Spatial Sequential Recurrent Neural Network for Hyperspectral Image Classification

Xiangrong Zhang, *Senior Member, IEEE*, Yujia Sun, Kai Jiang, Chen Li, Licheng Jiao, *Fellow, IEEE*, and Huiyu Zhou

***Abstract*—In hyperspectral image processing, classification is one of the most popular research topics. In recent years, research progress made in deep-learning-based hierarchical feature extraction and classification has shown a great power in many applications. In this paper, we propose a novel local spatial sequential (LSS) method, which is used in a recurrent neural network (RNN). Using this model, we can extract local and semantic information for hyperspectral image classification. First, we extract low-level features from hyperspectral images, including texture and differential morphological profiles. Second, we combine the low-level features together and propose a method to construct the LSS features. Afterwards, we build an RNN and use the LSS features as the input to train the network for optimizing the system parameters. Finally, the high-level semantic features generated by the RNN is fed into a softmax layer for the final classification. In addition, a nonlocal spatial sequential method is presented for the recurrent neural network model (NLSS-RNN) to further enhance the classification performance. NLSS-RNN finds nonlocal similar structures to a given pixel and extracts corresponding LSS features, which not only preserve the local spatial information, but also integrate the information of nonlocal similar samples. The experimental results on three publicly accessible datasets show that our proposed method can obtain competitive performance compared with several state-of-the-art classifiers.**

***Index Terms*—Deep learning, high-level semantic feature, hyperspectral image (HSI) classification, low-level feature, recurrent neural network (RNN).**

I. Introduction

OVER the last decade, hyperspectral image (HSI) processing has played an important role in many applications such as the assessment of environmental damage, growth regulation, land-use monitoring, urban planning, and reconnaissance [1], [2]. Among these applications, HSI classification aims at classifying image pixels into multiple categories, and many classification methods have achieved promising results.

Among the existing classification methods, pixel-based methods are one of the popularly used methods, based on the individual pixels' inherent information. Due to their performance in a high-dimensional space, support vector machine (SVM) classifiers [3], [4] have become one of the most commonly used and powerful classifiers for pixel-based HSI classification. For example, Chen *et al.* [5], [6] used sparse representation for HSI classification, which regards a pixel as an approximated linear combination of a few dictionary atoms, drawing great attention from the community. Recently, researchers have found that HSIs usually have strong local space consistency [7], [8], so a few methods take both spectral and spatial (spectral–spatial) information into consideration and have achieved promising success. Chen *et al.* proposed a joint sparse representation classifier based on the joint sparsity model [9] containing both spectral and spatial information. Li *et al.* [10] utilized the standard collaborative representation mechanism [11], [12] to support HSI classification. Some methods actually are semisupervised, achieving better classification results when the number of the training samples is small [13], [14]. However, most of these classification methods mentioned above are based on low-level features, e.g., spectral, texture [15], and morphological features [16]. Developing a more powerful and discriminative classifier using the semantics of HSIs is still a challenge.

Nowadays, as an important technique in machine learning, deep learning architectures have shown promising performance in many research areas, including classification or regression for images [17], [18], language [19], speech [20], and remote sensing [21]–[23], where they have usually outperformed traditional methods in large-scale datasets. A deep learning architecture is a multilayer stack of basic modules, all (or most) of which are subject to learning, and many of which look for nonlinear input–output mappings [24]. Based on its hierarchical feature representation mechanism, deep learning methods are able to extract abstract concepts through stacked layers composed of multiple nonlinear transformations [25]. Typically, deep learning architectures include stacked (denoising) autoencoders (SAEs) [26], deep belief networks (DBNs) [27], deep Boltzmann machines [28], convolutional neural networks (CNNs) [29], recursive neural networks [30], recurrent neural networks (RNNs) [31], [32],

Manuscript received February 16, 2018; revised May 10, 2018; accepted May 28, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61772400, Grant 61501353, Grant 61772399, Grant 91438201, and Grant 61573267 and in part by the joint fund of the Equipment Research of Ministry of Education. The work of H. Zhou was supported in part by the U.K. Engineering and Physical Sciences Research Council under Grant EP/N508664/1, Grant EP/R007187/1, and Grant EP/N011074/1, and in part by the Royal Society-Newton Advanced Fellowship under Grant NA160342. *(Corresponding author: Xiangrong Zhang.)*

X. Zhang, Y. Sun, K. Jiang, and L. Jiao are with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, School of Artificial Intelligence, Xidian University, Xi'an 710071, China (e-mail: xrzhang@ieee.org; yjsun@stu.xidian.edu.cn; jk123vip@gmail.com; lchjiao@mail.xidian.edu.cn).

C. Li is with Xi'an Jiaotong University, Xi'an 710049, China (e-mail: cli@xjtu.edu.cn).

H. Zhou is with the University of Leicester, Leicester LE1 7RH, U.K. (e-mail: h.zhou@ecit.qub.ac.uk).

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

Digital Object Identifier 10.1109/JSTARS.2018.2844873

and long short-term memory (LSTM) [33], [34]. For specific tasks, these models can efficiently extract high-level abstract features from low-level features in different ways. For example, in computer vision, 2-D color images or 3-D videos can be fed into a CNN composed of two types of layers, convolutional and pooling layers, to generate high-level features. Overall, deep learning architectures provide a general-purpose learning procedure that can automatically fuse low-level features together to exploit the property of the input signals and obtain higher level features by composing lower level ones.

Deep learning techniques can be used to map different levels of abstractions from images and combine them to form high-level features. Therefore, it is possible to use a deep learning architecture as an effective approach to optimally fuse low-level features of an HSI to produce more abstract and powerful high-level features describing the image itself for different classification tasks. There have been some interesting achievements deploying deep learning architectures for processing HSIs. In [35], Chen *et al.* verified the strength of the SAE for spectral information-based classification. One year later, they proposed a new feature extraction and classification framework for hyperspectral data analysis based on the DBN [36]. In [37], Li *et al.* also introduced a DBN and a restricted Boltzmann machine for HSI processing and feature extraction. Hu *et al.* employed a CNN to classify HSIs in the spectral domain [38]. In [39], Mou *et al.* applied an RNN to classify HSIs directly in the spectral domain. In [40], Rubwurm and Korner used LSTM networks to extract temporal characteristics from a sequence of SENTINEL 2A observations. Lyu *et al.* proposed a method that relies on an improved LSTM model to acquire and record the change information of remote sensing data [41]. In [42], Liu *et al.* employed bidirectional-convolutional LSTM to classify HSIs based on spectral–spatial features. In these methods, deep learning architectures have shown powerful performance and achieved competitive results.

Although the methods mentioned above have shown a great capability in dealing with HSI features, some important issues need to be solved. One of these methods' shortcomings is that most of the methods directly deploy the spectral features of the original image as the input, which may be insufficient to describe the image. Extracting other low-level HSI features that can be more powerful and discriminative than the original spectral features will benefit both high-level feature extraction and classification. In terms of spatial features, these methods often combine the neighboring pixels' spectral features together into a vector as the input of the deep learning network after having reduced the spectral feature dimensionality, ignoring the relationship between the neighboring pixels. Though most of the local space areas are homogeneous in an HSI, simply selecting all the pixel information in a local area creates other issues. For example, the pixels in a local window may have significant appearance variations, and this probably affects the final classification accuracy.

To deal with the problems mentioned above, in this paper, we introduce a novel method for HSI classification based on the RNN, named local spatial sequential recurrent neural network (LSS-RNN), which uses powerful low-level features instead of the original spectral features. The proposed scheme not only extracts local spatial information of an HSI, but also exploits the relationship between the local regions to strengthen the "good" features and reduce the impact of the "bad" features. Our work mainly focuses on using an RNN, which is one of the widely used deep-architecture-based models in natural language processing and speech recognition fields for hierarchically generating high-level contextual features from low-level features. First, we extract traditional useful texture and morphological features as the low-level ones. Afterwards, in our method, local spatial sequential (LSS) features are used as the input of the RNN architecture, which not only utilizes spectral and spatial information, but also exploits the relationship between the local neighboring pixels effectively. As we know, an RNN model has the ability to maintain the sequence information, and our LSS features in the structure and concept are suitable as the input vector of the RNN. For each local image area, we enhance the influence of the supportive pixels, while reducing the effect of other nonsupportive pixels. Finally, high-level contextual features learned from LSS-RNN are fed into the softmax layer of the RNN model for classification. In summary, based on the conventional RNN model, we provide a generic framework named LSS-RNN to extract low-level features and combine them together with local space sequence information to form powerful high-level contextual features for achieving better classification.

On the basis of LSS, inspired by nonlocal spatial features, a hyperspectral classification method based on the LSS-RNN and the nonlocal spatial sequential recurrent neural network (NLSS-RNN) is proposed. Besides the local information, NLSS uses a number of samples with similar structures from the perspective of the whole HSI data. Both local and nonlocal similar pixels are adopted to construct the sequential features of the RNN. Compared with the local features, NLSS can integrate the local and nonlocal information of the samples, which is effective for HSI classification tasks. The framework of the proposed NLSS-RNN is shown in Fig. 1.

The remainder of this paper is organized as follows. Section II briefly introduces deep learning and the traditional RNN model. In Section III, the proposed classification framework is introduced, including low-level feature extraction, LSS and NLSS feature constructions, and LSS-RNN and NLSS-RNN. Experimental results on three datasets are shown in Section IV. Conclusion is given in the final section.

II. Feedforward Neural Networks and RNNs

A. Feedforward Neural Networks

As a biologically inspired classification algorithm, a feedforward neural network [43] is regarded as the first and simplest artificial neural network in the community. As shown in Fig. 2(a), the simplest feedforward neural network comprises of an input layer, a hidden layer, and an output layer, and each layer contains several units. For a training example (x, y), neural networks give a way of defining a complex nonlinear form of the output vector $\hat{y}$ through

$$h = \sigma_h(W_h x + b_h) \tag{1}$$

$$\hat{y} = \sigma_y(W_y h + b_y) \tag{2}$$

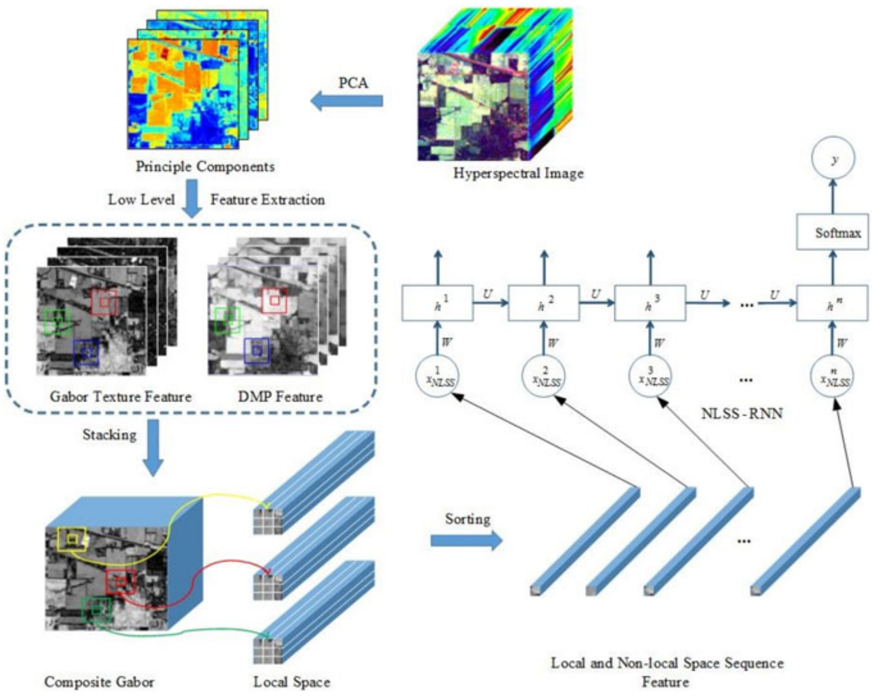


Fig. 1. Framework of the proposed NLSS-RNN method.

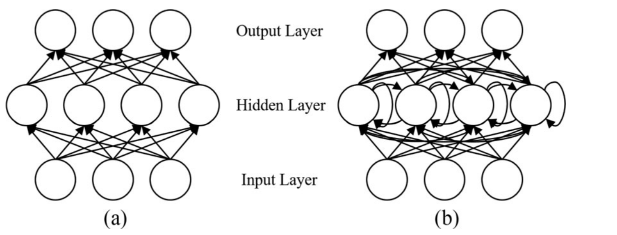


Fig. 2. Network architectures. (a) Feedforward neural network. (b) RNN.

where h is the hidden layer state, $\hat{y}$ denotes the prediction result of the input vector x, W_h and W_y correspond to the input-to-hidden and hidden-to-output weight matrices, respectively, b_h and b_y denote the bias of the hidden and output units, respectively, and σ_h and σ_y denote the activation functions.

The cost function with respect to the input vector x is defined as

$$L(y, \hat{y}) = \frac{1}{2}\|y - \hat{y}\|^2. \tag{3}$$

Given a training set of M examples, the overall cost function is defined as

$$L(y, \hat{y}) = \frac{1}{2M}\sum_{i=1}^{M}\|y_i - \hat{y}_i\|^2. \tag{4}$$

The ultimate goal is to minimize $L(y, \hat{y})$ as a function of parameters W_h, W_y, b_h, and b_y, and stochastic gradient descent can be used to efficiently solve this problem. Normally, a backpropagation technique [44] is used to train the system and optimize these parameters.

B. RNNs

RNNs are feedforward neural networks augmented by the inclusion of edges that span over adjacent time steps, introducing a notion of time to the model [45]. Unlike feedforward neural networks that can only map from a single input vector to an output vector, an RNN can, in principle, map from the entire history of the previous inputs to each output [46]. The idea behind the RNN is to make use of sequential information, and the reason why it is called "recurrent" is that this architecture performs the same task for every element of a sequence with the output depending on the previous computational results. This means the recurrent connections allow a memory of the previous inputs to persist in the network's internal state and thereby influence the network output [45].

As shown in Fig. 2(b), for a simple RNN, let our input x be a sequence whose length is T, $x = \{x_1, x_2, \ldots, x_T\}$, and each item x_t is a feature vector. At time step t, given the previous hidden layer state h_{t-1}, the current hidden layer state h_t and the output layer state y_t can be calculated by

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{5}$$

$$y_t = \sigma_y(W_y h_t + b_y) \tag{6}$$

where W_h and W_y denote the input-to-hidden and hidden-to-output weight matrices, respectively, U_h is the matrix of the recurrent weights between the hidden layer and itself at two adjacent time steps, b_h and b_y are the biases, and σ_h and σ_y denote the activation functions.

At each time step, the input is propagated in a standard feedforward fashion, and then, a learning rule is applied. The back connections lead to the result that the context units always maintain a copy of the previous values of the hidden units (since they propagate over the connections before the learning rule is applied). Thus, the network can maintain a state, allowing it to perform such tasks as sequence prediction that are beyond the power of standard multilayer perception.

III. Proposed Method

In this section, we will introduce our proposed LSS-RNN and NLSS-RNN methods based on the improvement of the traditional RNN for HSI classification. Briefly speaking, our approach is mainly composed of four parts: low-level feature extraction, LSS and nonlocal spatial sequential (NLSS) feature construction, training of the RNN, and final classification. We introduce these parts step by step in the following.

A. Low-Level Feature

Based on the fact that deep learning architectures are able to extract robust abstract concepts from low-level features [47] through their layer-by-layer feature extraction mechanism, the extraction of powerful and discriminative low-level features of the original HSI plays an important role in the classification stage.

Considering intrinsic properties of an HSI, combining multiple features together may enhance the discriminability and be helpful to the classification task. In this paper, we extract two low-level features widely used in the HSI processing domain: the first one is the HSI texture feature, extracted by Gabor filtering [48], [49], and the other is the differential morphological profile (DMP) feature [50]–[53], representing the shape information of the HSI.

1) Gabor Texture Features: The Gabor transform fulfills human visual characteristics and has a unique advantage to extract texture features [54]. The kernel function of a 2-D Gabor filter is shown as follows:

$$\varphi(x', y', f, \theta, \gamma, \sigma) = \exp\left(-\frac{x''^2 + \gamma^2 y''^2}{2\sigma^2}\right)\cos(2\pi f x'' + \phi) \tag{7}$$

where $x'' = x'\cos\theta + y'\sin\theta$, $y'' = -x'\sin\theta + y'\cos\theta$, and x' and y' represent the coordinates of a pixel in the image. f and θ indicate the frequency and rotation of the sinusoidal plane wave, respectively. ϕ is the phase of the Gabor function, and σ and γ are the radius and orientation angle of the Gaussian, respectively.

To sufficiently extract the texture information from an HSI, a set of Gabor filters are constructed by setting different frequencies and rotations. Then, for a 3-D HSI dataset, $D \in R^{K_1 \times K_2 \times B}$, where K_1 and K_2 are length and width, respectively, and B indicates the number of the spectral bands. After having convolved D with the family of Gabor filters with various frequencies and rotations, the Gabor texture features $X_{\text{texture}} \in R^{K_1 \times K2 \times l_t}$, where l_t is the length of the Gabor texture feature vector, can be obtained.

2) DMP Shape Features: DMP [50], [53] features represent the shape information of an HSI. Similar to the extended morphological profiles [51], the principal components (PCs) of the HSI are used as the base images for the construction of MPs in the DMP. A series of morphological opening and closing operations with a family of structuring elements (SEs) of increasing sizes are performed to remove small bright or dark details, while maintaining the overall shape features.

For an image I, $\gamma^{\text{SE}}(I)$ and $\phi^{\text{SE}}(I)$ are the morphological opening and closing by reconstruction with the SE. MPs are defined by a series of SEs with increasing sizes

$$\text{MP}_\gamma = \{\text{MP}_\gamma^\lambda(I) = \gamma^\lambda(I), \forall\lambda \in [0, n]\} \tag{8}$$

$$\text{MP}_\phi = \{\text{MP}_\phi^\lambda(I) = \phi^\lambda(I), \forall\lambda \in [0, n]\} \tag{9}$$

where λ is the radius of the disk-shaped SE and $\lambda^0(I) = \phi^0(I) = I$. Then, DMPs are defined as the vectors where the measures of the slopes of the MPs are stored for every step of an increasing SE series

$$\text{DMP}_\gamma = \left\{\text{DMP}_\gamma^\lambda(I) = \left|\text{MP}_\gamma^\lambda(I) - \text{MP}_\gamma^{\lambda-1}(I)\right|, \lambda \in [1, n]\right\} \tag{10}$$

$$\text{DMP}_\phi = \left\{\text{DMP}_\phi^\lambda(I) = \left|\text{MP}_\phi^\lambda(I) - \text{MP}_\phi^{\lambda-1}(I)\right|, \lambda \in [1, n]\right\}. \tag{11}$$

In our experiments, the first five PCs of the HSI dataset are selected for extracting DMP shape features and deriving the parameter λ that is set to $(2, 4, 6, 8, 10)$. The final DMP shape feature $X_{\text{shape}} \in R^{K_1 \times K_2 \times l_s}$ is constructed by concatenating DMP_γ and DMP_ϕ together, where l_s is the length of the DMP shape feature vector. Experimentally, l_2 is set to be 50 for better performance.

B. LSS Features

The two low-level feature matrices do not contain enough information for further processing. It has been widely recognized that properly combining multiple features together may result in good classification performance [55]. In this paper, after having got the texture feature X_{texture} and DMP shape feature X_{shape}, we stack them together to derive a more powerful low-level feature matrix $X = [X_{\text{texture}}\ X_{\text{shape}}] \in R^{K_1 \times K_2 \times l}$, where $l = l_t + l_s$ is the length of the composite low-level feature vector.

Using the local homogenous property, we assume that the pixels in a local region usually belong to the same class. A number of methods have used local spatial information [5], [6], [56]. In this paper, we collect the spatial information using a $w \times w$ neighboring region of every pixel in the training and testing examples. For the ease of understanding, w is regarded as the local spatial window size.

The RNN can process the sequential inputs by having a recurrent hidden state, whose activation at each step depends on that of the previous step. For the hidden layer state h^t at time t, we use h^t as the memory of the network, which captures the information of what happens in all the previous time steps. In order to deal with the problem that different pixels within the same local area may affect the classification result, we propose a method to construct the LSS features, which efficiently exploit the relationship between the local pixels.

For the pixels of a small region, we sort them by their individual importance. The most important pixel (the "best" one) is set to be the first time input vector, and the least important pixel (the "worst" one) is set to be the last time input vector. Actually, we treat the concept "importance" as the degree of the similarity between a pixel and the given pixel that we want to classify, hereby the center pixel of the rectangle region. The final pixel sequence is called the LSS features.

More specifically, for a center pixel $x = (x_1, x_2, \ldots, x_l) \in R^l$), the $w \times w$ neighboring region contains n pixels, $x_{\text{local}} = \{x^1, x^2, \ldots, x^n\}$, where $n = w \times w$. $x_{\text{local}} \in R^{l \times n}$ can also be seen as a 2-D feature matrix of x consisting of all n neighbor pixels. We use the Euclidean distance $I_{(x,x^i)}$ to measure the similarity between the center pixel x and the neighbor pixel $x^i (i = 1, \ldots, n)$

$$I_{(x,x^i)} = \sqrt{(x_1 - x_1^i)^2 + (x_2 - x_2^i)^2 + \cdots + (x_l - x_l^i)^2} \tag{12}$$

which represents the importance of the pixel x^i ("good" or "bad").

Then, all the pixels in x_{local} are sorted by $I_{(x,x^i)}$ to obtain the LSS feature $x_{\text{LSS}} = \{x^1_{\text{LSS}}, x^2_{\text{LSS}}, \ldots, x^n_{\text{LSS}}\}$; $x_{\text{LSS}} \in R^{l \times n}$ is a sequence feature constructed by n composite low-level feature vectors. So, the importance of each pixel decreases gradually in the feature sequence, and the first input vector of x_{LSS} is the center pixel x.

RNN models are good at dealing with the sorted sequence features. The pixel x is the most important one, and it is set to the first time input vector. According to the RNN property mentioned above, x influences the whole sequence. The pixel most similar to x is set to the second time input vector, affecting the rest sequence besides x, and so on. For the other pixels, the less important they are, the latter input vector they are set to (the less they can affect the classification result). So, the positive influence of the "good" ones is strengthened, while the negative effects of the "bad" ones are reduced. In other words, the concept of the time sequence has been applied to the local spatial sequence, which is the LSS feature's contribution to the proposed model.

C. LSS-RNN

After having got the LSS features, it is time to construct an RNN model so that the training examples can be fed into the input layer to train the model. In this paper, we use a fully connected RNN architecture as our basic RNN model. The number of the time steps is set to n, which is exactly the number of the pixels in a local spatial window. The output dimension of the internal projections is set to l, which is the dimension of the LSS feature matrix, also known as the low-level feature vector's length. Then, the LSS feature x_{LSS} can be fed into the RNN model, and each low-level feature vector in x_{LSS} corresponds to an input vector of time steps in the RNN. For time step t with the corresponding input feature vector x^t_{LSS}, $t = 1, \ldots, n$, the hidden layer state h^t and the output layer state y^t can be calculated by

$$h^t = \sigma_h(W_h x^t_{\text{LSS}} + U_h h^{t-1} + b_h) \tag{13}$$

$$y^t = \sigma_y(W_y h^t + b_y). \tag{14}$$

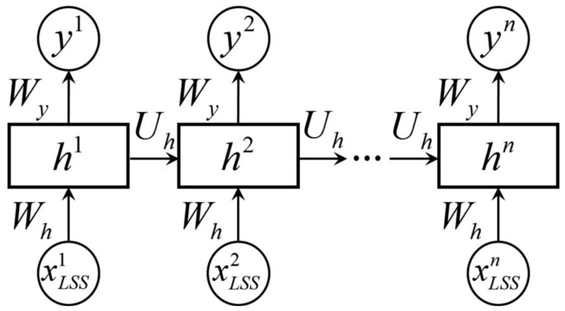


Fig. 3. How x_{LSS} works in LSS-RNN.

In recent research on deep learning, there is a consensus that for deep neural networks, rectified linear units (ReLUs) are easier to train than the sigmoid or tanh units that were used for many years [57]. In our work, ReLU is selected as the input-to-hidden activation function σ_h. Also, inspired by the approach reported in [58], the recurrent weight matrix is initialized to be the identity matrix, and the biases are initialized to be zero. In this way, the RNN is composed of the ReLUs that are relatively easy to train, while modeling the long-range dependencies.

Then, the last time hidden layer output is a high-level contextual feature, which can be fed into a softmax layer to complete the classification task. So, (14) is transformed to

$$y^n = \text{softmax}(W_y h^n + b_y) \tag{15}$$

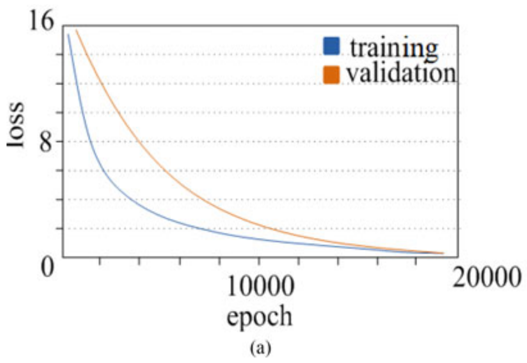
where y^n is the classification result. How x_{LSS} works in LSS-RNN is shown in Fig. 3, which makes it easier to understand. As shown in Fig. 3, $x^t_{\text{LSS}}, t \in \{1, \ldots, n\}$ represents the tth low-level feature vector of the low-level feature matrix x_{LSS}. We use the traditional cross entropy loss as the loss function. Then, backpropagation through the (BPTT) method [46] is applied to train the network over time steps, and this results in conceptual simplicity and efficiency.

In this paper, we use the ReLU as the activation function. The learning rate is set to 0.0001, and the batch size is set to 100. We provide training and validation loss curves, as shown in Fig. 4, to demonstrate the convergence of the algorithm. From Fig. 4, we can see that the loss values on the training set and the loss on the validation set gradually decrease and finally converge, showing that the RNN works well on both the training and validation sets.

D. LSS and NLSS Features

Besides the local spatial information, the spatial information from nonlocal regions is also important for HSI classification. As we know, there are some pixels that are characteristically similar to the original pixel, but locate at different regions. Exploring the nonlocal spatial information will further improve the discriminability of the learned features. Instead of extracting features from the local regions around the sample only, we propose the feature extraction method of using LSS and NLSS features for better classification performance.

Specifically, for a given center pixel x, we first find K nonlocal neighbors $x_{\text{nonlocal}} = \{x^1, x^2, \ldots, x^K\}$, which are determined by the K nearest neighbor (KNN) algorithm according



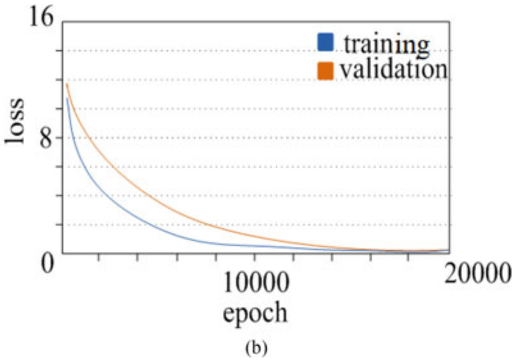


Fig. 4. Training and validation loss curves. (a) For Indian Pines. (b) For University of Pavia.

to the extracted low-level features. For every sample in x_{nonlocal}, the corresponding LSS feature is extracted x^i_{LSS}, $i = 1, \ldots, K$. Stacking the LSS features of the original sample and the K LSS of the nonlocal samples together, we get the NLSS feature $x_{\text{NLSS}} = \{x^1_{\text{LSS}}, x^2_{\text{LSS}}, \ldots, x^n_{\text{LSS}}\}, x_{\text{LSS}} \in R^{l\times K}, x_{\text{NLSS}} \in R^{l\times n'}$, where l denotes the length of the positive and lower eigenvectors, $n' = n \times K$ denotes the sequence length of NLSS feature, $n = w \times w$ denotes the LSS feature sequence length of each sample, w denotes the size of the local space window, and K denotes the number of the samples most similar to x.

For the NLSS features, the sample x itself and its LSS feature x_{LSS} will be located at the front of the nonlocal features. The next place will be the sample most similar to x and its LSS feature. In the architecture of NLSS, the local regions are first sorted according to the similarity between the center pixel and x, and then, the pixels in each local region are sorted like LSS does. So, the importance of a window is higher than that of a pixel.

E. NLSS-RNN

Similar to the LSS model, after extracting NLSS features of the samples, the training samples and the corresponding class labels are used as the inputs to train the related parameters of the RNN model. The whole framework is called NLSS-RNN.

For the NLSS feature x_{NLSS} of sample x, there are n' low-level features, which are already sorted in order. Then, an RNN network with n' time steps is constructed. The NLSS feature x_{NLSS} is fed into the RNN model, and each low-level feature vector in x_{NLSS} corresponds to a time-step input vector in the RNN. For time step t with the corresponding input feature vector, the hidden layer state h^t is calculated by

$$h^t = \sigma_h(W_h x^t_{\text{NLSS}} + U_h h^{t-1} + b_h) \tag{16}$$

where W_h denotes the input-to-hidden matrices, U_h is the matrix of the recurrent weights between the hidden layer and itself at two adjacent time steps, b_h is the biases, and σ_h denotes the activation function.

It should be noted that the hidden layer state of the last time step is also used as a high-level semantic feature. The NLSS is a high-level nonlocal semantic feature, and the time step becomes n', the value of which is K length of the local space sequence n of the superposition; then, the final output of the classification $y^{n'}$ is calculated by

$$y^{n'} = \text{softmax}(W_y h^{n'} + b_y) \tag{17}$$

where $y^{n'}$ is the classification result.

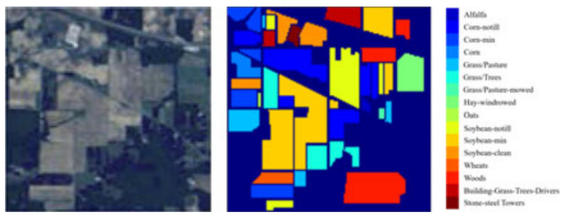


Fig. 5. AVIRIS the Indian Pines dataset false color composite image and corresponding ground-truth areas representing 16 land-cover classes.

IV. Experimental Results

A. Description of Datasets

In our study, three well-known hyperspectral datasets, including Indian Pines and University of Pavia, with different environmental settings are employed to evaluate the proposed method.

The Indian Pines dataset refers to a mixed vegetation site over the Indian Pines test area in Northwestern Indiana. It was gathered by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor. This dataset has 145 $\times$ 145 pixels and 220 spectral bands in the wavelength range of 0.4–2.5 μm. After removing water absorption bands, the number of the bands is reduced to 200. There are 16 different land-cover classes in the ground truth. The false color composite image and the ground-truth maps are shown in Fig. 5. The numbers of the samples of each class are displayed in Table I.

The second dataset University of Pavia was recorded by the reflective optics imaging spectrometer (ROSIS-3) over the University of Pavia, Italy. It has 115 bands with 610 $\times$ 340 pixels. The image has a spatial resolution of 1.3 m per pixel and was collected in the spectral range of 0.43–0.86 μm. In the experiments, 12 noisy channels are removed, and the remaining 103 bands are used for the classification task. The false color composite image and the ground-truth maps of University of Pavia are shown in Fig. 6. The numbers of the samples of each class are displayed in Table II.

TABLE I
LAND-COVER CLASSES AND THE NUMBER OF PIXELS ON THE INDIAN PINES DATASET

Class		No. of Samples	
Code	Name	Train	Test
1	Alfalfa	5	41
2	Corn-notill	143	1285
3	Corn-mintill	83	747
4	Corn	24	213
5	Grass-pasture	48	435
6	Grass-trees	73	657
7	Grass-pasture-mowed	3	25
8	Hay-windrowed	48	430
9	Oats	2	18
10	Soybean-notill	97	875
11	Soybean-mintill	246	2209
12	Soybean-clean	59	534
13	Wheat	20	185
14	Woods	126	1139
15	Buildings-Grass-Tress	39	347
16	Stone-Steal-Towers	9	84
Total		1025	9224

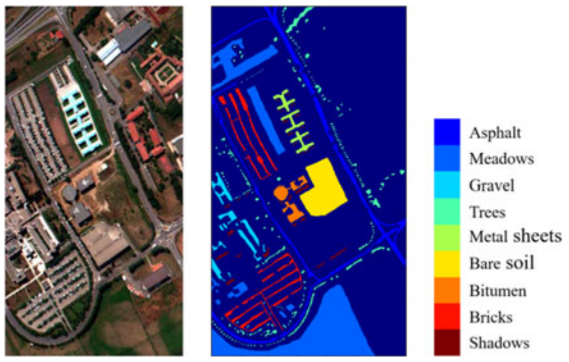


Fig. 6. ROSIS-3: the University of Pavia dataset, false color composite image, and the corresponding ground-truth areas representing nine land-cover classes.

TABLE II
LAND-COVER CLASSES AND THE NUMBER OF PIXELS ON THE UNIVERSITY OF PAVIA DATASET

Class		No. of Samples	
Code	Name	Train	Test
1	Asphalt	597	6034
2	Meadows	1678	16971
3	Gravel	189	1910
4	Trees	276	2788
5	Metal sheets	121	1224
6	Bare soil	453	4576
7	Bitumen	120	1210
8	Bricks	331	3351
9	Shadow	85	862
Total		3850	38926

The final dataset was acquired over Salinas Vally, CA, USA, by the AVIRIS sensor in 1998. The original dataset is composed of 224 bands, with a spectral range of 0.4–2.5 μm. The image spatial resolution is 3.7 m, and it has a size of 512 $\times$ 217 pixels. Similar to other datasets, 20 water absorption bands are removed and 204 bands are left. The false color image and 16 classes of the ground truth are shown in Fig. 7. The numbers of the samples of each class are listed in Table III.

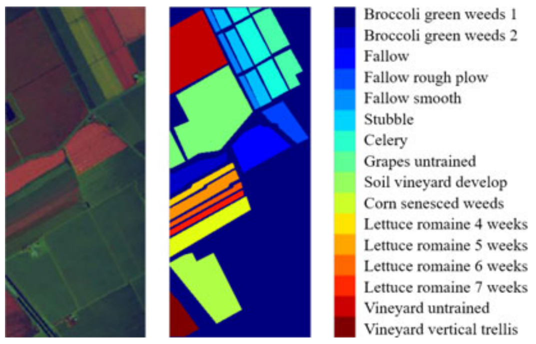


Fig. 7. AVIRIS: the Salinas dataset, false color composite image, and the corresponding ground-truth areas representing 16 land-cover classes.

TABLE III
LAND-COVER CLASSES AND THE NUMBER OF THE PIXELS ON THE SALINAS DATASET

Class		No. of Samples	
Code	Name	Train	Test
1	Broccoli green weeds 1	20	1989
2	Broccoli green weeds 2	37	3689
3	Fallow	20	1956
4	Fallow rough plow	14	1380
5	Fallow smooth	27	2651
6	Stubble	40	3919
7	Celery	36	3543
8	Grapes untrained	113	11158
9	Soil vineyard develop	62	6141
10	Corn senesced weeds	33	3245
11	Lettuce romaine 4 weeks	11	1057
12	Lettuce romaine 5 weeks	19	1908
13	Lettuce romaine 6 weeks	9	907
14	Lettuce romaine 7 weeks	11	1059
15	Vineyard untrained	73	7195
16	Vineyard untrained	18	1789
Total		543	53586

B. *Experimental Design*

In order to investigate the effectiveness of the proposed methods, we compare the results of LSS-RNN and NLSS-RNN for the HSI classification task with several well-known methods and deep learning architectures, including SVM, simultaneous orthogonal matching pursuit (SOMP), SAE, and CNN [59]. It should be noted that the parameters of the SVM are obtained by cross validation, and the parameter setting of these methods, which will be mentioned in the following, come from those reported in the references. On the other hand, taking the Gabor-DMP features into consideration, different performance between individual features and composite features is also demonstrated in our experiments. We also implement specific experiments to explore the effect of the window size on the classification accuracy. For the CNN-based method, we use a large neighborhood window (27 $\times$ 27) for the first PC as the input 2-D image for the three datasets. When the spatial resolution of the image is not very high, 4 $\times$ 4 kernel and 5 $\times$ 5 kernel can be selected to run convolution and the 2 $\times$ 2 kernel is used for pooling. In addition, there are several parameters, which should be selected in the experiments. Considering the small size of these three datasets, three convolution layers and three

TABLE IV
CLASSIFICATION ACCURACY (%) FOR THE INDIAN PINES IMAGE ON THE TEST SET

Class	SVM	SVM (Gabor-DMP)	SOMP	SAE	CNN	LS-RNN	LSS-RNN (Gabor)	LSS-RNN (DMP)	LSS-RNN	NLSS-RNN
1	63.42	95.10	89.80	90.24	**100**	90.24	82.93	95.12	95.12	95.12
2	81.17	95.30	89.78	91.75	96.34	98.13	91.52	97.12	97.51	**98.21**
3	70.55	98.80	90.41	93.57	99.49	97.72	93.98	97.46	97.05	**99.60**
4	53.52	93.40	87.68	90.14	**100**	96.24	92.49	97.18	94.37	97.18
5	91.72	98.62	94.87	93.79	**99.91**	94.71	97.01	94.48	98.16	97.93
6	93.91	98.50	99.11	94.82	**99.75**	96.19	98.48	97.87	97.87	98.33
7	76.00	92.00	41.67	84.00	**100**	84.00	68.00	84.00	96.00	96.00
8	94.65	100	99.77	100	10	99.53	98.60	100	100	**100**
9	22.22	100	0	61.11	100	77.78	72.22	100	100	94.44
10	69.03	92.34	86.70	89.49	**98.72**	94.74	87.66	95.20	96.46	97.60
11	83.48	96.56	95.72	94.30	95.52	98.55	95.88	98.64	**99.59**	99.28
12	71.16	96.63	89.33	93.45	**99.47**	91.95	92.13	95.51	98.50	98.31
13	97.84	99.46	98.95	99.46	100	86.49	97.30	98.92	98.92	**100**
14	95.00	99.39	99.40	99.30	99.55	**99.91**	99.56	99.21	99.47	99.21
15	43.52	98.56	92.40	88.18	**99.54**	97.12	89.34	98.27	98.85	98.85
16	92.86	97.62	95.35	92.86	99.34	100	100	98.81	100	**100**
OA	81.05	96.96	93.51	93.93	97.56	97.09	94.50	97.61	98.36	**98.75**
AA	75.00	97.02	84.43	91.03	**99.23**	93.96	91.07	96.74	97.99	98.13
Kappa	0.78	0.97	0.93	0.93	0.97	0.97	0.94	0.97	0.98	**0.99**

pooling layers are used here. In the training procedure, we use the minibatch-based backpropagation method, where the size of the minibatch is set as 100. The learning rate of all the CNNs is set to be 0.01, and the number of the training epochs of the CNN is 200.

For evaluating the performance of the proposed methods efficiently, for each dataset, the labeled samples are separated into a training set and a test set randomly. The numbers of the sample selection process is shown in Tables I and II. To be specific, we run the experiments more than ten times with different initial training samples and record the average results of the measures, including overall accuracy (OA), average accuracy (AA), Kappa coefficient, and accuracy of each class.

C. *AVIRIS Indian Pines*

For the Indian Pines dataset, we randomly select 10% examples per class as the training set and the remaining 90% samples as the test set for all the methods. The numbers of the training and test samples are shown in Table I. First of all, to produce the Gabor texture features, the standard PCA process is implemented on the Indian Pines dataset, and the first ten PCs are selected. In our work, four orientations and three scales are considered to construct the Gabor filters. For each PC, the length of the texture feature vector is 12, and for the whole image, we have a texture feature matrix $X_{\text{texture}} \in R^{145\times145\times120}$. For the DMP morphological features, we only use the first five PCs, and the sizes of the structure elements are set to 2, 4, 6, 8, and 10. Then, we have the morphological feature matrix $X_{\text{shape}} \in R^{145\times145\times50}$. For the proposed LSS-RNN method, the window size is 7×7, and we have used 170 units (the same as the length of the input feature vector) in the hidden layer. The number of the iteration is set to 1000, which is large enough for our model to converge. For the proposed NLSS-RNN method, the window size is also set to 7×7, and we have 170 units in the hidden layer. The number of the iteration is set to 1000, and the number of nonlocal neighbors (K) is set to 2; the other model parameters set consistent with LSS-RNN. Thirty-two 4×4 convolution kernels and one 2×2 pooling kernel are used in the CNN for this dataset. The quantitative results averaged over ten runs for various methods are tabulated in Table IV.

In our experiments, for SVM and SVM (Gabor-DMP), we use a one-against-one strategy with RBF kernels, and the parameters are obtained by cross validation. The LS-RNN model only uses the local spatial (neighborhood) pixels without sorting them by their importance. The window sizes of SOMP and LS-RNN are also set to 7×7, the same as that of the proposed LSS-RNN method. In the SAE, we have used 90 units in the first hidden layer and 50 units in the second hidden layer. A traditional softmax layer is connected to complete the classification task. In Table IV, for SVM (Gabor-DMP), SAE, LS-RNN, LSS-RNN, and NLSS-RNN, the composite Gabor-DMP features are used.

As shown in Table IV, for the SVM model, compared with the original spectral features, the composite Gabor-DMP features clearly lead to a better performance. Regarding the integration of LSS-RNN with different features, the composite Gabor-DMP features are more discriminative than individual Gabor or DMP morphological features. On the other hand, different features have different discrimination capabilities for HSIs, for example, the DMP morphological features perform much better than the Gabor texture features on classes 1, 7, 10, and 15. It is clear that local spatial information plays an important role in the classification task. For LS-RNN and LSS-RNN, because of the sorting by importance, the local spatial sequence in the LSS-RNN model contains the center pixel and its neighborhoods, reducing the influence of other nonsupportive pixels. In Table IV, we can see that the OA of LSS-RNN have increased about 1.3% than that of the LS-RNN. Local spatial sequences not only come up with local spatial information, but also allow us to exploit the relationship of pixels in a neighboring area, which is ignored by SOMP. Thanks to LSS information, the best result is obtained by LSS-RNN: OA $=$ 98.36%, AA $=$ 97.99, and Kappa $=$ 0.9813. We can see that the CNN gets a better result than SVM, SVM (Gabor-DMP), SAE, LS-RNN, LSS-RNN (Gabor), and LSS-RNN (DMP) on the Indian Pines dataset. Our proposed LSS-RNN (Gabor-DMP) method gets better results in terms of

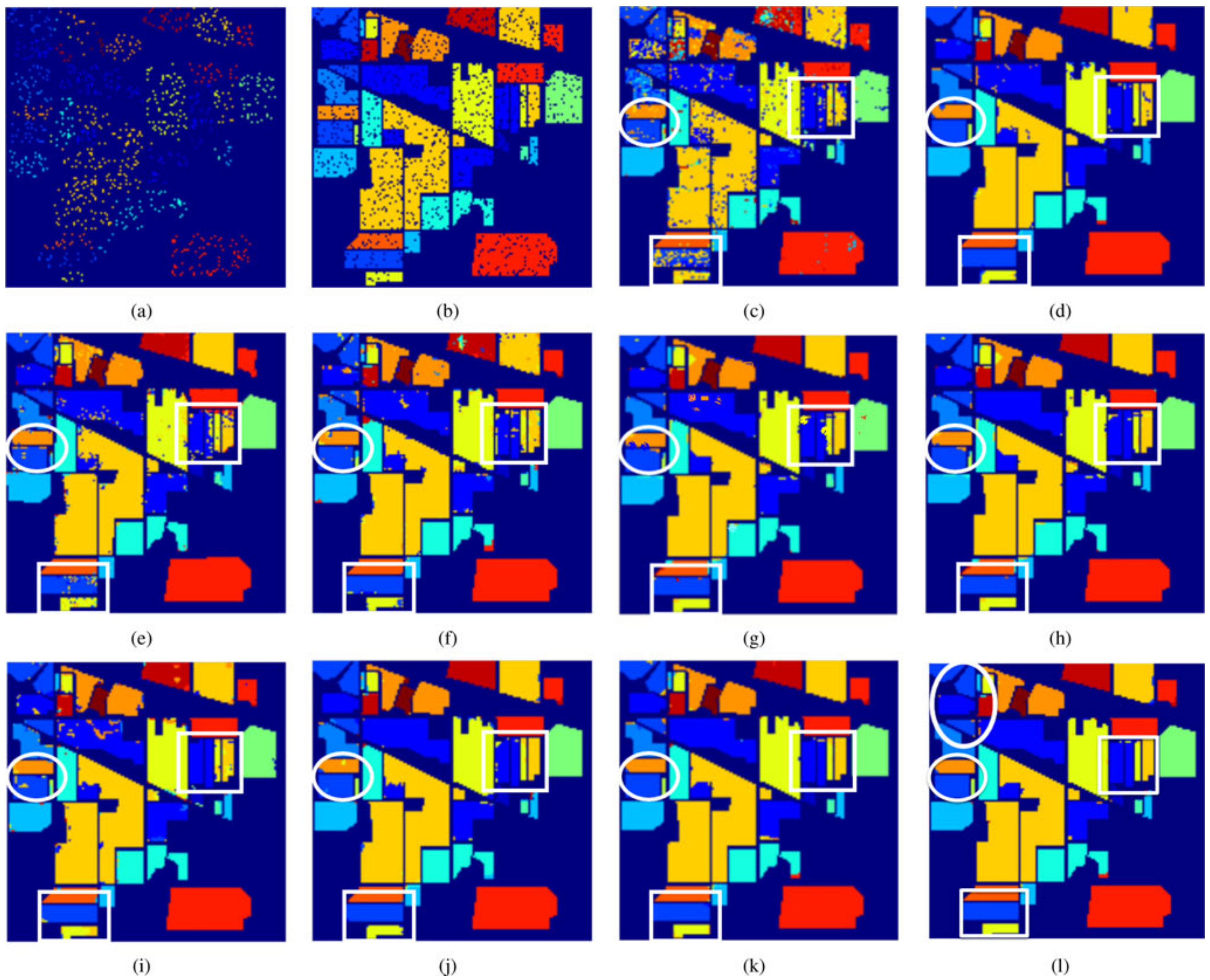


Fig. 8. For the Indian Pines image. (a) Training samples. (b) Test samples. (c) SVM. (d) SVM (Gabor-DMP). (e) SOMP. (f) SAE. (g) CNN. (h) LS-RNN. (i) LSS-RNN (Gabor). (j) LSS-RNN (DMP). (k) LSS-RNN. (l) NLSS-RNN.

OA and Kappa. Since NLSS-RNN maintains more information than LSS-RNN by integrating local and nonlocal sequence features of the samples, it obtains better results in terms of OA and AA.

The classification results obtained from the various methods with the fixed training samples are shown in Fig. 8(c)–(l). It is witnessed that because of the high discriminability of the composite Gabor-DMP feature, SVM (Gabor-DMP) can obtain more accurate results than SVM. As shown in the regions marked by the white ellipses, compared with the other individual low-level features, the composite Gabor-DMP low-level features help the methods achieve better results in some areas. As we can see from Fig. 8, taking local spatial information into consideration is beneficial for the HSI classification task, especially for reserving local information (marked by the white rectangles). Looking closely at Fig. 8(g) and (j), LSS-RNN did achieve a better result than the LS-RNN and leads to more clear boundaries. Moreover, compared with LSS-RNN, NLSS-RNN can get more accurate classification results in some details, such as the area marked by the white ellipses. In summary, the proposed LSS-RNN and NLSS-RNN (Gabor-DMP) methods achieve comparable performance against the other traditional methods.

D. ROSIS-3 University of Pavia

In this experiment, for each of the nine classes, 9% of the labeled pixels are randomly selected for training, while the rest 91% are used to test the classifiers (see Table II). The Gabor and DMP feature are extracted using the same process implemented on the Indian Pines dataset with the same parameters. Fig. 9 illustrates the classification results of the different methods on the University of Pavia dataset.

Considering both the computing complexity of the training model and classification accuracy, we set the local window size of SOMP, LS-RNN, LSS-RNN, and NLSS-RNN to 7×7 ($w = 7$). For NLSS-RNN, the number of the iteration is set to 1000, and the number of nonlocal neighbors (K) is set to 3. For SAE, the numbers of the first and second hidden layers are set to 90 and 50, respectively. For CNN, 64 5×5 convolution kernels and one 2×2 pooling kernel are used.

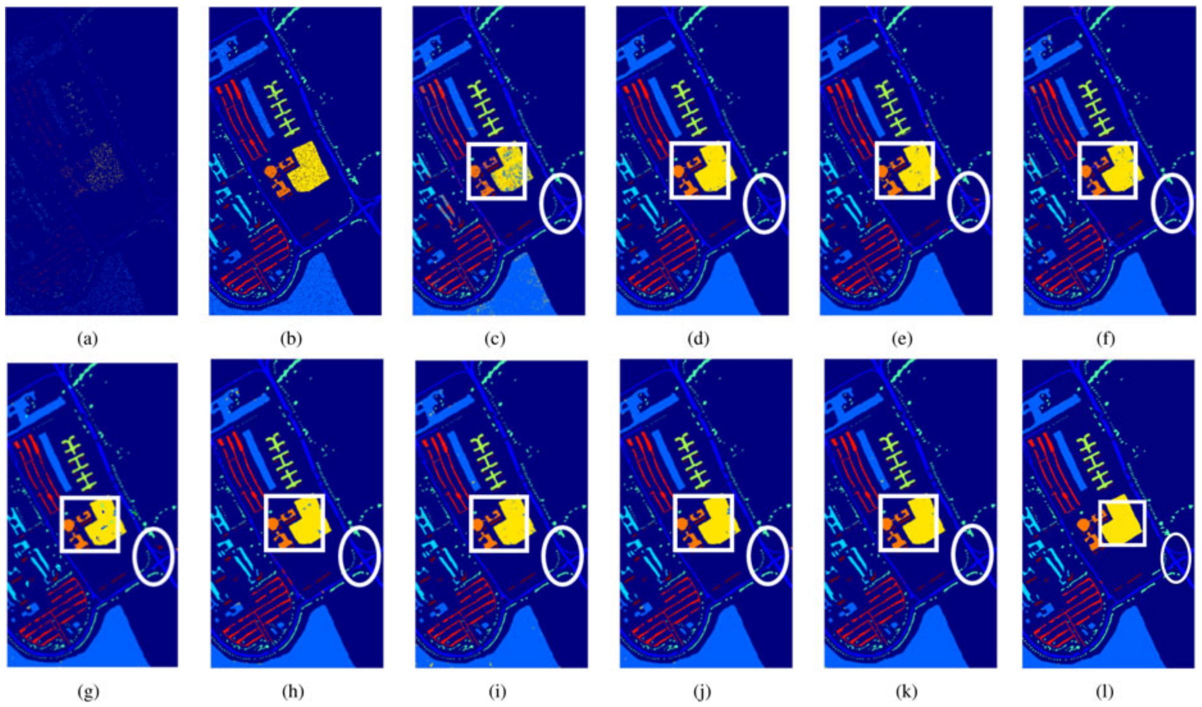


Fig. 9. For the Pavia image. (a) Training samples. (b) Test samples. (c) SVM. (d) SVM (Gabor-DMP). (e) SOMP. (f) SAE. (g) CNN. (h) LS-RNN. (i) LSS-RNN (Gabor). (j) LSS-RNN (DMP). (k) LSS-RNN. (l) NLSS-RNN.

TABLE V
Classification Accuracy (%) for The University of Pavia Image on the Test Set

Class	SVM	SVM (Gabor-DMP)	SOMP	SAE	CNN	LS-RNN	LSS-RNN (Gabor)	LSS-RNN (DMP)	LSS-RNN	NLSS-RNN
1	92.56	98.99	94.61	96.06	99.36	96.85	99.24	99.57	99.80	**99.67**
2	97.00	99.59	**99.91**	99.06	99.36	99.32	98.92	99.70	99.75	99.90
3	71.20	97.17	96.60	87.59	99.69	99.69	94.55	98.32	99.06	**99.69**
4	93.40	99.39	91.04	98.31	99.63	97.49	98.17	**99.64**	99.57	99.21
5	99.92	99.92	99.75	99.84	99.95	99.59	100	99.51	100	**100**
6	77.05	97.66	97.97	96.37	**99.96**	98.01	98.08	97.36	99.43	99.85
7	79.26	97.02	97.19	88.76	**100**	98.35	95.12	96.36	99.50	99.34
8	84.09	99.22	96.81	96.06	99.65	99.07	97.91	97.94	99.22	**99.67**
9	98.26	99.88	86.08	98.61	99.38	98.38	98.61	99.42	99.65	**100**
OA	90.90	99.04	97.40	97.10	99.54	98.61	98.42	99.06	99.63	**99.77**
AA	88.83	98.76	95.55	95.63	99.66	98.53	97.84	98.65	99.55	**99.70**
Kappa	0.88	0.99	0.97	0.96	0.994	0.98	0.98	0.98	0.99	**0.997**

From Table V, we have the same occlusion as that shown in the last section. The methods containing local spatial features of the HSIs have better classification accuracy than those classifiers based on single pixels' information. Composite Gabor-DMP features are much more discriminative than individual Gabor of DMP features. By exploiting the relationship of local pixels, LSS-RNN is more accurate than LS-RNN. Take a look at classes 6 and 7, the LSS-RNN model achieves more than 99% accuracy, much higher than the other methods. For OA, AA, and Kappa, LSS-RNN also achieves the best performance among all the methods. At the same time, the validity of the NLSS-RNN algorithm is also proved. The highest classification accuracy is obtained for the third, fifth, eighth, and ninth classes, and OA, AA, and Kappa are also the highest among the comparison algorithms. The Kappa coefficient is even close to 1.

Fig. 9(c)–(l) shows the classification of the methods on the fixed training samples. It is observed that the classification results of those pixelwise methods are very poor. Using local area information, SOMP, LS-RNN, and LSS-RNN can achieve cleaner results (marked by the white rectangle). The proposed LSS-RNN algorithm not only reduces the noise, but also keeps clear boundaries and obtains better classification results in many small regions (marked by the white ellipse). With the powerful Gabor-DMP features, the Gabor features seem to be more influential than the DMP features. The classification of NLSS-RNN is more accurate and smooth in flat areas. Overall, the proposed NLSS-RNN method outperforms the other methods.

E. AVIRIS Salinas

For this dataset, as shown in Table III, we randomly select about 1% samples from each class to form the training set and the rest samples for testing. The classification results of the

TABLE VI
CLASSIFICATION ACCURACY (%) FOR THE SALINAS IMAGE ON THE TEST SET

Class	SVM	SVM (Gabor-DMP)	SOMP	SAE	CNN	LS-RNN	LSS-RNN (Gabor)	LSS-RNN (DMP)	LSS-RNN	NLSS-RNN
1	98.79	100	100	98.04	98.36	100	**100**	99.04	98.49	92.26
2	99.13	99.76	99.86	99.38	98.43	100	98.92	**100**	99.97	99.76
3	98.67	**99.80**	94.43	98.52	92.97	99.59	97.24	99.23	99.74	99.49
4	99.57	99.86	93.12	99.35	99.46	97.97	97.90	99.49	**99.86**	99.78
5	94.53	97.02	88.57	96.08	91.83	**98.45**	97.59	97.96	97.06	97.70
6	99.52	99.90	**100**	99.44	99.83	99.95	99.82	98.88	99.85	99.77
7	99.41	98.65	99.60	97.46	99.68	98.62	99.29	99.58	99.41	**99.75**
8	84.63	91.69	94.70	88.59	68.94	91.50	89.41	90.55	93.17	**94.76**
9	98.78	99.19	99.74	99.67	98.45	99.46	99.17	99.28	99.64	**99.92**
10	91.34	97.78	95.41	92.39	73.31	**98.74**	92.73	95.84	97.44	96.95
11	96.22	99.53	93.38	97.07	90.85	99.43	94.89	97.16	96.12	**99.91**
12	99.58	100	91.61	97.85	98.31	96.12	99.74	82.60	**100**	99.90
13	99.23	99.67	67.48	98.90	97.43	100	99.01	99.78	**100**	98.79
14	80.45	96.22	92.83	98.30	97.76	**99.34**	87.54	98.87	96.79	98.30
15	64.41	78.33	72.86	81.46	63.75	90.72	89.09	84.36	**94.66**	93.01
16	95.86	94.52	99.05	91.62	85.24	99.44	90.50	90.50	95.42	**99.89**
OA	90.22	94.59	92.81	93.58	85.24	96.43	94.69	94.58	97.11	**97.23**
AA	93.76	96.99	92.67	95.88	85.22	98.08	95.80	96.36	97.98	**98.12**
Kappa	0.89	0.94	0.92	0.93	0.85	0.96	0.94	0.94	0.97	**0.97**

different methods have been shown in Table VI, and the best results are marked in bold.

Due to the homogeneity of this image, the size of the local spatial window is larger than those used on the Indian Pines and University of Pavia datasets. For the Salinas dataset, the window size is set to 9×9 for LS-RNN, LSS-RNN (Gabor), LSS-RNN (Gabor), LSS-RNN, and NLSS-RNN. For NLSS-RNN, the number of nonlocal neighbors (K) is set to 2. The window size used in SOMP is set to 15×15. It should be noted that this window size such as 15×15 requires significant computational complexity. Other parameters are the same as those mentioned above. One hundred and twenty-eight 4×4 convolution kernels and one 2×2 pooling kernel are used for this dataset.

From Table VI, we can see that the CNN-based method cannot obtain a good performance in this dataset, and all the other methods have promising classification results. Our method achieves the best results. Especially, for class 15, which contains more than 7000 pixels, compared with the other methods, LSS-RNN is of 94.66%, which is much higher than the others. For most of the classes, LSS-RNN achieves comparative high classification accuracy against the others. LSS-RNN also achieves the highest OA and AA, 97.11% and 97.98%, respectively. The OA, AA, and Kappa coefficients of NLSS-RNN have been improved over LSS-RNN and achieve the best results among the comparison algorithms.

Fig. 10(a) and (b) shows the training and test samples, and the classification results of the different methods have been shown in Fig. 10(c)–(l). Fig. 10 shows that our methods sustain more accurate and smoother classification results. It is clear that the methods using local spatial information result in much clean outlines of the regions (marked by the white rectangles). The white ellipse region shows the ability of high-level contextual features to obtain more accurate detail information. Also, the composite Gabor-DMP low-level features perform better than the individual Gabor or DMP features. For LS-RNN and LSS-RNN, comparing Fig. 10(g) and (j), their classification results look very similar, and they are also only about 1% different in OA, as shown in Table VI. This may be due to the fact that because of the large homogeneity of the Salinas image, the sorting procedure does not help much.

F. Parameter Analysis

In this section, we focus on the parameters influencing classification results directly, including the size of the training set and the local spatial window size. We intend to understand how these parameters affect the different methods' performance on various datasets. We aim to find a general pattern of parameters' influence to our proposed LSS-RNN model. The experiments are conducted five times, and the average values of the experiment results are recorded to avoid the bias induced by random sampling.

1) Effect of the Number of Training Samples: We implement a few experiments to investigate the effect of the number of the training samples. For each method, the number of the selected training samples is the only parameter to be changed, while the others are fixed. 1%, 2%, 5%, 10%, 15%, 20%, 25%, and 30% of the labeled samples from each class are selected randomly as the training sets for the analysis. The OA results of the different methods on various sizes of the training sets are displayed in Fig. 11(a) for Indian Pines and in Fig. 11(b) for University of Pavia, where the x-axis represents the percentage of the training samples selected per class and the y-axis denotes the OA.

From Fig. 11, we can see that for every method, as the number of the training samples increases, the OA also increases. The LSS-RNN (Gabor-DMP) outperforms the other methods, especially when the training sample set is 2–10% for both the two datasets. When the number of the training samples is large enough, the OA of each method tends to be stable.

2) Effect of the Local Spatial Window Size: To investigate the local spatial window size attributing to the system performance, more experiments are performed to explore the effect of window sizes. The window size is set to 3×3, 5×5, 7×7, 9×9, and 11×11 individually. The OAs of LSS-RNN with different features are shown in Fig. 12. It is evident that at the

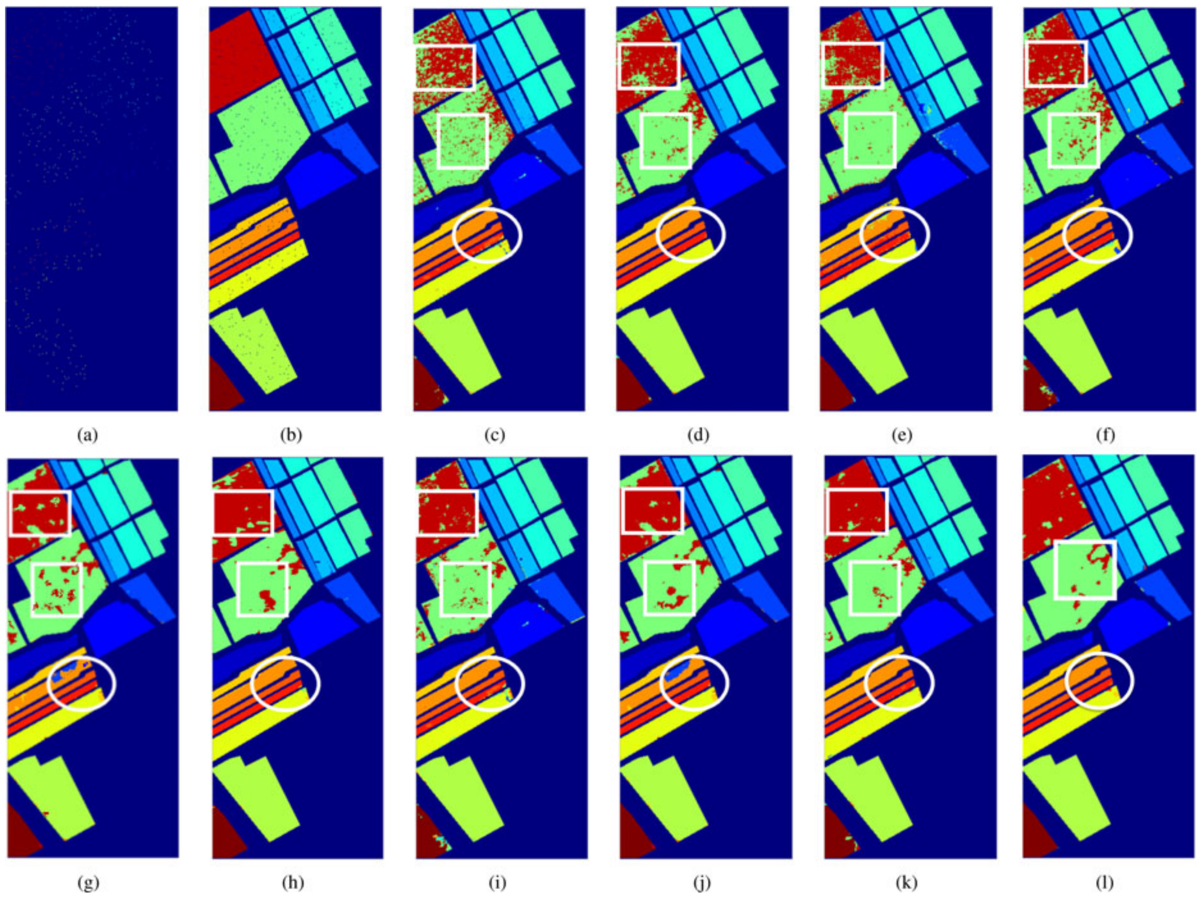


Fig. 10. For the Salinas image. (a) Training samples. (b) Test samples. (c) SVM. (d) SVM (Gabor-DMP). (e) SOMP. (f) SAE. (g) CNN. (h) LS-RNN. (i) LSS-RNN (Gabor). (j) LSS-RNN (DMP). (k) LSS-RNN. (l) NLSS-RNN.

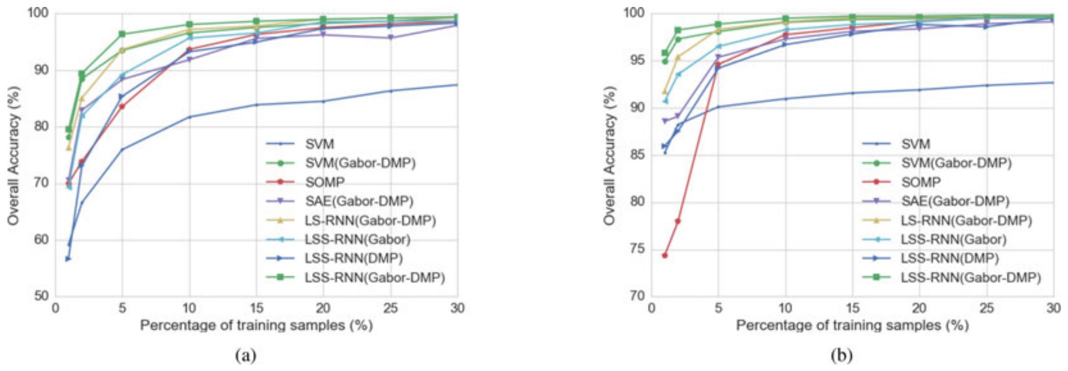


Fig. 11. Effects of the number of training samples. (a) For Indian Pines. (b) For University of Pavia.

beginning, when the window size is between 3 and 9, as the window size becomes bigger, more spatial information is included and the OA increases correspondingly. If the window size is large enough, the space information is more important than the Gabor-DMP features; the classification accuracy will decrease. It is crucial to choose an appropriate window size for combining both the feature vectors and LSS information together.

3) Effect of the Number of Nonlocal Neighbors and Spatial Window Size in NLSS-RNN: In NLSS-RNN, we adopt KNN to search the nonlocal neighbors for a given pixel. To investigate the effect of the number of nonlocal neighbors K and the local window size w onto the NLSS-RNN classification performance, we implement the experiments with different values of K and w. The number of nonlocal neighbors is set to 1, 2, 3, 4, and 5, and the window size is set to 3×3, 5×5, and 7×7, respectively. The OA curves of NLSS-RNN with different parameter values are shown in Fig. 13.

From Fig. 13, we can see that NLSS-RNN gets better results with the local window size $w = 7$ for both the Indian Pines and Pavia University datasets. When the number of nonlocal neighbors K is set to 2 or 3, NLSS-RNN can achieve the best result. Large values of K not only increase the computational cost, but

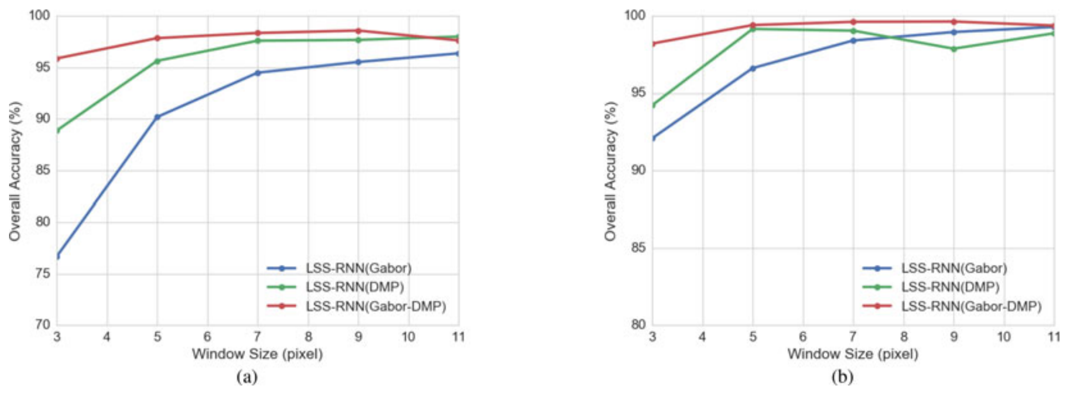


Fig. 12. Effect of different window sizes. (a) For Indian Pines. (b) For University of Pavia.

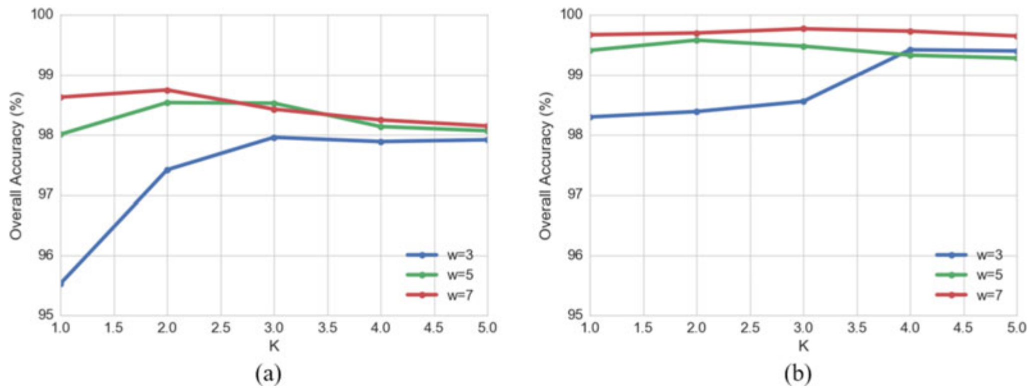


Fig. 13. Effects of the number of nonlocal neighbors and different window sizes on NLSS-RNN. (a) For Indian Pines. (b) For University of Pavia.

also bring more irrelevant pixels to NLSS feature learning, and thus, will lower the OA value.

V. Conclusion

In this paper, we have proposed a novel LSS-RNN deep learning framework for HSI classification, which constructs a powerful composite low-level feature and not only extracts the local spatial information of an HSI, but also exploits the relationship between local pixels to strengthen the positive effects of "good" pixels and reduce the negative impacts of "bad" pixels. First, different types of low-level features are extracted to construct more powerful composite low-level features to represent the complete information of the HSI. Second, in order to efficiently exploit the relationship between local pixels, we construct the LSS features from each sample based on the composite low-level feature matrix. Then, the LSS features and the corresponding standard labels of the training samples are fed into the standard RNN to optimize the parameters using BPTT. Finally, the well-trained LSS-RNN model is used to classify test samples. Besides, the nonlocal information is explored by constructing local and nonlocal spatial sequence features and NLSS-RNN. This method introduces a nonlocal thought, which not only retains the excellent properties of LSS features, but also increases the amount of information contained in features, and improves the classification accuracy. The experimental results on the three datasets have proved that our methods outperform the other state-of-the-art methods. Our further work will mainly focus on improving the RNN framework by adapting different layer architectures to the applications.

References

[1] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, no. 1, pp. S110–S122, 2009.
[2] J. A. Richards, *Remote Sensing Digital Image Analysis: An Introduction*. Berlin, Germany: Springer, 2006.
[3] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.
[4] J. A. Gualtieri and R. F. Cromp, "Support vector machines for hyperspectral remote sensing classification," *Proc. SPIE*, vol. 3584, no. 25, pp. 1–28, 1999.
[5] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3973–3985, Oct. 2011.
[6] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification via kernel sparse representation," in *Proc. IEEE Int. Conf. Image Process.*, Brussels, Belgium, Sep., 2011, pp. 1233–1236.
[7] R. Hang, Q. Liu, H. Song, and Y. Sun, "Matrix-based discriminant subspace ensemble for hyperspectral image spatial–spectral feature fusion," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 2, pp. 783–794, Feb. 2016.
[8] R. Hang *et al.*, "Robust matrix discriminative analysis for feature extraction from hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 5, pp. 2002–2011, May 2017.
[9] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. Part i: Greedy pursuit," *Signal Process.*, vol. 86, no. 3, pp. 572–588, 2006.
[10] J. Li, H. Zhang, Y. Huang, and L. Zhang, "Hyperspectral image classification by nonlocal joint collaborative representation with a locally adaptive dictionary," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 6, pp. 3707–3719, Jun. 2014.

[11] Z. Lei, Y. Meng, X. Feng, M. Yi, and D. Zhang, “Collaborative representation based classification for face recognition,” *Proc. IEEE Conf. Comput. Vis.*, 2012, pp. 471–478.

[12] S. Wang, “Relaxed collaborative representation for pattern classification,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2224–2231.

[13] G. Camps-Valls, T. Bandos Marsheva, and D. Zhou, “Semi-supervised graph-based hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 10, pp. 3044–3054, Oct. 2007.

[14] W. Liao, R. Bellens, A. Pizurica, and W. Philips, “Classification of hyperspectral data over urban areas using directional morphological profiles and semi-supervised feature extraction,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 4, pp. 1177–1190, Aug. 2012.

[15] L. Shen and S. Jia, “Three-dimensional Gabor wavelets for pixel-based hyperspectral imagery classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 12, pp. 5039–5046, Dec. 2011.

[16] J. Li, H. Zhang, L. Zhang, X. Huang, and L. Zhang, “Joint collaborative representation with multitask learning for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 9, pp. 5923–5936, Sep. 2014.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

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

[19] D. Yu, L. Deng, and S. Wang, “Learning in the deep-structured conditional random fields,” in *Proc. NIPS Workshop Deep Learn. Speech Recognit. Related Appl.*, 2009, pp. 1848–1852.

[20] A. R. Mohamed, T. N. Sainath, G. E. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny, “Deep belief networks using discriminative features for phone recognition,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2011, pp. 5060–5063.

[21] 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.

[22] H. Lyu *et al.*, “Long-term annual mapping of four cities on different continents by applying a deep information learning method to landsat data,” *Remote Sens.*, vol. 10, no. 3, 2018, Art. no. 471.

[23] L. Mou and X. X. Zhu, “Learning spectral-spatial-temporal features via a recurrent convolutional neural network for change detection in multispectral imagery,” arXiv:1803.02642, 2018.

[24] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[25] 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.

[26] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, 2010.

[27] G. E. Hinton, “Deep belief networks,” *Scholarpedia*, vol. 4, no. 6, p. 5947, 2009.

[28] R. Salakhutdinov and G. Hinton, “A better way to pretrain deep Boltzmann machines,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 2447–2455.

[29] Y. Lecun *et al.*, “Backpropagation applied to handwritten zip code recognition,” *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[30] C. Goller and A. Kuchler, “Learning task-dependent distributed representations by backpropagation through structure,” in *Proc. IEEE Int. Conf. Neural Netw.*, 1996, vol. 1, pp. 347–352.

[31] J. L. Elman, “Finding structure in time,” *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[32] B. Shuai, Z. Zuo, G. Wang, and B. Wang, “Dag-recurrent neural networks for scene labeling,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3620–3629.

[33] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki, “Scene labeling with LSTM recurrent neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3547–3555.

[35] 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.

[36] 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.

[37] T. Li, J. Zhang, and Y. Zhang, “Classification of hyperspectral image based on deep belief networks,” in *Proc. IEEE Int. Conf. Image Process.*, 2015, pp. 5132–5136.

[38] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, “Deep convolutional neural networks for hyperspectral image classification,” *J. Sens.*, vol. 2015, no. 2, pp. 1–12, 2015.

[39] L. Mou, P. Ghamisi, and X. X. Zhu, “Deep recurrent neural networks for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, Jul. 2017.

[40] M. Rubwurm and M. Korner, “Temporal vegetation modelling using long short-term memory networks for crop identification from medium-resolution multi-spectral satellite images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1496–1504.

[41] H. Lyu, H. Lu, and L. Mou, “Learning a transferable change rule from a recurrent neural network for land cover change detection,” *Remote Sens.*, vol. 8, no. 6, 2016, Art. no. 506.

[42] Q. Liu, F. Zhou, R. Hang, and X. Yuan, “Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification,” *Remote Sens.*, vol. 9, 2017, Art. no. 1330.

[43] D. Svozil, V. Kvasnicka, and J. Pospichal, “Introduction to multi-layer feed-forward neural networks,” *Chemometr. Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, 1997.

[44] D. Rumelhart and J. Mcclelland, *Learning Internal Representations by Error Propagation*. San Diego, CA, USA: Univ. California, 1986.

[45] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” arXiv: 1506.00019, 2015.

[46] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Germany: Springer, 2012.

[47] Y. Chen, L. Zhu, P. Ghamisi, X. Jia, G. Li, and L. Tang, “Hyperspectral images classification with Gabor filtering and convolutional neural network,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 12, pp. 2355–2359, Dec. 2017.

[48] M. Shi and G. Healey, “Hyperspectral texture recognition using a multiscale opponent representation,” *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 5, pp. 1090–1095, May 2003.

[49] T. C. Bau, S. Sarkar, and G. Healey, “Hyperspectral region classification using a three-dimensional Gabor filterbank,” *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 9, pp. 3457–3464, Sep. 2010.

[50] X. Huang and L. Zhang, “An SVM ensemble approach combining spectral, structural, and semantic features for the classification of high-resolution remotely sensed imagery,” *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 257–272, Jan. 2013.

[51] J. A. Benediktsson, M. Pesaresi, and K. Amason, “Classification and feature extraction for remote sensing images from urban areas based on morphological transformations,” *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 9, pp. 1940–1949, Sep. 2003.

[52] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, “Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles,” *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3804–3814, Nov. 2008.

[53] J. A. Benediktsson, M. Pesaresi, and K. Amason, “Classification and feature extraction for remote sensing images from urban areas based on morphological transformations,” *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 9, pp. 1940–1949, Sep. 2003.

[54] L. Shen and L. Bai, “Mutualboost learning for selecting Gabor features for face recognition,” *Pattern Recognit. Lett.*, vol. 27, no. 15, pp. 1758–1767, 2006.

[55] L. Zhang, L. Zhang, D. Tao, and X. Huang, “On combining multiple features for hyperspectral remote sensing image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 3, pp. 879–893, Mar. 2012.

[56] J. Li, H. Zhang, and L. Zhang, “Efficient superpixel-level multitask joint sparse representation for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 10, pp. 1–14, Oct. 2015.

[57] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, 2015, pp. 807–814.

[58] Q. V. Le, N. Jaitly, and G. E. Hinton, “A simple way to initialize recurrent networks of rectified linear units,” arXiv: 1504.00941, 2015.

[59] 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.

Xiangrong Zhang (SM'14) received the B.S. and M.S. degrees in computer science in 1999 and 2003, respectively, and the Ph.D. degree in electronic engineering in 2006, all from Xidian University, Xi'an, China.

She is currently a Professor with the Key Laboratory of Intelligent Perception and Image Understanding of the Ministry of Education, Xidian University. She was a Visiting Scientist with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, between February 2015 and March 2016. Her research interests include pattern recognition, machine learning, and remote sensing image analysis and understanding.

Yujia Sun received the B.Eng. degree in electronic engineering in 2017 from Xidian University, Xi'an, China, where she is currently working toward the master's degree at the School of Artificial Intelligence.

Her research interests include machine learning and hyperspectral image processing.

Kai Jiang received the B.Eng. degree in electronic engineering in 2014 from Xidian University, Xi'an, China, where he is currently working toward the master's degree.

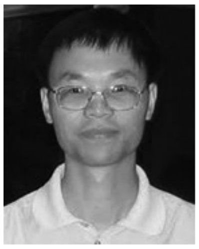
His research interests include machine learning and hyperspectral image processing.

Chen Li received the Ph.D. degree from the University of Cambridge, Cambridge, U.K.

He has been with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. He currently serves as a Professor with the Computer Science Department, Xi'an Jiaotong University, Xi'an, China. His research interests include data mining for studying semantic relations.

Licheng Jiao (SM'89–F'17) received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 1982, and the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1990, respectively.

He has authored or co-authored over 150 scientific papers. His research interests include signal and image processing, nonlinear circuit and systems theory, learning theory and algorithms, optimization problems, wavelet theory, and data mining.



Huiyu Zhou received the B.Eng. degree in radio technology from the Huazhong University of Science and Technology, Wuhan, China, the M.S. degree in biomedical engineering from the University of Dundee, Dundee, U.K., and the Ph.D. degree in computer vision from Heriot-Watt University, Edinburgh, U.K.

He is currently a Reader with the Department of Informatics, University of Leicester, Leicester, U.K. His research interests include medical image processing, computer vision, intelligent systems, and data mining.