2 and H_1 denote the heights of the feature map and input, respectively, which are generally equal to W_2 and W_1. F is the size of a convolution kernel. P denotes the number of zero padding. For example, if P is equal to 1, it means that a circle of 0 is padded to the edge of the original input. S represents the stride of a convolution kernel movement. According to (1) and (2), we can calculate the size of the feature map in Fig. 1(a), that is, $((5-3+2\times0)/1+1)\times((5-3+2\times0)/1+1)=3\times3$. Due to the parameter sharing mechanism, each feature map is obtained by a unique convolution kernel. That is to say, a convolution kernel can extract only one pattern of feature. By adding the number of convolution kernels, various patterns of features can be obtained.

As we see in Fig. 1(b), different from the process of convolution, the deconvolution output size is larger than the input size. However, deconvolution is not the inverse process of convolution. In Fig. 1(b), the entire deconvolution process is divided into two steps: 1) an upsampling process and 2) a convolution process, respectively. As shown in Fig. 1(b), given a 3×3 input, the first step to achieve deconvolution is to upsample the input. If a 5×5 output is required, the input should be upsampled to a larger size (e.g., 7×7). It can be seen that after upsampling, many pixels with a value of 0 appear in the input. The second step is to perform convolution on the upsampled input, where the convolution kernel size is 3×3, the stride is set to 1, and P is 0. According to (1) and (2), the output size is $((7-3+2\times0)/1+1)\times((7-3+2\times0)/1+1)=5\times5$. From the entire process of deconvolution, it is essentially a special kind of convolution. For the input, the stride becomes a fraction actually. Thus, deconvolutions are also called fractionally strided convolutions.

Deconvolution brings great benefits to the study of deep learning. First, deconvolution can be used to visualize CNN [44], which can help us better understand the obtained features. Deconvolution transforms the features from the feature space to the pixel space to find out the relationship between an input and a specific feature map and achieve the purpose of analyzing and understanding CNN. Second, since deconvolution has the characteristic that the output size can be larger than the input size, it is very suitable for upsampling. In [45], deep convolutional generative adversarial networks (DCGANs) are proposed to generate a picture that is similar to the real picture. Deconvolution is introduced to constitute the generator in DCGAN, which can effectively generate an image from noise with a specific distribution.

B. Motivation

As mentioned in Section I, the absence of labeled samples cannot support the training of traditional supervised deep learning networks, which limits the application of deep learning-based methods in HSI data. For the purpose of unsupervised FE, an encoder–decoder architecture is applied to structure our proposed unsupervised network. Specifically, the input is encoded by an encoder subnetwork to obtain a code that can represent the input, and then a decoder subnetwork is used to reconstruct the input, which is similar to an autoencoder. However, a traditional autoencoder is based on fully connected networks. The fully connected structure causes expansion of the number of parameters, easy overfitting, and local optimum. Compared with the fully connected network, CNN's local perception mechanism enables it to better discover the spatial features of the image, and multiple levels of spatial features can be extracted by using multilayer convolution. In addition, the parameter sharing mechanism of CNN can greatly reduce the number of network parameters so that the overfitting problem can be alleviated. Therefore, the convolution is applied in the design of the encoder, and in order to reconstruct input effectively, the deconvolution which is a special type of convolution, is applied to the design of the decoder.

In convolutional networks, the presence of pooling layers can result in the loss of a large amount of image information, due to its fixed downsampling strategy. However, when deconvolution is used to generate images, rich details are required to produce the final representation of the input. The final representation is, of course, an image with all of its details (e.g., position, posture, and texture) [46]. Consequently, fixed downsampling and upsampling strategies are not suitable for image reconstruction. For object recognition tasks, the all convolutional network can replace the pooling operation by stride convolution kernels [47]. Inspired by this, a novel all convolutional network is proposed in the SACNN, in which there are no pooling layers. In this case, SACNN can automatically

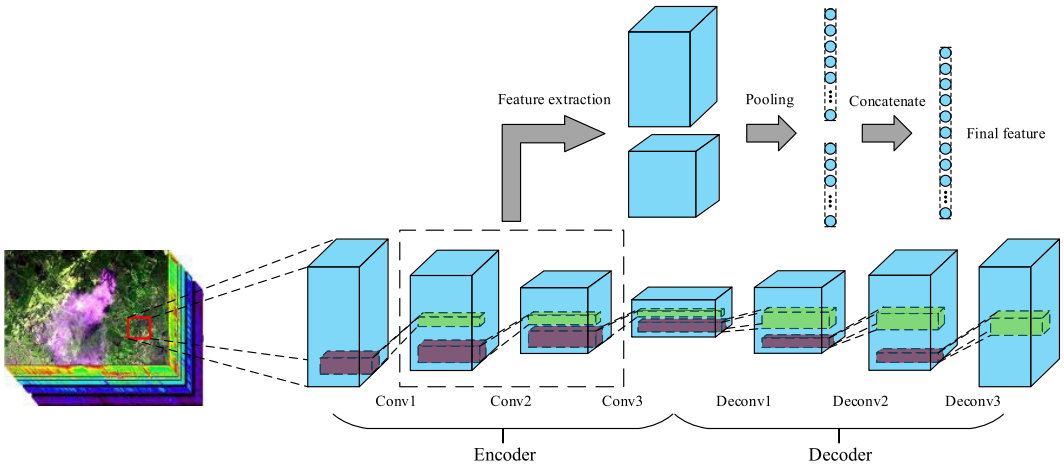


Fig. 2. Framework of 2D-SACNN for HSI FE. It is an SACNN and consists of two parts: the encoder and the decoder. Here, the first three convolutional layers constitute the encoder, which can transform the input into a representation. The latter three deconvolutional layers constitute the decoder, which can reconstruct input. Both the convolutional and deconvolutional layers are equipped with a batch normalization layer and an activation function. After training, the encoder can be used as a feature extractor. The output of the first and second convolutional layers are extracted and then pooled and combined to obtain the final feature.

learn downsampling and upsampling strategies in the encoder and the decoder, respectively.

III. Methodology

A. 2D-SACNN

In this section, a 2D-SACNN is proposed to extract features of HSI in an unsupervised approach. As we can see in Fig. 2, the proposed 2D-SACNN consists of an encoder and a decoder, and the encoder contains three convolutional layers while the decoder contains three deconvolutional layers.

1) Encoder: To extract multilevel features of input, the encoder is designed as a deep convolution structure. In the encoder, the key part is the convolution operation. The output of each convolutional layer in the encoder can be calculated as follows:

$$v_{ij}^{xy} = f\left(\sum_{m=1}^{M_{i-1}}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)} + b_{ij}\right) \tag{3}$$

where v_{ij}^{xy} represents the value at position (x, y) which exists in the jth feature map of the ith layer, and w_{ijm}^{pq} denotes the weight of position (p, q) which connects to the mth feature map. b_{ij} indicates the bias which exists in the jth feature map of the ith layer. M_{i-1} is the number of feature maps in the $(i-1)$th layer. P_i and Q_i denote the height and width of the convolution kernel, respectively.

During the training process of the deep neural network, if the parameters of the previous layer change, the input distribution of the following layers will also change, which is called the internal covariate shift [48]. It is hard to obtain the optimal parameters if the distribution of one layer in the network is always changing. Nevertheless, this problem can be solved by adding batch normalization layers [48], that is to say, the input of all layers in the network is normalized to a nearly same distribution, which is a standard Gaussian distribution. Consequently, it is critical to put batch normalization layers into the designed encoder.

After the convolution operation and batch normalization, the activation function needs to be applied to increase the nonlinearity of the neural network. The rectified linear unit (ReLU) [28] is selected as the activation function of the encoder, which is defined as

$$f(x) = \max(0, x). \tag{4}$$

According to (4), the output of some neurons in the network will be set to 0, which allows sparsity to be introduced into the network. Thus, the network can be trained more robust and faster.

Fig. 2 presents the 2D-SACNN framework for FE of HSI, and the first half of this network is the encoder subnetwork. As an input, an $N \times N \times B$ HSI cube is fed into the encoder, in which $N \times N$ represents the neighborhood size of a current pixel, and B denotes the number of spectral bands. In the encoder, convolutional layers are applied to the input so that multilevel features can be obtained. In order to enable the network to retain more detail, the pooling layer does not exist in the encoder and decoder. The size of the convolution kernel is set to 3×3. Compared with 5×5, 7×7, and even larger convolution kernels, the 3×3 kernel has a smaller receptive field, but the area of the receptive field can be enlarged by increasing the number of convolution kernel layers. More important, the 3×3 kernel can avoid expression bottlenecks and enhance nonlinear expression ability [49].

2) Decoder: In order to verify whether the features extracted by the encoder can correctly represent the original input, these features need to be restored to the input format, so that the unsupervised FE process can be achieved. Thus, the role of the decoder in 2D-SACNN is to reconstruct input, and the key operation is deconvolution. As mentioned in Section II-A, the biggest difference between deconvolution and convolution is that upsampling needs to be performed first during the process of deconvolution. The second step of deconvolution is exactly the same as the process of convolution. Therefore, 3×3 kernels and batch normalization layers are

also applied to the decoder. The ReLU is introduced to the decoder as the activation function except for the last layer. Since the tanh function can map the input to $[-1, 1]$, this ensures that the output of the network to be in the same range as the input. The tanh function is applied to the activation function of the last layer and is given by

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{5}$$

In Fig. 2, the second half of 2D-SACNN is the decoder whose input is the output of the encoder. A symmetrical structure is formed by the connection between the decoder and the encoder. Convolutional layers and deconvolutional layers in the network are one-to-one correspondence, and there is also no pooling layer in the decoder so that the information obtained by the convolution can be accurately used for reconstruction.

B. 3D-SACNN

In Section III-A, a 2D-SACNN is proposed to perform unsupervised FE of HSI. However, there is still a problem that the 2-D convolution only performs convolutions on each individual spectral band of the HSI, which means that only the spatial features can be extracted and the spectral features are ignored. Therefore, the 3D-SACNN is proposed to capture features in both spectral and spatial dimensions simultaneously.

Compared to 2-D convolution, 3-D Convolution implements a convolution operation not only in the spatial dimension but also in the spectral dimension. An example of a 3-D convolution is shown in Fig. 3. A 3-D convolution kernel is applied to a $5 \times 5 \times 4$ input, and then a $3 \times 3 \times 2$ output is produced. The size of the 3-D convolution kernel is $3 \times 3 \times 3$, and P and stride are set to 0 and 1, respectively.

Similar to 2D-SACNN, 3-D convolution requires 3-D deconvolution to correspond to it in 3D-SACNN. Accordingly, 3-D deconvolution is introduced to the decoder. As shown in Fig. 4, it attempts to turn a $3\times3\times1$ input into a $5\times5\times2$ output through 3-D deconvolution operation. To this end, upsampling is first used to enlarge the input to a size of $7 \times 7 \times 3$. Then, 3-D convolution with a kernel size of $3 \times 3 \times 2$ is used on this $7 \times 7 \times 3$ cube, and finally a $5 \times 5 \times 2$ output is obtained. According to the dimension of HSI data, in the proposed method, the sizes of 3-D convolution and deconvolution kernels are set as $3 \times 3 \times 32$. Specifically, 3×3 corresponds to the spatial dimensions of inputs, and 32 denotes the number of kernel channels. The stride is set as 1.

In 3D-SACNN, 2-D convolution is replaced by 3-D convolution, and the output of each convolutional layer in the encoder can be calculated by

$$v_{ij}^{xyz} = f\left(\sum_{m=1}^{M_{i-1}}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)} + b_{ij}\right) \tag{6}$$

where v_{ij}^{xyz} represents the value at position (x, y, z) which exists in the jth feature map of the ith layer, and w_{ijm}^{pqr} denotes the weight of position (p, q, r) which connects to the mth feature map. b_{ij} indicates the bias which exists in the jth feature map of the ith layer. M_{i-1} is the number of continuous feature maps in the $(i-1)$th layer which connects to the jth feature map. R_i is the size of the 3-D convolution kernel along the spectral dimension, and P_i and Q_i denote height and width of the 3-D convolution kernel, respectively.

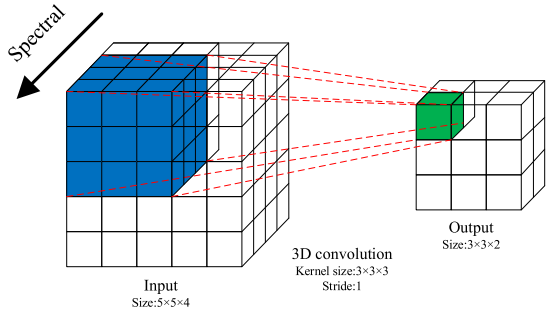


Fig. 3. Illustration of 3-D convolution. Unlike 2-D convolution, 3-D convolution also performs convolution on the spectral dimension.

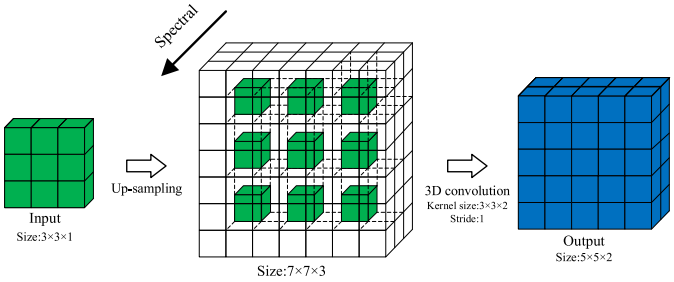


Fig. 4. Illustration of 3-D deconvolution. It is divided into two steps: 3-D upsampling process and 3-D convolution process.

Due to the parametric sharing mechanism of 3-D convolution, a 3-D convolution kernel can only capture one type of feature. It is well known that multilevel features require multiple convolution kernels to convolute low-level features; hence, the number of convolution kernels should be increased layer by layer. As shown in Fig. 5, in the encoder with multiple 3-D convolutional layers, the number of convolution kernels on the next layer is twice that of the previous layer. Correspondingly, in the decoder, the number of convolution kernels on the next layer is half that of the previous layer.

C. Training and Feature Extraction

In order to train SACNN (both 2D-SACNN and 3D-SACNN), an appropriate cost function needs to be defined. In the proposed method, the reconstruction error is used in the cost function. Accordingly, we choose the mean square error (MSE) as the cost function, and L2 regularization is also added to it to pursue sparsity. The cost function $L(X, R)$ is defined as (7), which is optimized by the Adam algorithm [50]

$$L(X,R) = \frac{1}{P_1 Q_1 R_1}\sum_{i=1}^{P_1}\sum_{j=1}^{Q_1}\sum_{k=1}^{R_1}\left(x_{ijk} - r_{ijk}\right)^2 + \frac{\lambda}{2}\sum_{l=1}^{N}\sum_{i=1}^{P_2}\sum_{j=1}^{Q_2}\sum_{k=1}^{R_2}\left(w_{ijk}^{l}\right)^2 \tag{7}$$

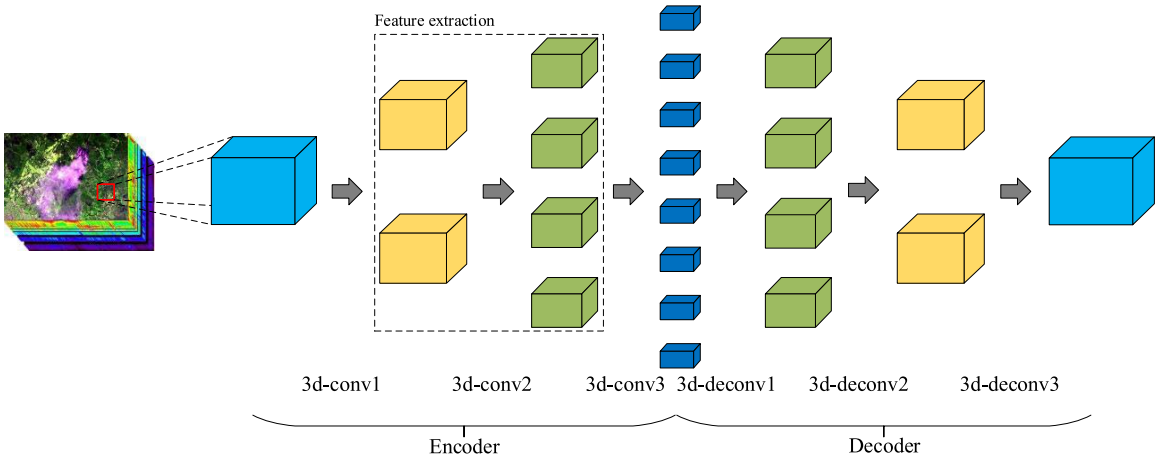


Fig. 5. Illustration of 3D-SACNN for HSI FE. It is an SACNN and consists of two parts: the encoder (the first three 3-D convolutional layers) and the decoder (the latter three 3-D deconvolutional layers). Both the 3-D convolutional and 3-D deconvolutional layers are equipped with a batch normalization layer and an activation function. After training, the encoder can be used as a feature extractor. The output of the first and second convolutional layers are extracted and then pooled and combined to obtain the final feature.

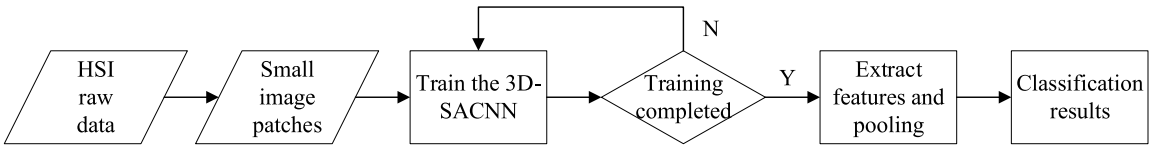


Fig. 6. Flowchart of the proposed algorithm.

where X and R represent the input and output of SACNN. P_1, Q_1, and R_1 denote the length, width, and height of both input cube and output cube, respectively. P_2, Q_2, and R_2 represent the length, width, and height of weight cube, respectively. N is the number of weights in the SACNN, and λ is the weight decay parameter. x_{ijk} and r_{ijk} denote the values of the input and output at position (i, j, k), respectively. w^l_{ijk} represents the value of the lth weight at the position (i, j, k). The second term in (7) represents the L2 regularization term.

Due to the strong fitting ability of the neural network and the limited HSI training samples, the overfitting problem often occurs. Therefore, in the proposed network, the ReLU active function and L2 regularization are used to alleviate this problem. L2 regularization can limit the value of weights so that the model can avoid to arbitrarily fit the random noise in training data. In addition, as mentioned in Section III-A, the usage of ReLU can make the network sparse and reduce the interdependencies of parameters. The above two methods can weaken the fitting ability of the network to a certain extent in order to alleviate the overfitting problem.

Once the SACNN algorithm training process is completed, the trained encoder network can be used as a feature extractor. In order to achieve complete unsupervised FE, any form of label information will not be used in the subsequent FE process. Since the encoder is a deep network, multilevel spectral–spatial features can be extracted by multilayer convolution in the encoder, and for the entire process of SACNN, each layer of the encoder can reach the output through different layers. Thus, the features in each layer of the encoder can, to some extent, represent the original input. However, although the dimensionality of the input has been reduced through the encoder, the obtained multilevel features are still too large in amounts, which may deteriorate subsequent classification tasks. In order to further reduce the dimension of the features, we use the maxpooling to make all of the encoder's feature maps into a 1-D vector, where the size of the pooling window is the same as the spatial dimension of feature maps. It is worth noting that the multilevel features concatenated together have better performance in the classification tasks than individual-level features. The above part of Fig. 2 shows the process of FE. Experiments on datasets show that the first two layers of the encoder have the best performance. Thus, we first feed the data into 2D-SACNN and then extract the raw features in the first two layers of the encoder and maxpooling them to produce two vectors. Eventually, these two vectors are concatenated to obtain the final feature. For example, given two features $7 \times 7 \times 169$ and $5 \times 5 \times 138$ from different layers, we use maxpooling operations to obtain $1 \times 1 \times 169$ and $1 \times 1 \times 138$, respectively, and then concatenate them to become $1 \times 1 \times 307$. The same FE strategy is applied to 3-D-SACNN. The flowchart of the entire algorithm is shown in Fig. 6.

IV. Experimental Study

A. Hyperspectral Dataset Description and Evaluation Criteria Introduction

In the experiment, three public available datasets, including Indian Pines, Salinas, and Kennedy Space Center (KSC), are employed to verify the effectiveness of the proposed algorithm.

The Indian Pines dataset was obtained over the Indian Pines test site in northwestern Indiana. It contains 145×145 pixels

TABLE I
NUMBER OF TRAINING AND TEST SAMPLES USED IN THE INDIAN PINES DATASET

Class		Samples	
No.	**Ground truth**	**Training**	**Test**
C1	Alfalfa	7	39
C2	Corn-notill	215	1213
C3	Corn-mintill	125	705
C4	Corn	36	201
C5	Grass-pasture	73	410
C6	Grass-trees	110	620
C7	Grass-pasture-mowed	5	23
C8	Hay-windrowed	72	406
C9	Oats	3	17
C10	Soybean-notill	146	826
C11	Soybean-mintill	369	2086
C12	Soybean-clean	89	504
C13	Wheat	31	174
C14	Woods	190	1075
C15	Bldg-Grass-Trees-Drives	58	328
C16	Stone-Steel-Towers	14	79
Overall		1543	8706

TABLE II
NUMBER OF TRAINING AND TEST SAMPLES USED IN THE SALINAS DATASET

Class		Samples	
No.	**Ground truth**	**Training**	**Test**
C1	Brocoli_green_weeds_1	302	1707
C2	Brocoli_green_weeds_2	559	3167
C3	Fallow	297	1679
C4	Fallow_rough_plow	210	1184
C5	Fallow_smooth	402	2276
C6	Stubble	594	3365
C7	Celery	537	3042
C8	Grapes_untrained	1691	9580
C9	Soil_vinyard_develop	931	5272
C10	Corn_senesced_green_weeds	492	2786
C11	Lettuce_romaine_4wk	161	907
C12	Lettuce_romaine_5wk	290	1637
C13	Lettuce_romaine_6wk	138	778
C14	Lettuce_romaine_7wk	161	909
C15	Vinyard_untrained	1091	6177
C16	Vinyard_vertical_trellis	272	1535
Overall		8128	46001

and 220 spectral bands in the 0.4–2.5 μm wavelength range. Since there are 20 bands covering the region of water absorption, the remaining 200 bands are used for classification. In this dataset, 16 species are selected as categorized samples. The number of each class and their corresponding training and test samples are given in Table I.

The Salinas dataset was collected over the Salinas Valley, CA. This scene consists of 512×217 pixels and 224 spectral bands. Due to the same reason as the Indian Pines dataset, 20 bands are discarded and 204 bands are used for the experiment. Salinas ground truth is labeled into 16 classes, and the details of each class are provided in Table II.

The KSC dataset was acquired over the KSC, FL, on March 23, 1996. After removing the water absorption and low signal-to-noise ratio bands, the number of bands drops to 176. To perform the classification task, 13 classes are defined to represent different land cover types. Table III provides the numbers of each class and their corresponding training and test samples.

TABLE III
NUMBER OF TRAINING AND TEST SAMPLES USED IN THE KSC DATASET

Class		Samples	
No.	**Ground truth**	**Train**	**Test**
C1	Scrub	115	646
C2	Willow swamp	37	206
C3	CP hammock	39	217
C4	Slash pine	38	214
C5	Oak/Broadleaf	25	136
C6	Hardwood	35	194
C7	Swamp	16	89
C8	Graminoid marsh	65	366
C9	Spartina marsh	78	442
C10	Cattail marsh	61	343
C11	Salt marsh	63	356
C12	Mud flats	76	427
C13	Water	140	787
Overall		788	4423

To quantificationally evaluate the quality of features extracted by various methods, classification performance is tested. We introduce three criteria to evaluate the classification performance as follows.

1) Overall Accuracy: Overall accuracy (OA) represents the proportion of the correctly sampled sample in all test samples.

2) Average Accuracy: Average accuracy (AA) indicates the average of the classification accuracies for all classes.

3) Kappa Coefficient (Kappa): The Kappa coefficient is mainly used to compare and analyze whether the difference between two images is caused by "accidental" or "inevitable" factors, and is an index that can indicate overall consistency and classification consistency.

Moreover, a statistical test is also introduced to compare the performance among comparison methods. McNemar's test can be applied to determine whether there is a difference in the matching ratio, which is denoted as follows:

$$z_{01} = \frac{e_{01} - e_{10}}{\sqrt{e_{01} + e_{10}}} \tag{8}$$

where e_{01} represents the number of samples correctly classified in classification 0 but misclassified in classification 1, whereas e_{10} is exactly the opposite. Since McNemar's test is a statistic test based on the standard normal distribution, the null hypothesis is rejected at $p = 0.05$ ($|z| > 1.96$).

B. Feature Extraction Strategy Experiments

In this part of the experiment, we investigate the FE strategy of 3D-SACNN. After the training process, the trained encoder can be applied as a feature extractor, then the extracted features are fed into the classifier. However, rather than sending all layers of the encoder directly into the classifier, the features of each layer need to be pooled and combined. Tables IV–VI show the effect of different FE strategies on the three datasets. As shown in Tables IV–VI, the "un-pooling" represents that there is no pooling operation before the features of each layer are used to classify, and the "pooling" means that the features of corresponding layers go through the max-pooling operation before being sent to the classifier, and the features of each layer are turned into a 1-D vector. For the Indian Pines dataset, these features that have performed the maxpooling operations have better classification performance

TABLE IV
EXPERIMENTAL RESULTS OF DIFFERENT FE STRATEGIES FOR THE INDIAN PINES DATASET

	Un-pooling		Pooling	
	OA(%)	**Feature size**	**OA(%)**	**Feature size**
h1	59.28	66248	95.48	1352
h2	50.51	55200	96.96	2208
h3	63.15	30816	93.04	3424
h1,h2	28.68	121448	**97.07**	3560
h1,h3	32.07	97064	94.22	4776
h2,h3	28.33	86016	94.07	5632
h1,h2,h3	24.66	152264	94.27	6984

TABLE V
EXPERIMENTAL RESULTS OF DIFFERENT FE STRATEGIES FOR THE SALINAS DATASET

	Un-pooling		Pooling	
	OA(%)	**Feature size**	**OA(%)**	**Feature size**
h1	91.95	50862	97.77	1038
h2	89.84	42600	98.23	1704
h3	49.34	23976	90.26	2664
h1,h2	81.63	93462	**99.17**	2742
h1,h3	46.74	74838	90.07	3702
h2,h3	47.07	66576	89.89	4368
h1,h2,h3	41.83	117438	89.72	5406

TABLE VI
EXPERIMENTAL RESULTS OF DIFFERENT FE STRATEGIES FOR THE KSC DATASET

	Un-pooling		Pooling	
	OA(%)	**Feature size**	**OA(%)**	**Feature size**
h1	90.38	56840	98.01	1160
h2	91.39	45600	96.63	1824
h3	88.04	23904	96.38	2656
h1,h2	86.39	102440	**98.21**	2984
h1,h3	81.95	80744	96.90	3816
h2,h3	84.08	69504	96.29	4480
h1,h2,h3	78.10	126344	96.63	5640

than those that have not been pooled. For example, the OA of $h1$ is increased by 36.2% after the maxpooling operation. There are three layers in the encoder, and $h1$ represents the feature of the first layer. After the pooling operation, $h1$ and $h2$ achieve relatively high values of OA, and Table IV shows that after concatenating $h1$ and $h2$, OA can reach 97.07%, which is the highest value of OA. Similarly, for the Salinas and KSC datasets, the features of each layer can be better for classification after the pooling operation, and through the pooling operation and concatenation operation, features of the first and second layers can achieve the highest OA of 99.17% and 98.21%, respectively.

Without pooling, the dimensionality of final extracted features is still high. Therefore, the Hughes phenomenon may occur, resulting in poor classification performance. The usage of pooling operations can reduce the dimensionality of features in order to avoid the appearance of the Hughes phenomenon. Experimental results show that the classification accuracy can be improved by concatenating features of different layers. This is because the features of different layers have different scales and abstract degree. The combination of them can increase the diversity of features. Moreover, it can be seen that the features of the first two layers are better for classification because they contain more shape and outline features on spatial and spectral domains. Since there are no labeled samples for guiding the training process, the features of the third layer are more abstract for reconstruction, which has less contribution to classification. It is also worth noting that the performance of the FE strategy proposed in this article has the best performance on the three datasets, which indicates that this strategy is robust to different datasets.

C. Comparison With Classification Performance

To evaluate the performance of an unsupervised FE algorithm, one common approach is to apply them as feature extractors on supervised datasets and then evaluate the classification performance on those obtained features. In order to verify the validity of extracted features, the proposed algorithm is compared with some state of the arts. For classic ones, a mathematical method and deep-learning-based FE methods are introduced to compare them with the proposed SACNN, including extended attribute profile (EAP) [51], stacked autoencoders (SAE-LR) [30], and CNN. In [51], an advanced EAP-based PCA is proposed for the classification task of HSI. After FE, in order to obtain higher classification accuracy, all spectrum data for each pixel are added to the features extracted by EAP. SAE-LR trains a stacked fully connected autoencoder as the feature extractor. The SAE-LR network has a total of five layers, including one input layer, three hidden layers, and one output layer. LR is applied as a classifier for the output layer. Parameters of the SAE-LR network are fine tuned, referring to [30]. CNN is an artificial neural network with convolutional calculation as its core. Corresponding to the number of layers in an SACNN encoder, the CNN network has five layers in total, including one input layer, three convolutional layers, and one softmax layer.

Besides, some recent state of the arts of unsupervised deep-learning-based FE methods are also introduced as competitors, including an unsupervised deep learning FE method (UDFE) [19], which proposed using greedy layerwise unsupervised pretraining to train a CNN model; a recursive autoencoders-based unsupervised feature learning method (RAE) [52], which proposed an unsupervised recursive autoencoders network model; an unsupervised deep residual convdeconv network-based method (RCDN) [43], which designed a deep residual convolution network for unsupervised feature learning; and a 3-D unsupervised spatial–spectral feature learning method (3D-USSFL) [53], which introduced 3-D convolutional kernels to improve unsupervised feature learning performance.

In the following experiments, during the SACNN training, the minibatch size is set to 64. The number of training iterations is 20 000, and the parameter λ of the L2 regularization is set to 0.0001. The learning rate of the Adam algorithm is chosen in the range of 0.00001 to 0.001 for different datasets. For the Indian Pines, Salinas, and KSC datasets, the sizes of input data are $11 \times 11 \times 200$, $9 \times 9 \times 204$, and $9 \times 9 \times 176$, respectively. Each experiment runs ten times independently. The values of evaluation criteria for experimental results are presented in the form of mean values $\pm$ standard deviation.

TABLE VII
CLASSIFICATION RESULTS ON THE INDIAN PINES DATASET

FE method	EAP	CNN -0.15	CNN -0.3	SAE-LR -0.15	SAE-LR -0.5	UDFE	RAE	RCDN	3D-USSFL	2D-SACNN	3D-SACNN
c1	89.17 ±3.80	76.67 ±0.00	94.25 ±0.17	78.97 ±10.21	96.50 ±5.68	94.80 ± 1.15	71.89 ± 3.63	96.10 ± 1.60	**100.00** ± 0.00	97.44 ±0.00	89.66 ±2.01
c2	93.36 ±0.96	85.56 ±3.67	85.96 ±4.05	85.77 ±1.33	93.31 ±2.52	93.53 ±0.34	**96.52** ± 0.41	96.04 ±0.25	94.91 ±0.21	94.10 ±0.05	96.30 ±0.14
c3	90.34 ±1.36	88.17 ±1.33	95.28 ±2.02	86.72 ±0.85	88.42 ±2.41	91.62 ±0.15	96.39 ±0.24	91.41 ±0.51	90.80 ±0.70	93.73 ±0.08	**98.45** ±0.15
c4	90.77 ±4.47	90.14 ±1.08	94.58 ±2.35	74.99 ±2.16	68.63 ±4.43	94.00 ±0.29	89.52 ±0.41	**98.10** ±0.27	95.06 ±0.55	98.03 ±0.01	95.10 ±0.15
c5	94.57 ±1.88	96.47 ±0.44	98.13 ±0.37	88.95 ±1.93	96.47 ±1.80	97.37 ±0.24	96.49 ±0.28	94.90 ±0.36	97.23 ±0.44	**98.72** ±0.00	98.52 ±0.27
c6	99.32 ±0.61	94.20 ±0.39	99.69 ±0.14	94.82 ±1.37	96.68 ±0.66	**99.89** ±0.17	98.46 ±0.21	99.50 ±0.15	99.80 ±0.12	99.03 ±0.00	96.19 ±0.34
c7	90.28 ±5.89	90.47 ±1.50	**100.00** ±0.00	91.65 ±6.32	**100.00** ±0.00	98.07 ±0.14	97.39 ±2.38	**100.00** ±0.00	97.06 ±0.00	**100.00** ±0.00	99.07 ±1.96
c8	**100.00** ±0.00	95.47 ±0.01	98.76 ±0.38	95.28 ±1.05	98.68 ±1.27	99.90 ±0.12	99.16 ±0.22	99.70 ±0.14	**100.00** ±0.00	**100.00** ±0.00	**100.00** ±0.00
c9	91.50 ±8.38	93.26 ±4.68	**100.00** ±0.00	85.98 ±9.89	79.50 ±1.58	**100.00** ±0.00	78.84 ±1.52	97.10 ±0.20	98.70 ±1.20	**100.00** ±0.00	**100.00** ±0.00
c10	89.53 ±1.40	85.47 ±1.64	96.32 ±2.11	86.40 ±1.29	89.69 ±2.40	82.30 ±0.81	89.50 ±0.22	90.73 ±0.50	91.07 ±0.32	95.80 ±0.01	**96.49** ±0.35
c11	93.56 ±1.11	88.99 ±2.67	93.23 ±3.48	87.57 ±0.41	90.11 ±1.83	92.26 ±0.20	94.81 ±0.29	95.08 ±0.27	95.30 ±0.10	96.30 ±0.04	**98.15** ±0.26
c12	**94.95** ±1.65	73.68 ±0.72	92.75 ±1.24	85.19 ±2.00	91.96 ±2.17	93.10 ±0.61	94.64 ±0.77	94.60 ±0.75	94.30 ±0.34	91.91 ±0.08	94.15 ±0.19
c13	99.30 ±0.38	98.82 ±0.00	**100.00** ±0.00	93.89 ±2.00	98.33 ±1.59	98.16 ±0.23	**100.00** ±0.00	**100.00** ±0.00	97.63 ±0.95	**100.00** ±0.00	96.58 ±0.40
c14	97.21 ±0.57	95.95 ±0.36	99.20 ±0.00	95.63 ±0.24	96.52 ±1.48	99.27 ±0.10	97.29 ±0.45	98.85 ±0.12	99.41 ±0.10	97.36 ±0.04	**99.71** ±0.03
c15	88.31 ±2.09	89.42 ±0.65	95.13 ±0.35	78.72 ±1.73	85.83 ±3.60	89.90 ±0.37	91.59 ±0.69	84.42 ±0.90	91.22 ±0.23	93.44 ±0.01	**97.51** ±0.27
c16	92.12 ±6.81	**100.00** ±0.00	**100.00** ±0.00	94.44 ±2.22	95.00 ±0.02	99.70 ± 0.47	98.65 ±0.00	98.26 ±0.80	96.31 ±0.77	**100.00** ±0.00	94.54 ±0.04
OA(%)	93.98 ±0.46	89.35 ±0.49	94.48 ±0.23	88.48 ±0.38	91.77 ±0.28	92.98 ±0.14	95.17 ±0.34	94.00 ±0.20	94.71 ±0.07	96.15 ±0.02	**97.47** ±0.02
AA(%)	93.39 ±0.96	90.17 ±0.54	96.45 ±0.29	87.81 ±1.25	91.60 ±0.48	95.24 ±0.34	93.20 ±0.72	95.92 ±0.42	96.18 ±1.00	**97.24** ±0.01	96.90 ±0.23
Kappa ×100	93.14 ±0.53	87.83 ±0.57	93.69 ±0.27	86.86 ±0.43	90.62 ±0.33	92.00 ±0.21	94.49 ±0.39	93.20 ±0.20	94.05 ±0.13	95.60 ±0.02	**97.12** ±0.02

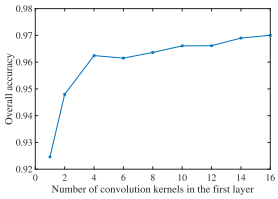
In SACNN, the trained encoder can be used as a feature extractor, and then the extracted features are fed into a widely used classifier, which is the support vector machine (SVM) classifier with RBF kernel. There are two parameters in SVM, namely, C ($C = 2^{-2}, 2^{-1}, \ldots, 2^{10}$) and γ ($\gamma = 2^{-8}, 2^{-7}, \ldots, 2^{3}$). The grid search method is applied to obtain the optimal parameters of SVM.

The experimental results of the above algorithm are listed in Tables VII–IX. It is shown that SACNN outperforms the mathematical method on all datasets. For example, for the Indian Pines dataset, 3D-SACNN can obtain OA, AA, and Kappa of 97.47%, 96.90%, and 97.12%, respectively, with an average of 3% improvement over EAP's experimental results. In Tables VII–IX, CNN-0.15 represents the training of CNN using a training sample ratio of 0.15. Taking the Salinas dataset as an example, as shown in Table VIII, when the training sample ratio is 0.15, CNN and SAE-LR cannot obtain satisfactory classification results. But when the training sample ratio is raised, CNN obtains OA, AA, and Kappa of 96.63%, 98.3%, and 96.25%, respectively, and the SAE-LR obtains OA of 95.17%, AA of 97.94%, and Kappa of 94.63%. This indicates that the absence of labeled samples has a bad influence on the training of conventional CNN and SAE-based feature learning methods. Compared with the CNN and SAE-LR, with a training sample ratio of 0.15, 2D-SACNN can obtain OA of 98.5%, AA of 99.24%, and Kappa of 98.33%, while 3D-SACNN can achieve the best classification results and obtains OA, AA, and Kappa of 99.19%, 99.38%, and 99.1%, respectively. This proves that the proposed SACNN can achieve satisfactory classification results with fewer training samples, which possesses great practical values for HSI classification. Moreover, the proposed method also outperforms the other unsupervised spatial–spectral deep-learning-based methods UDFE, RAE, RCDN, and 3D-USSFL, especially on some categories that are difficult to classify, such as the category Soybean-notill in the Indian Pines dataset, the category Vinyard_untrained in the Salinas dataset, and the category Hardwood in the KSC dataset. This is because the proposed 3-D all convolutional net structure can capture more subtle spatial–spectral features, which benefit the classification performance. It is worth noting that SACNN performs well on both small datasets (Indian Pines) and large datasets (Salinas and KSC).

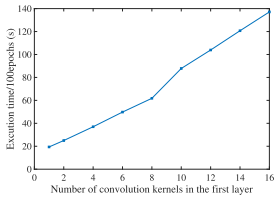
To evaluate the significance of the difference between the classification accuracies of 3D-SACNN and the other competitors, McNemar's test is conducted on the three datasets, and the results are listed in Table X. It can be seen that all of the values in Table X are greater than 1.96, which means that compared with other methods, the improvement of the 3D-SACNN for classification accuracy is statistically significant.

TABLE VIII
CLASSIFICATION RESULTS ON THE SALINAS DATASET

FE method	EAP	CNN -0.15	CNN -0.3	SAE-LR -0.15	SAE-LR -0.5	UDFE	RAE	RCDN	3D-USSFL	2D-SACNN	3D-SACNN
c1	99.89 ±0.12	**100.00** ±0.00	99.93 ±0.09	99.89 ±0.28	**100.00** ±0.00	98.42 ±0.11	98.49 ±0.16	98.37 ±0.20	**100.00** ±0.00	**100.00** ±0.00	99.88 ±0.00
c2	99.91 ±0.10	99.95 ±0.03	99.89 ±0.14	99.59 ±0.11	99.81 ±0.09	99.70 ±0.00	**100.00** ±0.00	**100.00** ±0.00	99.93 ±0.00	99.79 ±0.02	99.90 ±0.01
c3	99.84 ±0.14	96.64 ±0.62	98.00 ±1.13	99.41 ±0.38	99.97 ±0.08	99.62 ±0.13	**100.00** ±0.00	99.80 ±0.00	**100.00** ±0.00	99.84 ±0.03	99.23 ±0.00
c4	99.34 ±0.20	99.23 ±0.21	99.99 ±0.03	98.68 ±0.19	**100.00** ±0.00	99.35 ±0.34	**100.00** ±0.00	99.82 ±0.00	99.90 ±0.10	98.98 ±0.00	99.89 ±0.06
c5	99.35 ±0.29	99.12 ±0.60	99.41 ±0.15	98.31 ±0.53	98.99 ±0.25	99.42 ±0.22	**100.00** ±0.00	99.70 ±0.00	99.65 ±0.11	99.52 ±0.03	99.76 ±0.02
c6	99.95 ±0.06	99.85 ±0.00	99.94 ±0.01	99.98 ±0.02	99.81 ±0.09	99.67 ±0.13	99.71 ±0.00	99.70 ±0.16	**100.00** ±0.00	**100.00** ±0.00	**100.00** ±0.00
c7	99.82 ±0.10	97.61 ±0.21	99.42 ±0.07	99.79 ±0.06	**99.99** ±0.04	99.03 ±0.23	99.70 ±0.14	99.79 ±0.00	99.66 ±0.00	99.22 ±0.04	99.57 ±0.00
c8	93.15 ±0.42	89.15 ±2.57	91.73 ±1.34	89.72 ±0.91	92.77 ±0.76	91.13 ±0.21	97.67 ±0.17	95.40 ±0.10	90.07 ±0.26	96.67 ±0.05	**98.30** ±0.08
c9	99.82 ±0.11	99.81 ±0.03	99.60 ±0.04	99.58 ±0.03	99.71 ±0.09	99.75 ±0.00	99.90 ±0.10	99.81 ±0.00	**100.00** ±0.00	99.96 ±0.00	99.89 ±0.00
c10	97.96 ±0.60	98.64 ±0.26	98.32 ±1.24	98.05 ±0.45	97.78 ±0.54	97.00 ±0.32	99.07 ±0.11	98.67 ±0.00	98.24 ±0.00	**99.71** ±0.02	99.56 ±0.04
c11	99.52 ±0.40	96.91 ±0.61	97.88 ±0.64	96.98 ±0.89	99.76 ±0.45	97.95 ±0.10	98.84 ±0.10	98.74 ±0.09	**99.90** ±0.00	99.77 ±0.00	98.26 ±0.17
c12	99.99 ±0.02	99.91 ±0.03	99.13 ±0.09	98.78 ±0.19	99.84 ±0.22	98.53 ±0.28	99.07 ±0.17	98.96 ±0.00	99.72 ±0.00	**100.00** ±0.00	99.76 ±0.00
c13	99.44 ±0.45	99.56 ±0.11	99.86 ±0.04	99.31 ±0.38	**100.00** ±0.00	97.79 ±0.31	99.02 ±0.27	99.10 ±0.00	**100.00** ±0.00	**100.00** ±0.00	99.74 ±0.00
c14	98.81 ±0.52	98.36 ±0.19	99.34 ±0.05	97.03 ±0.52	99.58 ±0.68	97.70 ±0.57	99.24 ±0.29	99.10 ±0.28	**99.90** ±0.10	99.46 ±0.00	98.25 ±0.01
c15	86.04 ±0.81	87.83 ±3.39	90.93 ±1.78	82.49 ±2.14	79.20 ±1.78	86.37 ±0.20	95.70 ±0.35	92.84 ±0.29	95.20 ±0.26	95.24 ±0.07	**98.16** ±0.09
c16	99.51 ±0.43	98.88 ±0.18	99.42 ±0.10	99.70 ±0.34	99.83 ±0.14	93.44 ±0.78	95.13 ±0.46	94.71 ±0.21	99.60 ±0.16	99.72 ±0.07	**100.00** ±0.00
OA(%)	96.42 ±0.08	95.44 ±0.14	96.63 ±0.06	94.94 ±0.20	95.17 ±0.24	94.82 ±0.27	97.80 ±0.09	96.91 ±0.51	97.13 ±0.13	98.50 ±0.02	**99.19** ±0.01
AA(%)	98.27 ±0.07	97.59 ±0.04	98.30 ±0.11	97.33 ±0.14	97.94 ±0.10	97.68 ±0.25	98.85 ±0.15	98.41 ±0.08	98.86 ±0.06	99.24 ±0.01	**99.38** ±0.01
Kappa ×100	96.01 ±0.09	94.92 ±0.15	96.25 ±0.06	94.37 ±0.22	94.63 ±0.27	94.29 ±0.23	97.52 ±0.11	96.55 ±0.13	96.84 ±0.12	98.33 ±0.02	**99.10** ±0.01

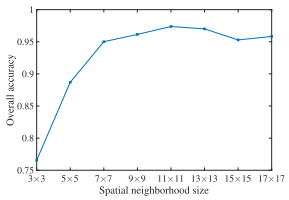


(a)

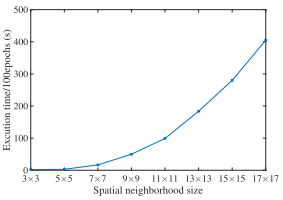


(b)

Fig. 7. Influence of the number of convolution kernels experimental results. (a) Classification results and (b) execution times on different number of convolution kernels.



(a)



(b)

Fig. 8. Influence of different spatial sizes. (a) Classification results and (b) experiment times on different spatial sizes.

D. Parameter Analysis

To further investigate the 3D-SACNN, the parameter analysis of the proposed method is performed on the Indian Pines dataset, and all the experiments are carried out on an NVIDIA Tesla K40C GPU. The quality of extracted features will be affected by the number of convolution kernels. The effect of the number of convolution kernels on OA is presented in Fig. 7. The number of convolution kernels of all layers in the 3D-SACNN is defined as D-2D-4D-2D-D-1, where D represents the number of convolution kernels of the first layer, as shown in the horizontal axis of Fig. 7. Fig. 7(a) indicates that when the number of convolution kernels in the first layer is greater than 4, the OA is improved slowly as the number of convolution kernels increases, but the running time is fast increasing, as shown in Fig. 7(b). It is worth noting that a small number of convolution kernels makes the model too simple to learn all the effective features of inputs. However, when the number of convolution kernels reaches a certain threshold value, the increase of the convolution kernel will not significantly improve the OA, but greatly increases the computation burden and execution time. To balance the quality of extracted features and execution time, the number of convolution kernels in the 3D-SACNN is set to 8-16-32-16-8-1.

Another important hyperparameter is the spatial size of input data, that is, the size of the spatial neighborhood. The

TABLE IX
CLASSIFICATION RESULTS ON THE KSC DATASET

FE method	EAP	CNN -0.15	CNN -0.3	SAE-LR -0.15	SAE-LR -0.5	UDFE	RAE	RCDN	3D-USSFL	2D-SACNN	3D-SACNN
c1	98.64 ±0.70	99.07 ±0.00	99.12 ±0.09	93.73 ±1.90	95.91 ±0.70	95.27 ±0.36	96.62 ±0.51	96.23 ±0.28	96.10 ±0.19	**100.00** ±0.00	98.60 ±0.06
c2	95.41 ±2.13	97.28 ±0.17	98.69 ±0.18	95.01 ±1.61	93.77 ±2.56	93.53 ±0.24	98.48 ±0.35	96.44 ±0.42	95.01 ±0.10	99.70 ±0.05	**100.00** ±0.00
c3	95.64 ±1.04	88.10 ±0.17	85.55 ±0.96	90.94 ±3.90	98.12 ±1.79	92.26 ±0.48	98.94 ±0.00	96.35 ± 0.14	98.57 ±0.23	**99.55** ±0.00	97.22 ±0.75
c4	86.07 ±3.03	79.55 ±0.22	84.01 ±0.67	81.99 ±3.16	82.12 ±3.49	57.90 ±1.60	82.24 ±1.24	83.06 ±0.45	90.11 ±0.32	88.18 ±0.20	**92.37** ±0.63
c5	75.77 ±4.88	58.07 ±0.15	80.35 ±2.43	83.20 ±6.74	90.52 ±6.38	76.63 ±0.80	83.55 ±0.30	86.04 ±2.21	87.32 ±0.25	76.06 ±0.21	**92.68** ±0.45
c6	82.41 ±4.13	95.88 ±0.27	87.09 ±0.44	71.10 ±6.03	81.60 ±2.06	79.62 ±0.31	86.50 ±0.43	90.86 ±0.61	92.22 ±0.37	**100.00** ±0.00	97.23 ±0.49
c7	96.63 ±1.50	84.61 ±1.06	88.47 ±1.54	80.41 ±4.28	86.37 ±6.36	97.31 ±0.36	98.66 ±0.80	98.40 ±0.30	**100.00** ±0.00	**100.00** ±0.00	**100.00** ±0.00
c8	96.48 ±1.32	95.39 ±0.00	99.49 ±0.29	97.06 ±0.84	88.49 ±2.77	93.70 ±0.30	97.42 ±0.36	96.00 ±0.65	94.37 ±0.11	**100.00** ±0.00	94.00 ±0.14
c9	98.89 ±0.51	96.85 ±0.00	99.32 ±0.14	93.49 ±0.88	98.27 ±0.66	96.63 ±0.20	99.03 ±0.10	98.80 ±0.23	98.42 ±0.17	99.77 ±0.00	**99.98** ±0.07
c10	96.65 ±3.00	99.40 ±0.00	99.54 ±0.00	97.14 ±1.63	99.63 ±0.83	97.45 ±0.13	99.72 ±0.25	97.16 ±0.00	97.40 ±0.26	**99.89** ±0.15	99.44 ±0.09
c11	98.23 ±1.50	98.12 ±0.29	99.32 ±0.00	96.00 ±1.39	99.29 ±0.61	99.00 ±0.00	100.00 ±0.00	98.82 ±0.10	99.20 ±0.17	**100.00** ±0.00	98.56 ±0.11
c12	95.77 ±1.72	98.14 ±0.13	97.51 ±0.11	94.14 ±0.73	**99.80** ±0.42	97.23 ±0.15	99.78 ±0.00	96.50 ±0.38	96.91 ±0.29	87.09 ±0.01	99.76 ±0.00
c13	98.52 ±0.83	98.60 ±0.11	99.54 ±0.00	99.81 ±0.24	99.73 ±0.29	99.80 ±0.00	100.00 ±0.00	99.90 ±0.00	100.00 ±0.00	99.92 ±0.06	**100.00** ±0.00
OA(%)	95.63 ±0.23	91.47 ±0.09	96.32 ±0.04	92.99 ±0.72	95.37 ±0.30	93.29 ±0.93	96.52 ±0.14	95.78 ±0.19	96.19 ±0.03	97.15 ±0.02	**98.25** ±0.04
AA(%)	93.47 ±0.47	94.67 ±0.02	93.69 ±0.08	90.31 ±0.93	93.36 ±0.67	90.49 ±0.38	95.46 ±0.33	94.97 ±0.44	95.82 ±0.19	96.17 ±0.03	**97.68** ±0.07
Kappa ×100	95.13 ±0.25	94.07 ±0.03	95.90 ±0.05	92.20 ±0.80	94.85 ±0.33	92.54 ±0.10	96.10 ±0.26	95.32 ±0.28	95.74 ±0.13	96.83 ±0.02	**98.05** ±0.04

TABLE X
MCNEMAR'S TEST RESULTS ON THE INDIAN PINES, SALINAS, AND KSC DATASETS

	EAP	SAE-LR -0.15	SAE-LR -0.5	CNN -0.15	CNN -0.3	UDFE	RAE	RCDN	3D-USSFL	2D-SACNN
Indian Pines	13.98	24.76	15.65	22.62	4.96	14.61	4.72	5.13	4.88	4.54
Salinas	27.50	38.89	33.20	38.25	24.64	40.78	13.59	20.27	17.95	9.47
KSC	9.41	11.02	6.07	10.11	9.89	11.36	5.73	7.19	6.12	4.17

number of input spectrum bands is fixed, but the input spatial neighborhood size is changeable. In the experiments, the size of each batch is set as 64. Experiments are carried out by extracting features on different spatial sizes: 3×3, 5×5, 7×7, 9×9, 11×11, 13×13, 15×15, and 17×17. The influence of different spatial sizes on the classification results is shown in Fig. 8(a). When the spatial size is too small, such as 3×3 and 5×5, the input data cannot provide sufficient spatial information. Thus, an increase in spatial sizes can enrich spatial information and benefit classification performance. However, because the classification of HSI is pixel-level classification, an oversized neighborhood may cause a problem, that is, there will be too many other class pixels in the neighborhood of the target pixel, especially when it comes to edges of a certain class. From Fig. 8(b), it can be seen that the running time is increasing exponentially with the increase of spatial size. For the Indian Pines dataset, the OA can achieve a maximum of 97.38% when spatial size is set to 11×11, which is an optimal tradeoff between classification performance and execution time.

V. CONCLUSION

In this article, we proposed a novel FE method that is based on the 3-D all CNN architecture. Experimental results have shown that 3-D convolutional operation was more suitable for learning effective spatial–spectral features in HSIs. To alleviate the heavy dependence of the proposed 3-D all convolutional nets, a novel end-to-end training framework was designed based on the encoder–decoder architecture. Specifically, the encoder subnetwork consisted of 3-D multilayer all convolutional kernels and the decoder subnetwork consisted of 3-D all deconvolutional kernels, which were symmetric in architecture. By using the reconstruction error, the proposed feature extractor, which is the encoder subnetwork, can be effectively trained in an unsupervised style. Moreover, instead of fixed pooling operations, the proposed method can adaptively learn the downsampling and upsampling strategy, which benefits the training process. In addition, multilayer and multiscale features are considered in the FE process, and experimental results have shown that they can significantly improve classification performance.

In the future, further study will be implemented on the optimization of hyperparameters in the proposed networks, such as the number of layers, learning rates, decay coefficients, and so on, which can reduce the manual intervention during the design of networks and enable the networks to learn the optimal hyperparameters adaptively from the view of data driven. Moreover, we will explore the application of the proposed method in more subtle analysis tasks, such as objective detection, anomaly detection, and saliency analysis.

REFERENCES

[1] C.-I. Chang, *Hyperspectral Data Exploitation: Theory and Applications*. Hoboken, NJ, USA: Wiley, 2007.

[2] P. Ghamisi *et al.*, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 37–78, Dec. 2017.

[3] P. S. Thenkabail and J. G. Lyon, *Hyperspectral Remote Sensing of Vegetation*. Boca Raton, FL, USA: CRC, 2016.

[4] E. Adam, O. Mutanga, and D. Rugege, "Multispectral and hyperspectral remote sensing for identification and mapping of wetland vegetation: A review," *Wetlands Ecol. Manage.*, vol. 18, no. 3, pp. 281–296, Jun. 2010.

[5] A. C. Watts, V. G. Ambrosia, and E. A. Hinkley, "Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use," *Remote Sens.*, vol. 4, no. 6, pp. 1671–1692, Jun. 2012.

[6] L. Goddijn-Murphy, S. Peters, E. Van Sebille, N. A. James, and S. Gibb, "Concept for a hyperspectral remote sensing algorithm for floating marine macro plastics," *Martine Pollut. Bull.*, vol. 126, pp. 255–262, Jan. 2018.

[7] L. M. Dale *et al.*, "Hyperspectral imaging applications in agriculture and agro-food product quality and safety control: A review," *Appl. Spectrosc. Rev.*, vol. 48, no. 2, pp. 142–159, Jan. 2013.

[8] H. Huang, G. Shi, H. He, Y. Duan, and F. Luo, "Dimensionality reduction of hyperspectral imagery based on spatial–spectral manifold learning," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2604–2616, Jun. 2020, doi: 10.1109/TCYB.2019.2905793.

[9] P. Bajcsy and P. Groves, "Methodology for hyperspectral band selection," *Photogramm. Eng. Remote Sens.*, vol. 70, no. 7, pp. 793–802, Jul. 2004.

[10] J. Li *et al.*, "Multiple feature learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1592–1606, Mar. 2015.

[11] Y. Yuan, J. Lin, and Q. Wang, "Hyperspectral image classification via multitask joint sparse representation and stepwise MRF optimization," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2966–2977, Dec. 2016.

[12] F. Luo, B. Du, L. Zhang, L. Zhang, and D. Tao, "Feature learning using spatial-spectral hypergraph discriminant analysis for hyperspectral image," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2406–2419, Jul. 2019.

[13] L. Zhang, Q. Zhang, B. Du, X. Huang, Y. Y. Tang, and D. Tao, "Simultaneous spectral-spatial feature selection and extraction for hyperspectral images," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 16–28, Jan. 2018.

[14] Z. Feng, S. Yang, M. Wang, and L. Jiao, "Learning dual geometric low-rank structure for semisupervised hyperspectral image classification," *IEEE Trans. Cybern.*, early access, Jan. 4, 2019, doi: 10.1109/TCYB.2018.2883472.

[15] A. J. Izenman, "Linear discriminant analysis," in *Modern Multivariate Statistical Techniques*. New York, NY, USA: Springer, 2013, pp. 237–280.

[16] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1027–1061, May 2007.

[17] H.-T. Chen, H.-W. Chang, and T.-L. Liu, "Local discriminant embedding and its variants," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. San Diego, CA, USA, Jun. 2005, pp. 846–853.

[18] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot, "Hyperspectral remote sensing data analysis and future challenges," *IEEE Geosci. Remote Sens. Mag.*, vol. 1, no. 2, pp. 6–36, Jun. 2013.

[19] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1349–1362, Mar. 2016.

[20] J. Shlens, "A tutorial on principal component analysis," 2014. [Online]. Available: arXiv:1404.1100.

[21] M. Dalla Mura, A. Villa, J. A. Benediktsson, J. Chanussot, and L. Bruzzone, "Classification of hyperspectral images by using extended morphological attribute profiles and independent component analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 3, pp. 542–546, May 2011.

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

[23] M. Li and S. Narayanan, "Robust talking face video verification using joint factor analysis and sparse representation on gmm mean shifted supervectors," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Prague, Czechia, Jul. 2011, pp. 1481–1484.

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

[25] N. Alajlan, Y. Bazi, F. Melgani, and R. R. Yager, "Fusion of supervised and unsupervised learning for improved classification of hyperspectral images," *Inf. Sci.*, vol. 217, pp. 39–55, Dec. 2012.

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

[27] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing System*. Cambridge, MA, USA: MIT Press, 2007, pp. 153–160.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing System*. New York, NY, USA: Assoc. Comput. Mach., 2012, pp. 1097–1105.

[29] X. X. Zhu *et al.*, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017.

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

[31] X. Ma, H. Wang, and J. Geng, "Spectral–spatial classification of hyperspectral image based on deep auto-encoder," *IEEE J. Sel. Topics Appl. Earth Observ.*, vol. 9, no. 9, pp. 4073–4085, Sep. 2016.

[32] T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *Proc. IEEE Int. Conf. Image Process.*, Paris, France, Oct. 2014, pp. 5132–5136.

[33] P. Zhong, Z. Gong, S. Li, and C.-B. Schönlieb, "Learning to diversify deep belief networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 6, pp. 3516–3530, Jun. 2017.

[34] Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2381–2392, Jun. 2015.

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

[36] 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.*, vol. 4, no. 2, pp. 22–40, Jun. 2016.

[37] 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.*, vol. 54, no. 8, pp. 4544–4554, Aug. 2016.

[38] 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 Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.

[39] Z. Zhong, J. Li, Z. Luo, and M. Chapman, "Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 847–858, Feb. 2018.

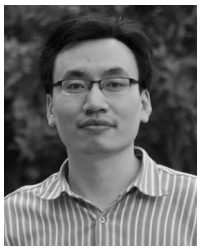
[40] M. Zhang, W. Li, Q. Du, L. Gao, and B. Zhang, "Feature extraction for classification of hyperspectral and LiDAR data using patch-to-patch CNN," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 100–111, Jan. 2020, doi: 10.1109/TCYB.2018.2864670.

[41] W. Li, G. Wu, F. Zhang, and Q. Du, "Hyperspectral image classification using deep pixel-pair features," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 2, pp. 844–853, Feb. 2017.

[42] M. Zhang, M. Gong, Y. Mao, J. Li, and Y. Wu, "Unsupervised feature extraction in hyperspectral images based on wasserstein generative adversarial network," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 5, pp. 2669–2688, May 2019.
[43] L. Mou, P. Ghamisi, and X. X. Zhu, "Unsupervised spectral–spatial feature learning via deep residual conv–deconv network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 1, pp. 391–406, Jan. 2018.
[44] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision ECCV*. Cham, Switzerland: Springer, 2014, pp. 818–833.
[45] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2005. [Online]. Available: arXiv:1511.06434.
[46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[47] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014. [Online]. Available: arXiv:1412.6806.
[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826.
[50] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: arXiv:1412.6980.
[51] P. R. Marpu, M. Pedergnana, M. D. Mura, J. A. Benediktsson, and L. Bruzzone, "Automatic generation of standard deviation attribute profiles for spectral–spatial classification of remote sensing data," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 293–297, Mar. 2013.
[52] X. Zhang, Y. Liang, C. Li, N. Huyan, L. Jiao, and H. Zhou, "Recursive autoencoders-based unsupervised feature learning for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 11, pp. 1928–1932, Nov. 2017.
[53] S. Mei, J. Ji, Y. Geng, Z. Zhang, X. Li, and Q. Du, "Unsupervised spatial–spectral feature learning by 3D convolutional autoencoder for hyperspectral classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 6808–6820, Sep. 2019.

Mingyang Zhang (Member, IEEE) received the B.S. degree in automation and the Ph.D. degree in pattern recognition and intelligent systems from Xidian University, Xi'an, China, in 2012 and 2018, respectively.

Since 2018, he has been a Lecturer with the School of Electronic Engineering, Xidian University. His research interests include computational intelligence and remote sensing image understanding.



Maoguo Gong (Senior Member, IEEE) received the B.Eng and Ph.D. degrees from Xidian University, Xi'an, China, in 2003 and 2009, respectively.

Since 2006, he has been a Teacher with Xidian University. He was promoted to an Associate Professor in 2008 and a Full Professor in 2010, both with exceptional admission. He has published over 100 papers in journals and conferences, and holds over 20 granted patents as the first inventor. His research interests are broadly in the area of computational intelligence, with applications to optimization, learning, data mining, and image understanding.

Dr. Gong was a recipient of the Prestigious National Program for Support of Top-Notch Young Professionals (selected by the Central Organization Department of China), the Excellent Young Scientist Foundation (selected by the National Natural Science Foundation of China), the New Century Excellent Talent in University (selected by the Ministry of Education of China), the Young Teacher Award by the Fok Ying Tung Education Foundation, and the National Natural Science Award of China. He is leading or has completed over ten projects as the Principle Investigator, funded by the National Natural Science Foundation of China, and the National Key Research and Development Program of China. He is the Executive Committee Member of the Chinese Association for Artificial Intelligence, a Senior Member of the Chinese Computer Federation, and an Associate Editor or the Editorial Board Member for over five journals, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Haibo He (Fellow, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1999 and 2002, respectively, and the Ph.D. degree in electrical engineering from Ohio University, Athens, OH, USA, in 2006.

He is currently the Robert Haas Endowed Chair Professor with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingstown, RI, USA.

Dr. He received the IEEE International Conference on Communications Best Paper Award in 2014, the IEEE CIS Outstanding Early Career Award in 2014, and the National Science Foundation CAREER Award in 2011. He was the General Chair of the IEEE Symposium Series on Computational Intelligence in 2014. He is currently the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Shengqi Zhu (Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from Xidian University, Xi'an, China, in 2005 and 2010, respectively.

He is currently a Professor with the National Laboratory of Radar Signal Processing, Xidian University. His research interests include space–time adaptive processing, multiple-input–multiple-output radar, SAR ground moving target indication, and sparse signal processing.

Prof. Zhu was awarded the Young Scientists Award for Excellence in Scientific Research by the International Union of Radio Science from 2011 to 2014. He is currently an Associate Editor of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS.