Multilevel Superpixel Structured Graph U-Nets for Hyperspectral Image Classification

Qichao Liu, *Student Member, IEEE*, Liang Xiao, *Member, IEEE*, Jingxiang Yang, *Member, IEEE*, and Zhihui Wei

***Abstract*— Limited by the shape-fixed kernels, convolutional neural networks (CNNs) are usually difficult to model difform land covers in hyperspectral images (HSIs), leading to inadequate land use. Recently, benefiting from the ability to conduct shape-adaptive convolutions and model complex patterns in graph-structured data, graph convolutional networks (GCNs) have been applied to HSI classification. However, due to the massive computation in GCNs, HSI is usually pretreated into a graph based on a specific superpixel segmentation, which limits the modeling of spatial topologies to the same scale. To break this limitation, we propose a multilevel superpixel structured graph U-Net (MSSGU) to learn multiscale features on multilevel graphs. Specifically, we construct several hierarchical segmentations from fine to coarse by progressively merging adjacent superpixels and then convert them into multilevel graphs. Meanwhile, based on the merging relations between hierarchical superpixels, we establish the pooling and unpooling functions to transfer features from one graph to another, thereby enabling different-level graphs to collaborate in a single network. Different from concatenating different-scale features straightforwardly in the feature fusion stage, MSSGU fuses them in a coarse-to-fine progressive manner, which can generate subtler fusion features adaptive to the pixelwise classification task. Moreover, we use a CNN instead of GCN to extract and fuse the pixel-level features, which greatly reduces the computation. Such a hybrid U-Net can exploit features of HSIs from a multiscale hierarchical perspective, and its performance has been proven competitive with other deep-learning-based methods by extensive experiments on three benchmark datasets.**

***Index Terms*— Convolutional neural network (CNN), graph convolutional network (GCN), graph U-Net, hierarchical superpixel segmentation, hyperspectral image (HSI) classification.**

Manuscript received July 15, 2021; revised August 21, 2021; accepted September 6, 2021. Date of publication September 24, 2021; date of current version January 26, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61871226, Grant 61571230, and Grant 62001226; in part by the Jiangsu Provincial Social Developing Project under Grant BE2018727; and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20200465. *(Corresponding author: Liang Xiao.)*

Qichao Liu, Jingxiang Yang, and Zhihui Wei are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: qc.l@njust.edu.cn; yang123jx@njust.edu.cn; gswei@njust.edu.cn).

Liang Xiao is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China, and also with the Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: xiaoliang@mail.njust.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TGRS.2021.3112586, provided by the authors.

Digital Object Identifier 10.1109/TGRS.2021.3112586

I. Introduction

HYPERSPECTRAL imagers can capture hundreds of imagery bands in the wavelength range of 0.3–2.5μm. Benefiting from the abundant spectral–spatial information, hyperspectral images (HSIs) can help analyze the types, components, and distributions of materials at the pixel level. Up until now, hyperspectral imaging technology has been widely applied to many fields, such as geological exploration, environmental monitoring, and medical diagnosis [1]–[3]. As the basis for these tasks, HSI classification plays a deterministic role.

In early studies, most HSI classification methods modeled spectra based on statistical theories. Typical machine learning models, such as linear or nonlinear regression [4], support vector machine (SVM) [5], and decision tree [6], were used to classify spectra. Due to the high dimensionality of spectra, such statistic-based methods are usually insufficient in generalization, which limits their application scope. To overcome this drawback, various representation-based methods have been exploited to reveal the inherent structures in spectral data, such as sparse representation [7]–[9], collaborative representation [10], [11], and low-rank representation [12], [13]. Nevertheless, caused by the dark current that exists in hyperspectral sensors, HSIs always contain massive noise that leads to many outliers in the classification results [14]. To alleviate this impact, various smoothing methods, such as wavelet denoising [15], total variation [16], and bilateral filtering [17], have been combined with the above spectrum classifiers to form spectral–spatial classification. Although such smoothing-based methods could improve HSI classification performance significantly, they still lack adaptability to land covers with various shapes and sizes. To make further use of spatial information, researchers have exploited the spatial relativities between land covers to reveal the latent topologies of HSIs, such as graph- [18], superpixel- [19], and morphology-based [20] methods. By modeling the spatial topologies of HSIs explicitly, the potential interaction between adjacent land covers could be employed to improve the classification performance. However, these methods usually require handcrafted features, which cannot capture complex spectral–spatial structure and nonlinear relationship, limiting their application scope.

Unlike traditional machine learning algorithms, deep neural architectures can extract high-level features automatically without handcrafted feature engineering. Up to the present,

deep-learning-based methods have made milestone breakthroughs in many fields, including computer vision [21], natural language processing [22], and HSI classification. Since the convolutional neural network (CNN) has the natural ability in extracting spectral–spatial features, most of the deep-learning-based HSI classification methods take CNN as the backbone. From the 1-D CNN [23] to the 2-D CNN [24] and 3-D CNN [25], also from the single-channel CNN [24] to the two-channel CNN (TCCNN) [26] and multichannel CNN (MCCNN) [27], numerous CNN-based HSI classification methods have been explored. As the network architectures get more and more complex and huge, Zhong *et al.* [28] and Wang *et al.* [29] introduced the residual connection [30] and dense connection [31], respectively, to make the deep HSI classification networks easier to train. Besides, to overcome the defect of shape-fixed kernels in regular convolutions, Liu *et al.* [32] and Zhu *et al.* [33] proposed the deformable convolutions that could dynamically adjust the kernel shapes according to the spatial content of HSI. Recently, the attention mechanism has also been applied to HSI classification, such as the double-branch dual-attention (DBDA) network [34] and spectral–spatial attention network [35]. Although multifarious CNNs have so far been applied to HSI classification, they still suffer from some limitations, including: 1) the difficulty in capturing complex topological information; 2) the short-range spatial dependence between land covers; and 3) the huge amount of network parameters to be learned.

Recently, the graph convolutional network (GCN) has attracted increasing attention due to its ability to process arbitrarily structured data. In contrast to the parameter-shared and shape-fixed kernels in regular CNNs, GCNs can learn adaptive kernel parameters according to the specific distribution of land covers and perform flexible convolutions on arbitrarily irregular land-cover regions. Therefore, GCNs have been utilized to learn the correlations between different land covers and model their spatial topologies on graphs. For instance, Mou *et al.* [36] proposed a nonlocal GCN to exploit the relations between any two pixels in HSI. Qin *et al.* [37] proposed a semisupervised GCN for HSI classification, in which the features of each pixel could propagate along the spatial dimension according to their spatial distance and spectral similarity. However, such a fashion of taking each pixel as a graph node could lead to a tremendous amount of computation, which limits their applications on general computers. To deal with this issue, Hong *et al.* [38] proposed to use subgraphs sampled from labeled samples to train the network, which could work on large HSIs but only make fragmentary use of spatial information. Wan *et al.* [39] used superpixels instead of pixels to be as the graph nodes, which enabled GCNs to model large-scale spatial structures of HSIs with acceptable computational overhead. Nevertheless, since the superpixel-based GCN can generate superpixel-level features only, it cannot extract subtle features for each pixel. As a result, the classification maps exhibit significant limitations, including the sensitivity to over smoothing and the spurious boundaries between cross-class regions. To alleviate this contradiction, Liu *et al.* [40] proposed a CNN-enhanced GCN (CEGCN) by integrating a CNN and GCN into a single network, in which the CNN could generate pixel-level features to supplement the superpixel-level features of GCN. However, since such GCNs conduct on graphs that are limited to a single scale defined by the presegmentation, the spatial information of HSIs could not be fully utilized.

To employ the spatial information of HSIs more comprehensively, in this work, we propose the multilevel superpixel structured graph U-Net (MSSGU) to explore their spatial topologies on multilevel graphs. Different from the fashion of applying multiscale convolutions on single-level graphs in [39], in our method, the HSI is unfolded on multilevel graphs to learn different-scale features. As mentioned before, due to the contradiction between the limited training samples and complex spatial topologies in HSIs, it is hard for a network to automatically learn the subtle hierarchical relations between scattered and difform land covers. Hence, same as [39] and [40], we define the multilevel graph structures in the preprocessing stage. Specifically, we initialize the HSI to a primary segmentation with each pixel as a superpixel, and then, a new segmentation with any number of superpixels can be acquired by progressively merging adjacent superpixels in this HSI [41]. Because the region merging is a progressive process, any segmentation can be further merged into a coarser one with fewer superpixels. In this way, we can get a series of hierarchical superpixel segmentations from fine to coarse. Then, for each segmentation, we convert it into a graph by establishing edges between adjacent superpixels and averaging each superpixel into a node. Moreover, based on the merging relations between hierarchical superpixels, we further establish the pooling and unpooling functions to transfer features between any two adjacent-level graphs. With the pooling and unpooling functions, we can integrate these different-level graphs into a single network, thereby enabling them to work collaboratively. As shown in Fig. 1, the input HSI is first fed into the encoders, in which different-scale features can be extracted on different-level graphs in a fine-to-coarse manner. Then, different-scale features are progressively fed into the decoders to fuse from coarse to fine. The skip connections between the encoder–decoder pairs ensure that the features will not be lost due to scale changes. However, as discussed in [40], taking each pixel as a node leads to a huge amount of computation, which blocks learning pixel-level subtle features on graphs. To alleviate this issue, in the first encoder and decoder, we use a CNN instead of GCN to extract and fuse pixel-level features. Also, in contrast to concatenating different-scale features at once, the feature fusion of MSSGU occurs gradually from coarse to fine, which can better fuse different-scale information and naturally adapt to the pixelwise classification task. Such a hybrid U-Net can exploit spatial topologies of HSIs hierarchically and generate multiscale subtle features for each pixel, which brings state-of-the-art performance on three benchmark datasets.

The main contributions of this work are summarized as follows.

1) We propose the multilevel structured graphs based on progressive superpixel merging to model the spatial topologies of HSIs in a multiscale hierarchical manner. To integrate different-level graphs, we also propose

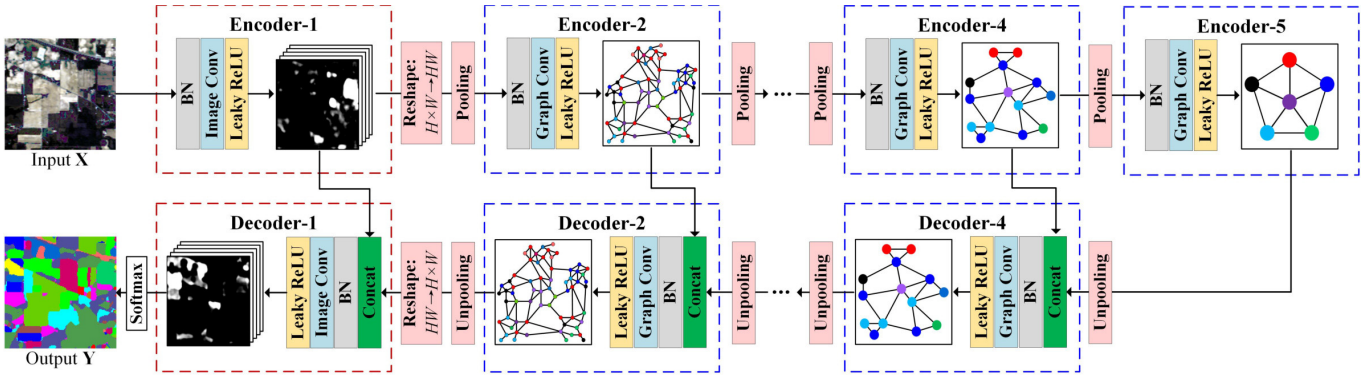

Fig. 1. Architecture of the proposed MSSGU, where $\mathbf{X}$ and $\mathbf{Y}$ denote the input HSI and output probability map of the network, and H and W denote the height and width of the input HSI, respectively. The first encoder and decoder (in the red dotted box) adopt image convolutions, and the remaining ones (in the blue dotted boxes) adopt graph convolutions.

the pooling and unpooling functions that can linearly transfer features from one graph to another, thereby enabling them to work jointly in a single network.

2) With the multilevel graphs and (un)pooling functions, we propose the MSSGU to learn different-scale features of HSIs on different-level graphs. Moreover, in contrast to concatenating or adding different-scale features straightforwardly, MSSGU fuses them in a coarse-to-fine manner, which can progressively refine fused features and naturally adapt to the pixelwise classification task.
3) To make the multilevel graphs more adaptive, we employ the attention mechanism to measure the similarity between adjacent nodes. Also, to reduce the computation, we improve the image convolution and use it to extract and fuse the pixel-level features, which can work well with the graph convolutions to achieve satisfactory performance.

The rest of this article is organized as follows. Section II introduces the methodology of the MSSGU. Section III exhibits the comparative experimental results. Section IV further analyzes the effectiveness of MSSGU. Section V provides the conclusion for this work.

II. Proposed Method

Let the input and output of MSSGU be $\mathbf{X} \in \mathbb{R}^{H\times W\times B}$ and $\mathbf{Y} \in \mathbb{R}^{H\times W\times C}$, respectively; the supervised dataset is denoted as $\mathcal{D} = \{(\mathbf{X}(\mathbf{p}_t), \mathbf{L}(\mathbf{p}_t))\}_{t=1}^{N}$, where $\mathbf{L}(\mathbf{p}_t)$ denotes the one-hot label of the pixel $\mathbf{X}(\mathbf{p}_t)$ at the spatial location $\mathbf{p}_t = (x, y)$, H, W, and B are the height, width, bands of the input HSI, and N and C denote the numbers of training samples and classes, respectively. Unlike the methods of taking small HSI patches as network inputs, our method takes the whole HSI as input, and then, MSSGU is constructed. Since superpixel segmentation can adaptively extract local spatial regions, our method can learn spatial structure more comprehensively without image patching procedure [39], [40].

In this section, we first introduce the methodology of constructing multilevel graphs, as well as the pooling and unpooling functions used to transfer features between them. Next, we simply introduce the graph convolution and image convolution used in our method. Finally, we integrate all the proposed mechanisms above and give the inference process of MSSGU and its corresponding training method.

A. Multilevel Superpixel Structured Graphs

Region-merging-based superpixel segmentation can generate any number of superpixels by continuously merging adjacent regions [41]. Instead of converting HSI into a scale-fixed graph based on a specific segmentation [39], [40], we propose to build multilevel graphs with progressive superpixel merging techniques to help exploit the spatial topologies on different scales.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the four-connected graph of a given HSI with n nodes $\mathbf{v} \in \mathcal{V}$ and m edges $e \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$; we build an auxiliary forest with n trees in which each tree is initialized with a node in $\mathcal{V}$. To reduce the potential impact of noise and redundant information on superpixel segmentation, we first employ the principal component analysis (PCA) to the HSI to extract its first three PCs as the node features of $\mathcal{G}$. Then, the L-1 distance is used to calculate the edge weights between two adjacent nodes, i.e., $e_{i,j} = \|\mathbf{v}_i - \mathbf{v}_j\|_1$. Next, the Boruvka algorithm [42] is applied to $\mathcal{G}$ to compute its minimum spanning tree (MST). Meanwhile, once an edge $e_{i,j}$ is added into the MST, the nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ in the auxiliary forest are connected, and the number of trees is reduced by one. By repeating merging trees in this manner until Z trees in the auxiliary forest are left, we can obtain Z spatially connected and spectrally similar components, i.e., superpixels [41]. Finally, by averaging each superpixel into a node and establishing edges between adjacent nodes, we can create a superpixel-structured graph containing Z nodes. In this way, we can further build different-level graphs with different numbers of nodes, which are then used to reveal the spatial topologies of HSI on different scales. The process of constructing multilevel graphs with the progressive superpixel merging technique is shown in Fig. 2.

Let $\{\mathcal{S}^{(l)}, \mathcal{G}^{(l)}\}_{l=1}^{L}$ be the L superpixel segmentations and their corresponding structured graphs, where $\mathcal{S}^{(l)} = \{\mathcal{C}_j^{(l)}\}_{j=1}^{Z^{(l)}}$ denotes the lth segmentation consisted of $Z^{(l)}$ superpixels, and $\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)})$ denotes the graph structured from $\mathcal{S}^{(l)}$, $Z^{(1)} > Z^{(2)} > \cdots > Z^{(L)}$. Specifically, to ensure

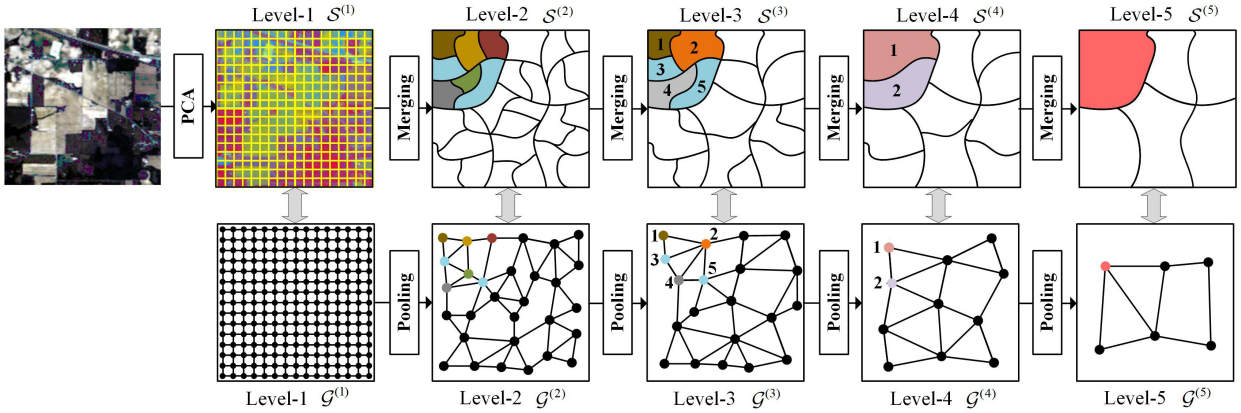

Fig. 2. Process of constructing multilevel graphs with progressive superpixel merging technique, where $\mathcal{S}^{(l)}$ and $\mathcal{G}^{(l)}$ denote the lth superpixel segmentation and its structured graph, respectively. Each superpixel in $\mathcal{S}^{(l)}$ corresponds to a unique node in $\mathcal{G}^{(l)}$. Since $\mathcal{S}^{(l)}$ is generated by merging some adjacent superpixels in $\mathcal{S}^{(l-1)}$ (e.g., $\{\mathcal{C}_1^{(3)}, \mathcal{C}_2^{(3)}, \mathcal{C}_3^{(3)}\} \to \mathcal{C}_1^{(4)}$ and $\{\mathcal{C}_4^{(3)}, \mathcal{C}_5^{(3)}\} \to \mathcal{C}_2^{(4)}$, where $\mathcal{C}_j^{(l)}$ denotes the jth superpixel in $\mathcal{S}^{(l)}$), the $\mathcal{G}^{(l)}$ can be also obtained by merging the related nodes in $\mathcal{G}^{(l-1)}$ [see $\mathcal{G}^{(3)}$ and $\mathcal{G}^{(4)}$]. Such merging relations between hierarchical superpixels are used to construct the graph pooling and unpooling functions.

that pixel-level features can be achieved, we define the raw four-connected graph of HSI as the initial graph $\mathcal{S}^{(1)}$ with $Z^{(1)} = H \times W$. In practice, the nodes $\mathcal{V}^{(l)}$ and edges $\mathcal{E}^{(l)}$ are usually represented by a node matrix $\mathbf{H}^{(l)} \in \mathbb{R}^{Z^{(l)} \times D}$ (D is the feature dimension) and an adjacency matrix $\mathbf{A}^{(l)} \in \mathbb{R}^{Z^{(l)} \times Z^{(l)}}$, respectively, where the ith row of $\mathbf{H}^{(l)}$ denotes the ith node in $\mathcal{V}^{(l)}$, and the ith row and the jth column of $\mathbf{A}^{(l)}$ denote the edge connecting the ith and jth nodes. Then, we can define the adjacency matrix of the unweighted graph $\mathcal{G}^{(l)}$ as

$$\mathbf{A}_{i,j}^{(l)} = \begin{cases} 1, & \text{if } \mathcal{C}_i^{(l)} \text{ and } \mathcal{C}_j^{(l)} \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where the superpixels $\mathcal{C}_i^{(l)}$ and $\mathcal{C}_j^{(l)}$ are adjacent when they spatially border on each other in the HSI, as shown in Fig. 2.

By using the progressive merging relations in hierarchical superpixels, i.e., any larger superpixel is generated by merging a set of smaller superpixels (see Fig. 2), we can also define the feature pooling and unpooling rule between different-level graphs. Let $\mathbf{Q}^{(l)} \in \mathbb{R}^{Z^{(l-1)} \times Z^{(l)}}$ $(2 \le l \le L)$ be the association matrix between $\mathcal{G}^{(l-1)}$ and $\mathcal{G}^{(l)}$; we can calculate it by

$$\mathbf{Q}_{i,j}^{(l)} = \begin{cases} 1, & \text{if } \mathcal{C}_i^{(l-1)} \subseteq \mathcal{C}_j^{(l)} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Then, the pooling function from $\mathcal{G}^{(l-1)}$ to $\mathcal{G}^{(l)}$ is defined as

$$\mathbf{H}^{(l)} = \text{Pool}\left(\mathbf{H}^{(l-1)}\right) = \left(\hat{\mathbf{Q}}^{(l)}\right)^{\text{T}} \mathbf{H}^{(l-1)} \tag{3}$$

where $\hat{\mathbf{Q}}^{(l)}$ denotes the normalized association matrix by column, i.e., $\hat{\mathbf{Q}}_{i,j}^{(l)} = \mathbf{Q}_{i,j}^{(l)} / \sum_m \mathbf{Q}_{m,j}^{(l)}$. Also, we can define its unpooling function as

$$\hat{\mathbf{H}}^{(l-1)} = \text{Unpool}\left(\mathbf{H}^{(l)}\right) = \mathbf{Q}^{(l)} \mathbf{H}^{(l)}. \tag{4}$$

Note that, because the nodes in $\mathcal{G}^{(l)}$ are fewer than that in $\mathcal{G}^{(l-1)}$, the unpooled $\hat{\mathbf{H}}^{(l-1)}$ contains less information than $\mathbf{H}^{(l-1)}$. With the pooling and unpooling functions, we can transfer features between any two adjacent-level graphs. Moreover, since the pooling and unpooling functions are linearly differentiable, they can be embedded into the network to link different-level graphs and enable them to model spectral–spatial structures of HSIs collaboratively.

B. Convolutions on Graphs

By converting the HSI into multilevel graphs, we can investigate its spatial topologies with the GCNs. Recently, the GCN has proved to be a powerful tool for learning features on graphs [43], [44]. Since the graph data are irregular in topological space, conducting convolutions on a graph requires adjacency relations between nodes to guide the flow of information. Because the graph $\mathcal{G}^{(l)}$ is unweighted, i.e., each element in the adjacency matrix $\mathbf{A}^{(l)}$ is equal to 0 or 1, we introduce the attention mechanism to adaptively measure the similarity between every two adjacent nodes. Considering a graph with the node matrix $\mathbf{H}$ and adjacency matrix $\mathbf{A}$, the attention-based graph convolution $f_{\mathcal{G}}$ used in our method is defined as

$$f_{\mathcal{G}}(\mathbf{H}, \mathbf{A}) = \sigma\left(\mathbf{D}^{-\frac{1}{2}} \hat{\mathbf{A}} \mathbf{D}^{-\frac{1}{2}} \mathbf{H} \mathbf{W} + \mathbf{B}\right) \tag{5}$$

with

$$\hat{\mathbf{A}} = \text{Sigmoid}\left((\mathbf{H}\mathbf{W}_\theta)(\mathbf{H}\mathbf{W}_\theta)^{\text{T}}\right) \odot \mathbf{A} + \lambda \mathbf{I} \tag{6}$$

where $\sigma(\cdot)$ denotes the leaky ReLU activation function [45], $\mathbf{D}$ is the degree matrix of $\hat{\mathbf{A}}$ (i.e., $\mathbf{D}_{i,i} = \sum_j \hat{\mathbf{A}}_{i,j}$), $\mathbf{W}$, $\mathbf{W}_\theta$, $\mathbf{B}$, and λ are learnable parameters, the operator $\odot$ denotes the Hadamard product, and $\mathbf{I}$ is the identity matrix used to improve learning stability by adding self-loop to each node [44]. The sigmoid function is used to compress the edge weights in the range of (0, 1) to prevent numerical instability. The process of convolution on a graph can be seen in Fig. 3(a).

C. Convolutions on Images

HSI classification is the task of assigning labels to pixels rather than superpixels. Limited by the shape-fixed kernels and massive parameters to be learned, it is hard for a regular CNN

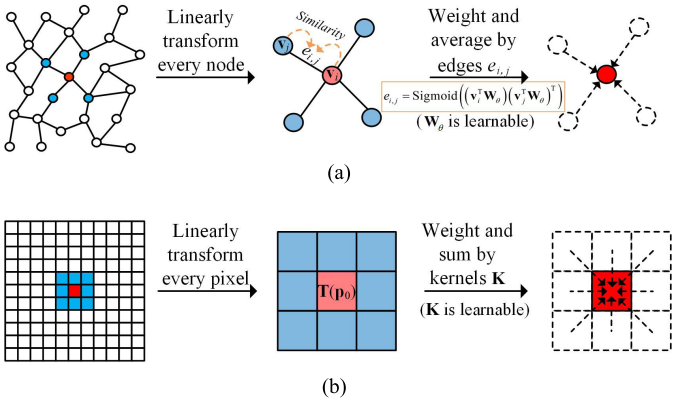

Fig. 3. Processes of convolutions on (a) graph and (b) image, respectively, where the red regions denote the central node or pixel, and the blue ones denote their neighbors.

to model spatial structures of HSI on a large scale. However, compared with applying graph convolution on pixel-structured graphs, the image convolution can generate pixel-level subtle features with acceptable computation. Hence, we propose to use image convolution instead of graph convolution to generate and fuse the pixel-level features.

The standard convolution is to weight and sum a 3-D image cube sampled in a receptive field (square window) with a 3-D kernel. Compared with the graph convolution in (5), the image convolution contains a large number of parameters to be learned due to the 3-D kernels. As shown in Fig. 3(a), the graph convolution is such a process of linearly transforming every node and then aggregating the features from the neighborhood to the central node according to the edge weights. If we replace the edge weights in every neighborhood of the graph $\mathcal{G}^{(1)}$ with the same learnable parameters, the graph convolution can degenerate into an image convolution composed of a pixelwise convolution and a depthwise convolution, as shown in Fig. 3(b). For each pixel at the location $\mathbf{p}_0 = (x, y)$ on input image, the improved image convolution with fewer parameters can be expressed as

$$\mathbf{T}_i^*(\mathbf{p}_0) = \sigma\left(\sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{K}_i(\mathbf{p}_n)\mathbf{T}(\mathbf{p}_0 + \mathbf{p}_n)\mathbf{W}_i + b_i\right) \tag{7}$$

where $\mathbf{T}$ denotes the input image, $\mathbf{W}_i$, $\mathbf{K}_i$, b_i, and $\mathbf{T}_i^*$ are the ith spectral 1-D kernel, spatial 2-D kernel, bias, and output feature band, respectively, $\mathcal{R}$ defines a square sampling grid (e.g., $\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$ defines a sampling grid with a spatial size of 3×3), and $\mathbf{p}_n$ enumerates the locations in $\mathcal{R}$. Compared with the standard convolution, the convolution in (7) has fewer parameters to be learned, which is beneficial to the task with limited samples. For the convenience of expression, we record (7) as $\mathbf{T}^* = f_{\mathcal{I}}(\mathbf{T})$.

D. Exploring Multilevel Topologies of HSIs

Multiscale GCNs have been applied to HSI classification [39], [40]. However, applying multiscale convolutions on single-level graphs could lead to the inadequate use of spatial information. Also, the feature fusion mechanism of simply concatenating or adding different-scale features may lead to scale conflict, i.e., small land covers may be more inclined to the small-scale features of low-level graphs but incompatible with the large-scale features of high-level graphs. To exploit spatial topologies of HSIs on multilevel graphs and improve the stability of multiscale feature fusion, we introduce the U-Net architecture.

Specifically, as shown in Fig. 1, the HSI is first put into the encoder-1 to learn pixel-level features using the image convolution. Then, the pixel-level features are transferred to different-level graphs to learn different-scale features in turn. Assuming that L encoders are contained in the MSSGU, we can give its feature extraction process as

$$\mathbf{H}^{(l)} = \begin{cases} \mathrm{R}_{\downarrow}\left(f_{\mathcal{I},\mathrm{BN}}(\mathbf{X})\right), & l = 1 \\ f_{\mathcal{G},\mathrm{BN}}\left(\mathrm{Pool}\left(\mathbf{H}^{(l-1)}\right), \mathbf{A}^{(l)}\right), & 2 \le l \le L \end{cases} \tag{8}$$

where $f_{\mathcal{I},\mathrm{BN}}(\cdot)$ and $f_{\mathcal{G},\mathrm{BN}}(\cdot)$ denote combining $f_{\mathcal{I}}(\cdot)$ and $f_{\mathcal{G}}(\cdot)$ with the batch normalization function, respectively, $\mathrm{R}_{\downarrow}(\cdot)$ denotes flattening the spatial dimension of the feature map (i.e., $\mathbb{R}^{H\times W\times D} \rightarrow \mathbb{R}^{HW\times D}$), and $\mathbf{H}^{(l)}$ is the output feature map of the lth encoder. The encoders can learn different-scale features on different-level graphs, which can exploit the spatial structures of HSI more comprehensively.

Instead of simply concatenating or adding different-scale features, we propose to fuse them from coarse to fine progressively using the decoders of MSSGU. Let $\mathbf{F}^{(l)}$ denote the fused feature map of the lth decoder; we have

$$\mathbf{F}^{(l)} = \begin{cases} f_{\mathcal{G},\mathrm{BN}}\left(\left[\mathrm{Unpool}\left(\mathbf{F}^{(l+1)}\right), \mathbf{H}^{(l)}\right], \mathbf{A}^{(l)}\right), & 2 \le l < L \\ f_{\mathcal{I},\mathrm{BN}}\left(\mathrm{R}_{\uparrow}\left(\left[\mathrm{Unpool}\left(\mathbf{F}^{(2)}\right), \mathbf{H}^{(1)}\right]\right)\right), & l = 1 \end{cases} \tag{9}$$

where the operator $[\ldots]$ denotes concatenating feature maps along channel dimension and $\mathrm{R}_{\uparrow}(\cdot)$ denotes expanding the original spatial dimension of the flattened data (i.e., $\mathbb{R}^{HW\times D} \rightarrow \mathbb{R}^{H\times W\times D}$). Particularly, $\mathbf{F}^{(L)} = \mathbf{H}^{(L)}$. The decoders can successively fuse the different-scale features learned in encoders and, finally, output the pixel-level multiscale feature map $\mathbf{F}^{(1)}$. Compared with concatenating different-scale features straightforwardly, the progressive fusion mechanism is more compatible with the pixelwise HSI classification task. Finally, we utilize a softmax classifier to calculate the probability map $\mathbf{Y}$, i.e., for each pixel at location $\mathbf{p}_0 = (x, y)$ on the fused feature map $\mathbf{F}^{(1)}$, we have $\mathbf{Y}(\mathbf{p}_0) = \mathrm{Softmax}(\mathbf{F}^{(1)}(\mathbf{p}_0))$.

The cross-entropy loss function is adopted to train our network. Besides, considering the class imbalance in the training samples, we propose to give the smaller class higher priority in backpropagation according to the number of samples in each class. Let N_c $(1 \le c \le C)$ be the number of training samples of the cth class; we weight the loss of the cth class with ${N_c}^{-1}$. For the training samples $\mathcal{D} = \{(\mathbf{X}(\mathbf{p}_t), \mathbf{L}(\mathbf{p}_t))\}_{t=1}^{N}$, we calculate the network loss by

$$\mathcal{L} = -\sum_{t=1}^{N}\sum_{c=1}^{C} {N_c}^{-1}\mathbf{L}_c(\mathbf{p}_t)\log(\mathbf{Y}_c(\mathbf{p}_t)) \tag{10}$$

where $\mathbf{Y}_c(\mathbf{p}_t)$ and $\mathbf{L}_c(\mathbf{p}_t)$ denote the predicated probability and real probability of the pixel $\mathbf{X}(\mathbf{p}_t)$ that belongs to the cth class, respectively.

TABLE I
NETWORK CONFIGURATION OF THE MSSGU

Level	Type of convolution	Number of nodes	Convolution in encoder	Convolution in decoder
1	image	–	5×5@128	5×5@128
2	graph	2048	@64	@64
3	graph	1024	@32	@32
4	graph	512	@16	@16
5	graph	256	@8	–

III. EXPERIMENTS

To estimate the proposed MSSGU, we adopt three public HSI datasets as the benchmark, including the Indian Pines, the University of Pavia, and the Salinas datasets. There are totally seven deep-learning-based HSI classification methods used to compare with the MSSGU, including the deep CNN (DCNN) [24], TCCNN [26], MCCNN [27], spectral–spatial residual network (SSRN) [28], fast dense spectral–spatial convolutional (FDSSC) network [29], DBDA [34], and CEGCN [40]. We adopt the overall accuracy (OA), average accuracy (AA), and kappa statistics (KPP) as the evaluation criteria of classification performance.

A. Network Configuration

As shown in Fig. 1, the first-level codecs use image convolutions, and the others adopt graph convolutions. For the second-level codecs, the number of nodes (or superpixels) should satisfy such a condition: the pixels in each superpixel belong to the same category as much as possible. To adapt the three benchmark HSIs, we set the number of nodes in the second-level codecs to 2048. For the subsequent codecs, we suggest halving the numbers of nodes in turn, i.e., [1024, 512, 256]. $\mathbf{W}_\theta$ in (6) maps each node to a 128-D vector to learn the weights of edges. The detailed scales and convolution parameters of each level encoder and decoder are shown in Table I. All the activation functions in the MSSGU adopt the leaky ReLUs [45]. Moreover, Adam optimizer [46] is used to train our network with the learning rate equal to 5×10^{-4} and epochs equal to 6×10^2. Our code[1] is based on the Python-3.6 and PyTorch-1.7, which runs on the computing platform consisted of a GTX-1080Ti GPU with 11-GB video memory and an i7-8700K CPU with 32-GB memory.

B. Dataset

The Indian Pines dataset is composed of 200 imagery bands with an image size of 145 × 145, which was captured over the agricultural site in the Indian Pines. There are 10 366 labeled samples of 16 terrain classes contained in this HSI, in which 5%, 1%, and 94% of samples of each class are randomly partitioned into disjoint training, validation, and testing samples, as detailed in Table II.

The University of Pavia dataset contains 103 bands with a spatial size of 610 × 340, which was captured over the University of Pavia. A total of 42 776 samples of nine terrain classes are labeled in this HSI, in which 0.5%, 0.5%, and 99% of samples of each class are randomly partitioned into disjoint training, validation, and testing sets, as detailed in Table III.

The Salinas dataset contains 204 bands with an image size of 512 × 217, which was captured over the Salinas Valley. A total of 54 129 samples of 16 crops are labeled for the research, in which about 0.5%, 0.5%, and 99% of samples of each class are randomly partitioned into disjoint training, validation, and testing sets, as shown in Table IV.

[1]Code is available at https://github.com/qichaoliu/MSSG-UNet

TABLE II
CATEGORY INFORMATION OF THE INDIANA PINES DATASET

No.	Class	Train.	Val.	Test.
1	Alfalfa	3	1	50
2	Corn-notill	72	15	1347
3	Corn-mintill	42	9	783
4	Corn	12	3	219
5	Pasture	25	5	467
6	Trees/Grass	38	8	701
7	Pasture-mowed	2	1	23
8	Hay-windrowed	25	5	459
9	Oats	1	1	18
10	Soybeans-notill	49	10	909
11	Soybean-mintill	124	25	2319
12	Soybean-cleantill	31	7	576
13	Wheat	11	3	198
14	Woods	65	13	1216
15	Building-Grass	19	4	357
16	Stone-steelTowers	5	1	89
	Total	524	111	9731

TABLE III
CATEGORY INFORMATION OF THE UNIVERSITY OF PAVIA DATASET

No.	Class	Train.	Val.	Test.
1	Asphalt	34	34	6563
2	Meadows	94	94	18461
3	Gravel	11	11	2077
4	Trees/Grass	16	16	3032
5	Metalsheets	7	7	1331
6	Baresoil	26	26	4977
7	Bitumen	7	7	1316
8	Bricks	19	19	3644
9	Shadows	5	5	937
	Total	219	219	42338

TABLE IV
CATEGORY INFORMATION OF THE SALINAS DATASET

No.	Class	Train.	Val.	Test.
1	Brocoli_green_weeds_1	11	11	1987
2	Brocoli_green_weeds_2	19	19	3688
3	Fallow	10	10	1956
4	Fallow_rough_plow	7	7	1380
5	Fallow_smooth	14	14	2650
6	Stubble	20	20	3919
7	Celery	18	18	3543
8	Grapes_untrained	57	57	11157
9	Soil_vinyard_develop	32	32	6139
10	Corn_senesced_green_weeds	17	17	3244
11	Lettuce_romaine_4wk	6	6	1056
12	Lettuce_romaine_5wk	10	10	1907
13	Lettuce_romaine_6wk	5	5	906
14	Lettuce_romaine_7wk	6	6	1058
15	Vinyard_untrained	37	37	7194
16	Vinyard_vertical_trellis	10	10	1787
	Total	279	279	53571

TABLE V

INDIVIDUAL CLASS, OA, AA, AND KPP OF ALL METHODS ON THE INDIAN PINES DATASET WITH 5% TRAINING SAMPLES

Class	DCNN [24]	TCCNN [26]	MCCNN [27]	SSRN [28]	FDSSC [29]	DBDA [34]	CEGCN [40]	MSSGU
1	96.78±7.00	95.05±7.28	93.66±11.08	98.14±3.72	98.49±4.52	**99.53±1.39**	68.88±17.12	97.59±1.95
2	83.62±3.69	89.77±2.34	85.43±4.39	94.89±2.77	97.51±1.81	93.63±5.23	97.15±0.85	**97.69±0.92**
3	80.32±4.11	89.63±3.88	85.34±4.06	91.23±9.65	96.32±1.50	94.36±8.97	97.15±2.04	**99.01±0.67**
4	87.14±6.57	89.32±6.70	89.65±4.59	92.10±8.85	**97.29±2.15**	96.41±5.34	92.45±5.48	92.89±7.29
5	96.37±1.22	95.07±2.98	96.20±1.64	**97.76±0.83**	97.69±0.95	96.94±1.15	96.17±1.76	96.61±1.47
6	93.72±3.02	93.62±2.20	93.67±2.31	97.20±1.57	98.15±1.80	96.57±2.31	**99.50±0.43**	98.92±1.32
7	90.00±30.00	89.99±15.96	94.19±10.35	87.08±30.30	**96.73±6.56**	89.60±16.36	95.83±5.89	95.83±4.56
8	93.34±2.75	96.36±2.75	94.25±1.58	97.15±1.91	99.54±0.59	98.70±2.24	99.95±0.08	**100±0**
9	70.00±45.82	94.40±8.09	**100±0**	10.00±30.00	10.00±30.00	64.84±43.50	42.38±23.57	94.73±6.65
10	88.01±2.81	88.02±3.15	86.82±3.55	94.99±3.46	96.59±1.62	89.94±10.55	96.43±1.55	**97.07±3.10**
11	86.32±3.27	90.97±2.89	86.66±2.50	92.57±2.83	98.25±1.17	94.83±11.08	**98.58±0.77**	98.24±0.61
12	80.94±6.50	87.05±5.92	83.25±3.35	93.09±4.86	96.80±1.95	97.26±1.62	97.52±1.27	**97.72±1.47**
13	97.99±1.63	96.49±4.09	98.09±1.26	99.05±1.13	99.11±1.16	97.45±3.03	**99.34±0.92**	98.49±1.68
14	94.27±1.86	95.97±1.48	95.56±1.41	97.87±1.41	99.03±0.87	98.58±1.06	99.55±0.28	**99.91±0.09**
15	89.24±2.36	89.03±4.42	88.11±4.34	97.68±1.31	98.19±0.93	96.28±2.67	96.15±3.03	**99.43±0.87**
16	92.98±4.14	88.23±5.28	92.10±5.31	96.44±3.60	**97.23±2.92**	86.66±6.41	94.81±4.13	97.06±2.92
OA(%)	88.04±1.49	91.36±0.78	89.03±1.12	94.56±1.21	97.82±0.45	94.33±4.74	97.63±0.26	**98.25±0.36**
AA(%)	88.82±5.11	91.81±1.31	91.44±1.59	89.83±2.89	92.31±1.94	93.22±3.79	91.99±1.86	**97.57±0.32**
KPP(×100)	86.31±1.73	90.14±0.90	87.46±1.28	93.79±1.38	97.52±0.51	93.51±5.54	97.30±0.30	**98.01±0.41**

TABLE VI

INDIVIDUAL CLASS, OA, AA, AND KPP OF ALL METHODS ON THE UNIVERSITY OF PAVIA DATASET WITH 0.5% TRAINING SAMPLES

Class	DCNN [24]	TCCNN [26]	MCCNN [27]	SSRN [28]	FDSSC [29]	DBDA [34]	CEGCN [40]	MSSGU
1	88.91±2.34	86.97±2.73	85.06±2.78	97.77±3.31	97.47±2.17	96.73±1.60	**99.02±0.83**	98.41±1.54
2	94.68±1.20	95.78±1.86	95.19±1.74	98.41±0.91	99.34±0.39	99.32±0.38	**99.92±0.15**	99.09±0.45
3	75.95±9.72	63.12±9.42	58.48±7.91	84.04±15.73	90.26±8.22	90.04±6.67	88.44±6.78	**97.31±5.93**
4	98.17±1.68	96.51±1.85	97.84±1.37	97.56±4.03	**99.50±0.23**	97.97±1.15	94.56±1.65	95.86±1.09
5	98.87±1.18	98.09±2.11	97.68±2.90	99.82±0.34	99.52±1.01	99.12±0.79	99.98±0.04	**100±0**
6	94.76±2.70	90.76±2.53	87.29±4.80	98.54±1.22	98.44±1.26	97.97±2.23	**99.92±0.21**	99.91±0.25
7	88.95±8.47	78.01±10.27	79.71±7.09	87.60±19.99	99.05±0.68	96.84±3.01	98.50±3.50	**99.81±0.57**
8	82.50±3.58	75.56±3.89	74.53±4.09	86.59±4.78	91.90±5.19	89.46±5.97	97.81±2.89	**98.40±1.45**
9	98.96±1.41	92.37±4.53	96.81±1.89	**99.71±0.27**	99.61±0.56	98.39±1.27	98.63±1.61	98.13±1.32
OA(%)	92.00±0.63	90.08±1.20	89.04±1.19	95.41±2.63	97.75±0.31	97.12±0.69	98.58±0.33	**98.73±0.40**
AA(%)	91.31±1.53	86.35±1.61	85.84±1.61	94.45±2.95	97.23±0.65	96.21±0.96	97.42±0.82	**98.54±0.75**
KPP(×100)	89.29±0.86	86.77±1.66	85.37±1.62	93.94±3.42	97.02±0.42	96.19±0.88	98.12±0.44	**98.32±0.53**

TABLE VII

INDIVIDUAL CLASS, OA, AA, AND KPP OF ALL METHODS ON THE SALINAS DATASET WITH 0.5% TRAINING SAMPLES

Class	DCNN [24]	TCCNN [26]	MCCNN [27]	SSRN [28]	FDSSC [29]	DBDA [34]	CEGCN [40]	MSSGU
1	99.34±0.67	98.32±1.56	98.31±1.61	99.98±0.03	**100±0**	99.62±1.13	99.97±0.06	99.68±0.93
2	98.61±1.02	97.31±1.57	98.35±1.38	98.90±2.64	99.82±0.37	99.14±2.36	**100±0**	**100±0**
3	94.38±1.15	93.58±3.10	92.01±4.19	96.11±1.91	97.25±1.27	97.45±1.49	**100±0**	**100±0**
4	97.04±1.95	88.67±5.54	94.02±4.10	98.80±1.03	97.94±1.76	94.77±5.01	**99.71±0.44**	98.83±1.94
5	98.24±1.45	96.85±2.58	97.76±2.24	98.89±1.23	**99.67±0.35**	98.02±3.36	98.59±1.30	98.52±1.17
6	99.03±0.99	96.95±1.94	97.42±1.80	98.80±2.92	99.97±0.03	**99.99±0.01**	99.95±0.07	99.89±0.15
7	96.89±3.38	96.25±3.78	95.94±4.20	99.23±1.26	99.76±0.48	97.63±3.85	**99.97±0.04**	99.88±0.17
8	86.55±2.87	86.36±1.66	80.07±2.68	87.87±7.77	93.21±4.25	89.57±8.56	97.58±2.59	**99.63±0.31**
9	99.11±0.68	97.66±2.38	97.88±1.82	99.67±0.15	99.64±0.27	99.18±0.67	**100±0**	**100±0**
10	89.80±3.02	89.31±2.80	88.76±3.63	98.03±1.36	98.83±1.09	96.55±2.96	96.65±1.29	**99.32±0.75**
11	91.13±3.62	80.55±6.34	83.62±8.89	95.80±1.65	95.62±1.77	96.00±2.96	99.51±0.71	**99.95±0.06**
12	97.92±1.09	94.48±6.01	97.99±1.30	98.71±1.23	99.42±0.60	99.44±0.70	**100±0**	98.81±1.38
13	99.48±0.83	99.68±0.30	98.34±2.17	99.76±0.49	**99.89±0.26**	99.42±0.71	99.29±0.92	97.45±4.13
14	97.57±1.01	96.90±1.94	95.05±5.02	97.74±1.55	98.15±1.32	96.44±2.89	98.87±0.74	**99.02±0.97**
15	81.75±3.34	82.81±3.15	72.15±7.56	80.99±11.98	91.39±4.49	88.80±7.42	98.42±0.33	**98.89±1.13**
16	99.38±0.56	97.73±2.08	97.96±2.47	99.82±0.36	**100±0**	99.95±0.09	99.23±0.62	98.65±1.57
OA(%)	92.88±0.95	91.69±0.62	89.30±1.52	93.13±1.47	96.84±0.89	94.70±1.93	98.93±0.51	**99.45±0.14**
AA(%)	95.39±0.59	93.34±0.88	92.85±1.04	96.82±0.45	98.16±0.40	97.00±0.82	99.23±0.18	**99.28±0.32**
KPP(×100)	92.08±1.06	90.75±0.69	88.09±1.68	92.35±1.63	96.48±0.99	94.09±2.17	98.80±0.56	**99.39±0.16**

C. Classification Performance

Seven deep-learning-based methods are compared with the MSSGU on three benchmark datasets. Each experiment is repeated ten times to report the mean and standard deviation of each index. Tables V–VII detail the classification accuracies of different methods on each dataset, and the corresponding classification maps are illustrated in Figs. 4–6, respectively.

In the experiments on the Indian Pines and University of Pavia datasets, the proposed MSSGU achieves the best classification performance in the comparison results. Since

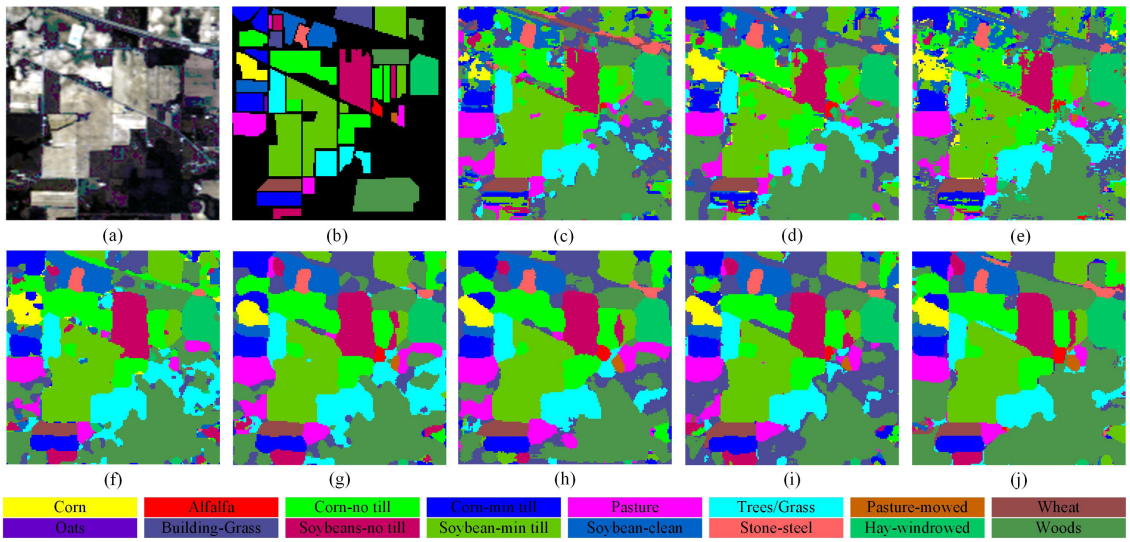

Fig. 4. Classification maps of different methods for the Indian Pines dataset. (a) False-color image. (b) Ground truth. (c) DCNN (OA = 88.04%). (d) TC-CNN (OA = 91.36%). (e) MCCNN (OA = 89.03%). (f) SSRN (OA = 94.56%). (g) FDSSC (OA = 97.82%). (h) DBDA (OA = 94.33%). (i) CEGCN (OA = 97.63%). (j) MSSGU (OA = 98.25%).

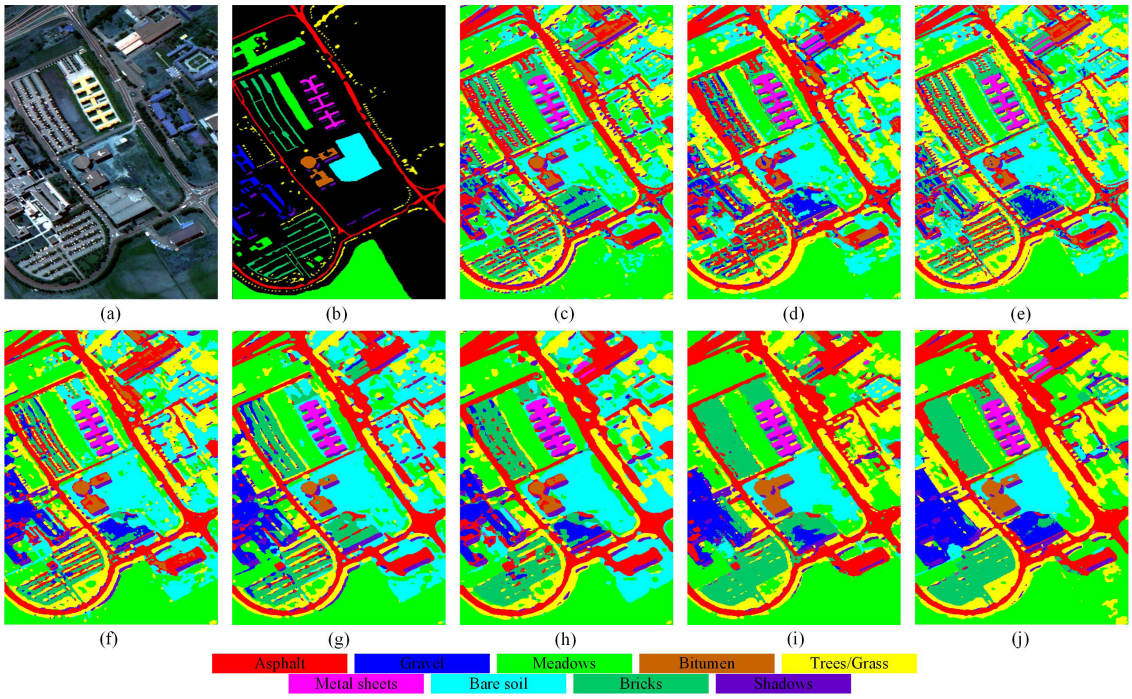

Fig. 5. Classification maps of different methods for the University of Pavia dataset. (a) False-color image. (b) Ground truth. (c) DCNN (OA = 92.00%). (d) TC-CNN (OA = 90.08%). (e) MCCNN (OA = 89.04%). (f) SSRN (OA = 95.41%). (g) FDSSC (OA = 97.75%). (h) DBDA (OA = 97.12%). (i) CEGCN (OA = 98.58%). (j) MSSGU (OA = 98.73%).

the HSIs contain land covers with various shapes and sizes, the multilevel graph learning mechanism plays an important role in modeling the complex spatial topologies. Specifically, a lower level graph with more nodes can model small land covers well, but it is hard to learn the continuously smoothed features on large objects. In contrast, a higher level graph consisted of fewer nodes can model the large land covers commendably while failing to distinguish small ones. Due to the complementary between different-scale features learned on different-level graphs, the MSSGU can adaptively learn the most suitable features for various land covers, thereby achieving the best comprehensive performance. As shown in Figs. 4 and 5, the classification maps tend to be smoothed at large homogeneous regions, which helps to suppress the outliers caused by the discrete noise in HSIs and improve the classification performance for large land covers; at small

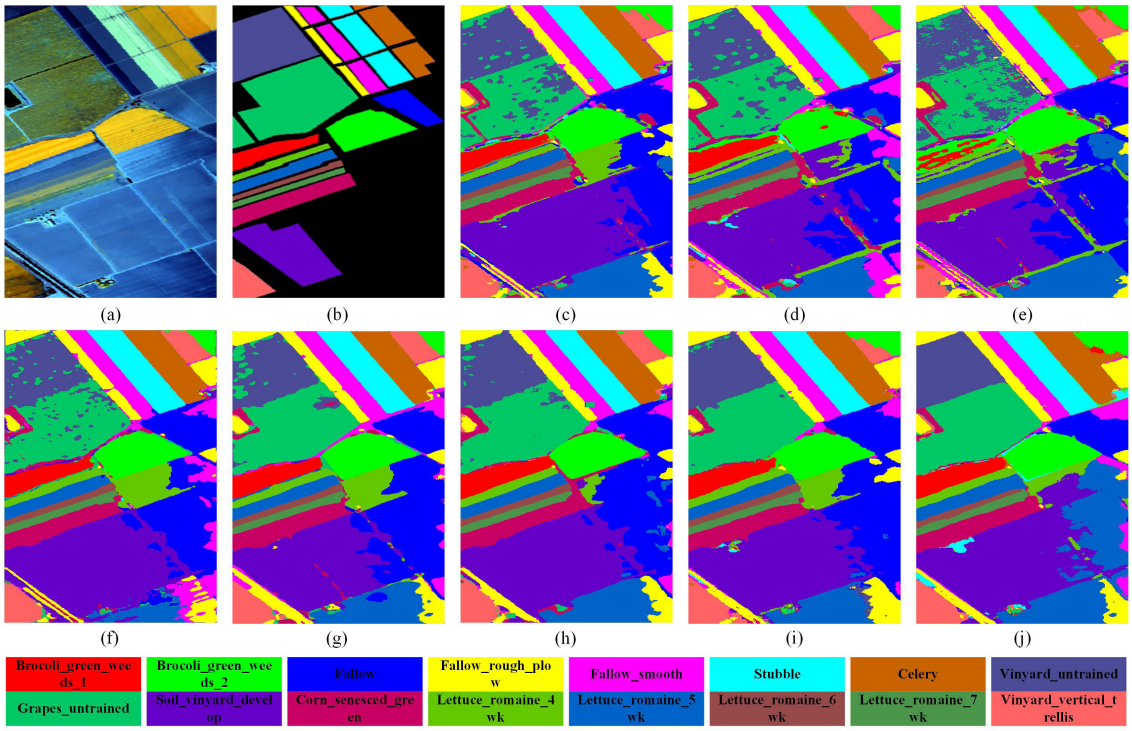

Fig. 6. Classification maps of different methods for the Salinas dataset. (a) False-color image. (b) Ground truth. (c) DCNN (OA = 92.88%). (d) TC-CNN (OA = 91.69%). (e) MCCNN (OA = 89.30%). (f) SSRN (OA = 93.13%). (g) FDSSC (OA = 96.84%). (h) DBDA (OA = 94.70%). (i) CEGCN (OA = 98.93%). (j) MSSGU (OA = 99.45%).

land-cover regions, MSSGU gives subtle classification results rather than smoothing them out, which proves the adaptive multiscale learning ability of MSSGU.

For the Salinas dataset, our method achieves a much better classification performance than the compared methods. As can be seen from Fig. 6, this HSI is almost composed of large-scale land covers. Hence, high-level graphs with large-scale features are required to model the spatial topologies of these land covers. Especially, for the class-8 and class-15 land covers that are similar in spectra but easily distinguishable in space, the MSSGU and CEGCN can distinguish them well, whereas that is hard for the CNN-based methods due to the lack of ability in modeling large-scale spatial structures. Although both the CEGCN and MSSGU can achieve excellent results on this HSI, due to the lack of hierarchical learning and fusion mechanism, the CEGCN is still inferior to the MSSGU, which also proves the effectiveness of the multilevel graph learning of MSSGU.

In summary, due to the multilevel graph learning mechanism, the MSSGU can learn different-scale features on different-level graphs and then fuse them into multiscale features with the adaptability to various land covers. Hence, the MSSGU can obtain piecewise smoothed classification results that show high consistency with the ground truths.

IV. Discussion

In this section, we will further verify the effectiveness of MSSGU by conducting multiform qualitative and quantitative experiments, including the analyses of hyperparameters, graph learning, feature extraction and fusion, small-sample learning ability, and running efficiency. If not specified, all experiments in this section are in the same configuration as in Section III.

A. Influence of Network Depth

The network depth, i.e., the number of graph levels L in the MSSGU, has a significant influence on the classification performance. Specifically, the more the graph levels, the deeper the network, and the more complex the spatial topologies can be exploited. In this section, we set the L to 1, 2, 3, 4, and 5, respectively, to test the classification performance of the MSSGU.

It is illustrated in Fig. 7 that the classification accuracies of MSSGU increase with the increase in the network depth. Especially, the biggest performance increase occurs when the depth increases to 2, which indicates that the graph convolution can be well combined with the image convolution to improve performance collaboratively. Also, on the Indian Pines and Salinas datasets, deeper MSSGU with higher level graphs further boosts the classification performance evidently; in contrast, on the University of Pavia dataset, there is little performance improvement when L is larger than 2. It is because there are more piecewise continuous land covers in the Indian Pines and Salinas datasets so that higher level graphs can provide more smoothed features for large objects, while, in the University of Pavia dataset, most of the land covers are discontinuous and fragmented; as a result, smoothed features on high-level graphs seem to have little effect. However, due to the hierarchical feature fusion architecture, the redundant graphs in the MSSGU do not cause performance degradation.

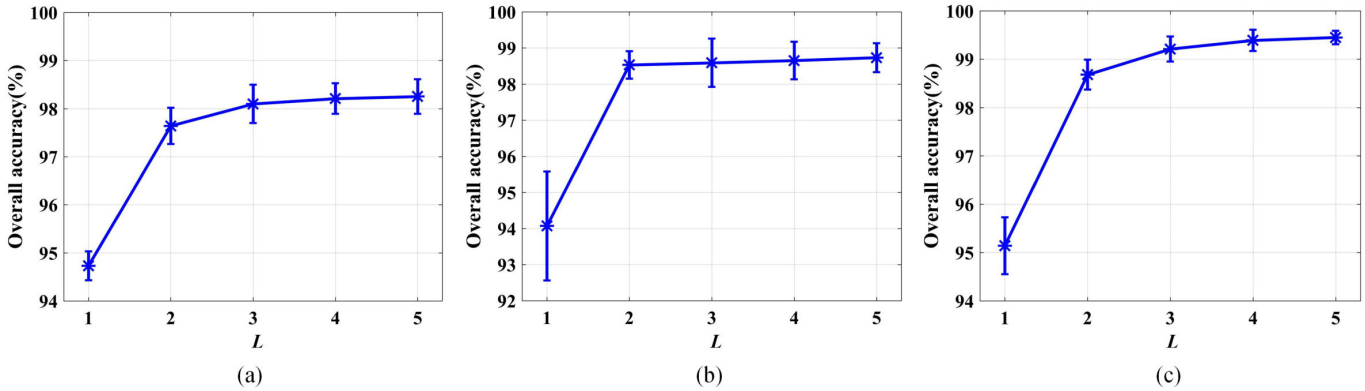

Fig. 7. Classification accuracies of MSSGU with different network depths. (a) Indian Pines. (b) University of Pavia. (c) Salinas.

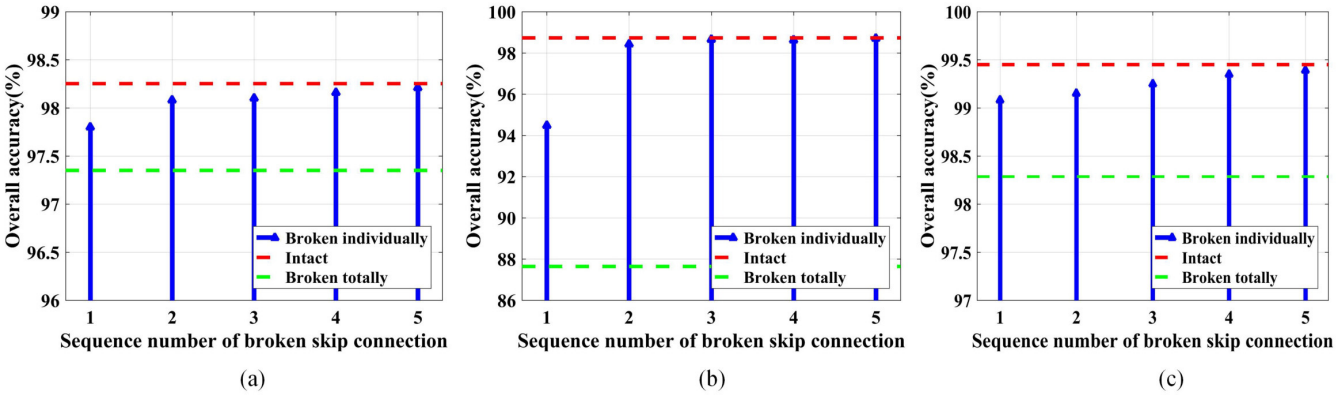

Fig. 8. Performance degradation of MSSGU when breaking the skip connections between encoder–decoder pairs. (a) Indian Pines. (b) University of Pavia. (c) Salinas.

In summary, with the increase in the network depth, more topological information of HSI can be exploited, and higher classification performance can be achieved. Moreover, benefiting from the progressive feature fusion mechanism, MSSGU can automatically adjust the fusion weights of different-scale features without worrying about the influence of ineffective features on performance.

B. Analysis of Skip Connection

As the information bridge between each encoder–decoder pair, the skip connections help decoders to fuse different-scale features generated by the encoders. To study the effectiveness of skip connections, we break all the skip connections (except for the last) in MSSGU and test its classification performance. To further analyze each skip connection, we also test the performance of MSSGU when each skip connection was individually broken. If the connection between the lth encoder–decoder pair is broken, the features generated by the lth encoder will not participate in the feature fusion. These experiments can also help analyze the importance of different-scale features to classification.

As shown in Fig. 8, in the experiments on all datasets, the performance degradation caused by breaking the skip connection between the first encoder–decoder pair is the most serious, which indicates that the pixel-level features contribute most to the final classification, especially for the University of Pavia. It implies that, although abundant superpixel-level features can be extracted by graph convolutions, they still cannot completely replace the pixel-level features generated by image convolutions. Besides, the performance degradation decreases as the sequence number of broken skip connections increases, which indicates that the features learned on lower level graphs contribute more than that on the higher level graphs to final classification. It is because the lower level graph with more nodes can generate more meticulous features than the higher level ones, and finer features can better fit the pixelwise classification task. However, when we break all the skip connections, a huge performance degradation occurs. It is because, when all skip connections are removed, the MSSGU will degenerate into a conventional encoder–decoder network. In this instance, due to the information loss when pooling features between graphs, MSSGU can employ only the spatial structure of the scale defined by the highest-level (level-5) graph, leading to coarse classification results and low accuracies. The results above indicate that each feature scale plays a role in classification, and these different-scale features can complement each other to collectively boost classification performance.

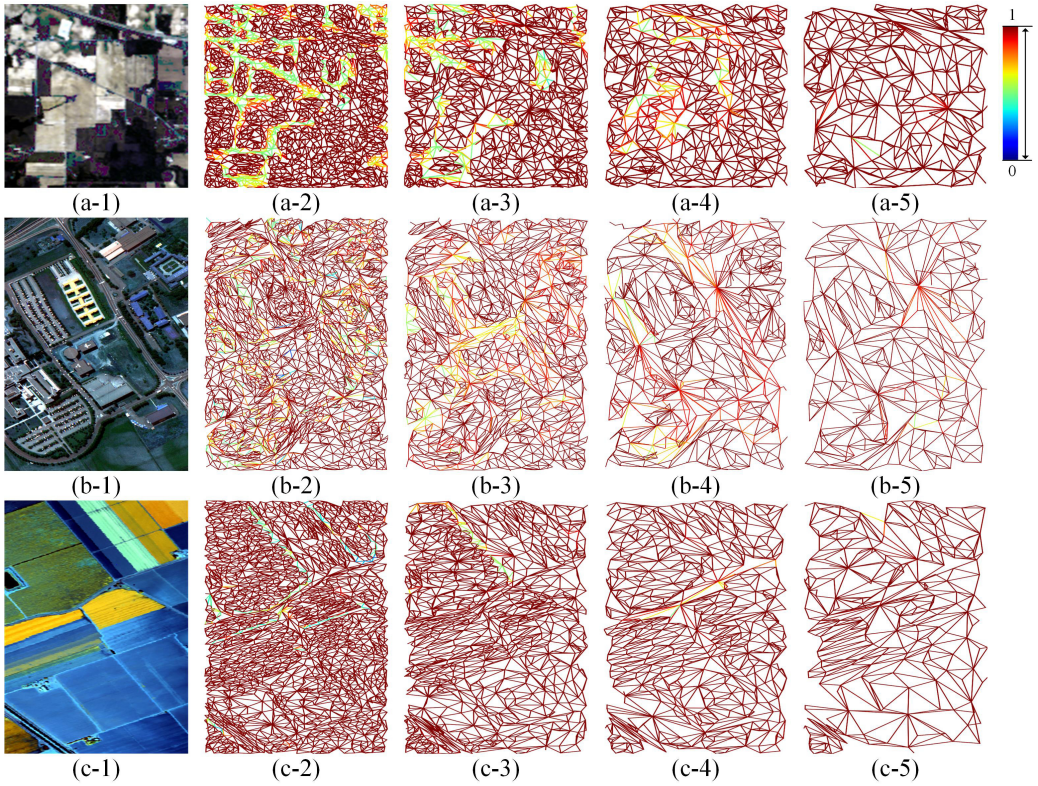

Fig. 9. Visualization of graph edges learned in each encoder where the colors of lines denote the weights. (a-1), (b-1), and (c-1) False-color images. (a-2), (a-3), (a-4), (a-5), (b-2), (b-3), (b-4), (b-5), (c-2), (c-3), (c-4), and (c-5) Visualized graph edges learned in the second to fifth encoders. (a) Indian Pines. (b) University of Pavia. (c) Salinas.

TABLE VIII
OA (%) OF MSSGU WITH DIFFERENT NEIGHBOR COUNTS

Data set	5	10	15	20	Adaptive
Indian Pines	97.84±0.18	97.63±0.33	97.69±0.43	97.73±0.34	**98.25±0.36**
University of Pavia	98.02±0.74	98.06±0.58	98.25±0.63	98.21±0.54	**98.73±0.40**
Salinas	98.99±0.34	99.20±0.25	99.19±0.33	99.20±0.29	**99.45±0.14**

C. Analysis of Graph Learning

How to establish the edges between graph nodes is a considerable problem. In this work, if two superpixels border on each other, we think of them as adjacent and create an edge between them. Such a fashion of creating edges can maintain the adjacency of land covers while blocking the information propagation between distant land covers. To prove the superiority of doing so, in this section, we compare it with the common way of establishing a fixed number of neighbors (edges) for each node using the K-nearest neighbor (KNN) algorithm in [38]. Specifically, we set the number of neighbors in the KNN to 5, 10, 15, and 20, respectively, and then compare their classification performance with the adaptive fashion. The comparison results are shown in Table VIII.

As can be seen, no matter what the number of neighbors is set to, their performance is all lower than the adaptive one, which manifests the superiority of establishing edges adaptively. As each local land cover has its unique shape, size, and neighborhood, each node in the graph also has a different number of neighbors. Therefore, establishing edges with fixed neighbor count cannot fit different HSIs, and adaptively establishing edges between nodes can improve the accuracy of topological modeling.

TABLE IX
OA (%) OF MSSGU UNDER DIFFERENT EDGE SETTINGS

Data set	Without edges	With unweighted edges	With weighted edges
Indian Pines	97.50±0.70	98.23±0.36	**98.25±0.36**
University of Pavia	97.67±0.64	98.38±0.55	**98.73±0.40**
Salinas	98.49±0.67	99.34±0.21	**99.45±0.14**

The edges in graphs enable messages to pass between adjacent nodes, thereby enabling graph nodes to model spatial structures of HSIs jointly. To prove the importance of graph edges, we conduct ablation experiments on different datasets, in which all the graph edges are removed. Moreover, because the edge weights in our method are learned from training data, to prove the effectiveness of edge learning, we also test the classification performance of MSSGU with unweighted edges, where the edge weights are equal to 0 or 1 only. As shown in Table IX, without graph edges, the classification accuracies

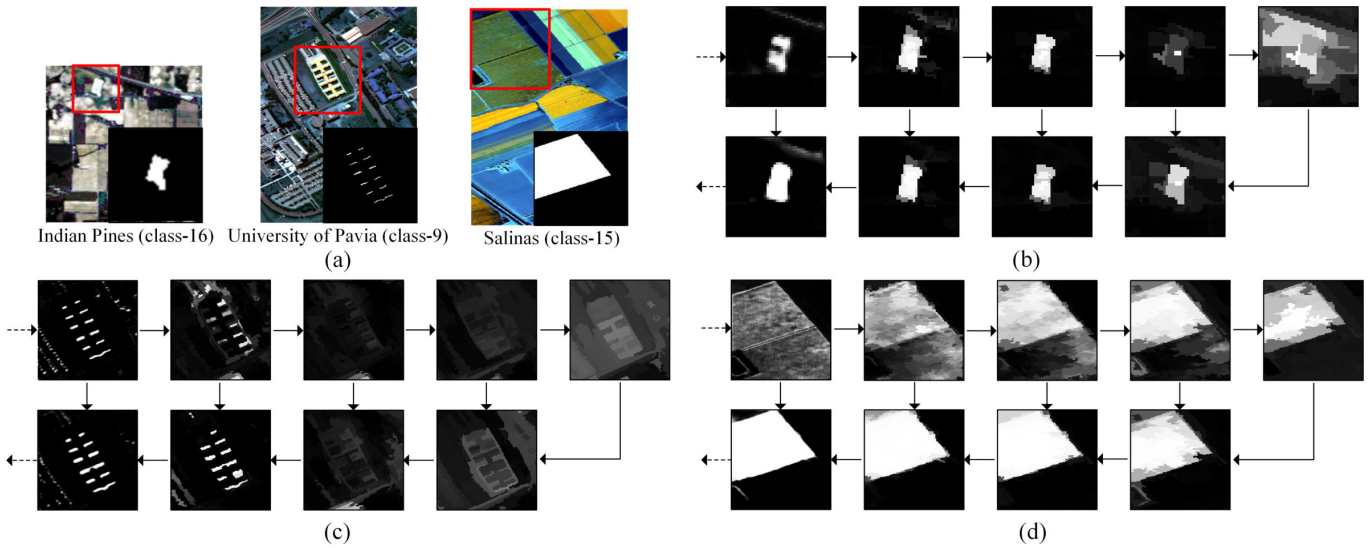

Fig. 10. Visualizations of feature maps learned in each encoder and decoder. (a) Ground truths of different datasets. (b) Indian Pines (class-16). (c) University of Pavia (class-9). (d) Salinas (class-15).

of MSSGU on all datasets exhibit obvious degradation, which proves the significance of edges in modeling spatial topologies of HSIs. Also, with only unweighted graphs, the MSSGU obtains lower performance than that with adaptively learned weights. It indicates that the edge learning mechanism can enhance the adaptability of graphs to various HSIs and, thus, improve classification performance.

Besides, to further verify the effectiveness of the graph learning in MSSGU, we visualize the graphs learned in each encoder (except for the $\mathcal{G}^{(1)}$). As shown in Fig. 9, the MSSGU can learn adaptive edge weights for different HSIs. The learned edge weights on the graph $\mathcal{G}^{(2)}$ (see a-2, b-2, and c-2) can distinguish land covers well, i.e., edges between land covers in the same category always have larger weights; in contrast, at cross-class edge regions, the edge weights tend to be smaller. However, it is interesting that, with the increase in the graph level, the weak edges decrease gradually, which indicates that higher level graphs may pay more attention to the smoothness of the feature maps rather than focusing on the distinction between different land covers. In other words, the features learned on higher level graphs have less discrimination than those learned on lower level graphs, which is also the reason why we reduce the output feature channels of encoders by order.

In summary, the edges in graphs enable different nodes to model the spatial structures of HSIs collaboratively. Compared with establishing a fixed number of neighbors for each node, adaptively building edges according to the specific distribution of land covers can enhance the accuracy of spatial modeling. Also, by learning the edge weights from training data, different-level graphs can learn complementary features for different-scale objects, thereby leading to further improvement of classification performance.

D. Analysis of Feature Extraction and Fusion

In the MSSGU, the encoders extract different-scale features from fine to coarse in turn, and then, the decoders fuse them in a coarse-to-fine manner. In this section, to reveal the processes of the feature extraction and fusion in the MSSGU, we transform the feature maps learned in each encoder and decoder into the probability map of each class and then visualize them in Fig. 10.

It is revealed that, on the Indian Pines and Salinas datasets [see Fig. 10(b) and (d)], different-scale features generated by encoders have different but complementary representations, i.e., small-scale features can distinguish the boundaries and textures of land covers; in contrast, large-scale features prefer to capture their spatial structures. Then, the decoders fuse these features from coarse to fine progressively, and the fused features gradually become refined and consistent with the ground truths. However, on the University of Pavia dataset [see Fig. 10(c)], the bottom encoders (from third to fifth) seem to fail to extract distinguishable features. It is because the class-9 land covers in this HSI are small and scattered, and it may be difficult for these high-level graphs to learn such subtle features. As a result, only low-level graphs can extract features with discrimination for this HSI. This phenomenon is also consistent with the experimental results in Section IV-A.

E. Small-Sample Learning Ability

Most HSI classification methods can achieve satisfactory performance when given enough training samples. Nevertheless, labeled samples are very limited due to the expensive human and material resources in most cases. Hence, the small-sample learning ability, i.e., the performance of HSI classification methods under very few training samples, becomes another important evaluation index. In this section, we compare the OA indices of the MSSGU with other state-of-the-art HSI classification methods under the condition of limited samples. Specifically, for each method on each dataset, five, ten, 15, 20, and 25 samples per class are randomly selected as the training samples, and one sample per class is

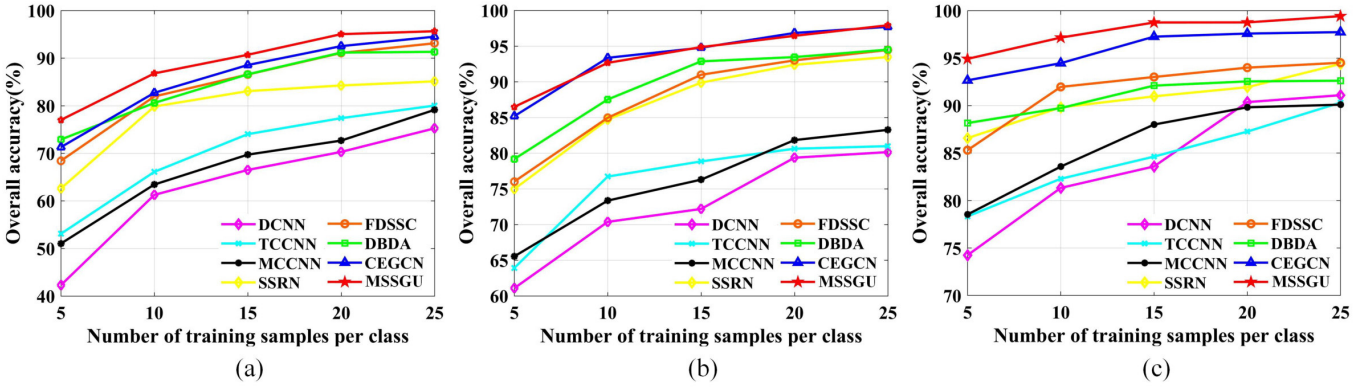

Fig. 11. Classification accuracies of different methods under different numbers of training samples on each dataset. (a) Indian Pines. (b) University of Pavia. (c) Salinas.

TABLE X
COMPARISON RESULTS ON ADDITIONAL DATASETS

Data set	Indices	DCNN [24]	TCCNN [26]	MCCNN [27]	SSRN [28]	FDSSC [29]	DBDA [34]	CEGCN [40]	MSSGU
Botswana	OA(%)	94.72±0.90	96.47±1.29	95.07±2.06	97.20±1.12	98.67±0.54	98.96±0.89	97.69±1.59	**99.23±1.18**
	AA(%)	94.49±1.07	96.70±1.28	95.08±1.97	97.62±1.03	98.77±0.49	99.14±0.66	98.05±1.34	**99.36±0.99**
	KPP(×100)	94.28±0.98	96.17±1.40	94.66±2.23	96.96±1.22	98.56±0.59	98.87±0.96	97.49±1.73	**99.16±1.28**
Kennedy Space Center	OA(%)	80.05±5.08	83.82±2.89	87.51±1.87	93.80±2.27	94.54±1.34	96.22±1.61	97.09±0.60	**98.15±0.87**
	AA(%)	74.15±5.44	77.54±3.93	82.85±2.42	91.81±3.02	92.40±1.69	94.35±1.77	95.96±0.93	**98.12±0.92**
	KPP(×100)	77.76±5.56	81.96±3.23	86.08±2.09	93.09±2.53	93.91±1.50	95.79±1.79	96.75±0.67	**97.94±0.97**

TABLE XI
COMPARISON OF RUNNING EFFICIENCY BETWEEN DIFFERENT METHODS

Data set	Time (s)	DCNN [24]	TCCNN [26]	MCCNN [27]	SSRN [28]	FDSSC [29]	DBDA [34]	CEGCN [40]	MSSGU
Indian Pines	Train.	40.2	68.3	59.7	185.7	111.6	278.3	**16.3**	24.5
	Test.	5.7	5.8	6.7	11.6	16.9	19.9	**0.6**	1.1
University of Pavia	Train.	**22.1**	40.5	36.5	120.8	70.2	136.5	163.7	317.7
	Test.	30.9	28.8	35.8	61.2	93.1	110.7	**5.2**	9.9
Salinas	Train.	137.6	209.5	192.1	176.6	142.4	230.1	**79.6**	166.1
	Test.	14.3	17.4	19.9	51.0	89.4	103.8	**3.6**	5.4
Number of parameters		295.03K	312.53K	273.55K	346.14K	2574.88K	382.35K	166.37K	**126.57K**

chosen as the validation sample. We repeat each experiment ten times to report the mean of the OA indices.

As shown in Fig. 11, the MSSGU outperforms all the compared methods on the Indian Pines and Salinas datasets. However, because the CEGCN composed of a CNN and GCN can be taken as a simplified version of the MSSGU with $L = 2$, on the University of Pavia dataset, the MSSGU achieves similar performance to the CEGCN, which is also consistent with the results in Section IV-A. These results above strongly prove the satisfactory small-sample learning ability of the MSSGU.

F. Comparison on Additional Datasets

Deep networks can achieve high accuracies when given sufficient training samples, leading to small differences in the performance of MSSGU and CEGCN. Therefore, additional experimental comparisons under limited samples are conducted to further illustrate the superiority of MSSGU. Specifically, additional two datasets, i.e., Botswana and Kennedy Space Center,[2] are adopted to evaluate our method. In each dataset, ten samples and one sample are randomly selected from each class as the training set and the validation set, respectively, and the rest are used to evaluate performance. The network configuration of MSSGU is the same as Section III-A. The comparison results are exhibited in Table X.

As can be seen, the proposed MSSGU still achieves the best classification results in all comparison methods. On the Botswana and Kennedy Space Center datasets, the OA indices of MSSGU are 1.54% and 1.06% higher than that of CEGCN; meanwhile, the AA indices of MSSGU are 1.31% and 2.16% higher than that of CEGCN, respectively. Besides, on the Botswana dataset, the results obtained by the FDSSC and DBDA are better than that of CEGCN, while, on the Kennedy Space Center dataset, the opposite is true. It indicates that the proposed MSSGU not only has outstanding classification performance but also possesses better robustness than CEGCN to various HSIs. More results (e.g., individual-class accuracies and classification maps) can be found in the Supplementary Material.

G. Comparison of Running Efficiency

To evaluate the running efficiency of our method, we compare different methods on the running time and numbers

[2]Datasets are available at: http://ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes

of network parameters. The network parameters are counted based on the Indian Pines dataset with 200 imagery bands and 16 classes. Note that, to simulate the applications in the real world, we record the time of classifying all pixels instead of only the labeled pixels. All the methods are tested on the same computing platform described in Section III-A. Each experiment is repeated ten times to report the average time cost.

As shown in Table XI, the training time cost of the proposed method is moderate and acceptable compared with other methods. Because both the CEGCN and MSSGU take the whole HSI rather than small patches as network inputs, all the pixels can be classified in parallel, which results in a shorter classification time. However, due to the more complex architecture with different-scale graphs, MSSGU needs more time than CEGCN for inference. Besides, MSSGU achieves the best classification accuracies with the fewest parameters in all compared methods, which also proves the superiority of our method. In real applications, users can adjust the depth and scales of MSSGU to achieve a tradeoff between classification accuracy, inference speed, and model size.

V. Conclusion

Different land covers usually have different shapes and scales, which leads to the complex and diversified spatial topologies of HSIs. To fully use spatial information, we propose to generate multilevel graphs by successively merging adjacent regions in HSIs to model their spatial topologies in a multiscale hierarchical manner. The pooling and unpooling functions can pass messages between different-level graphs, thereby enabling them to collaborate in the network. The attention-based graph convolution can learn the adaptive edge weights for various land covers. Also, the improved image convolution with fewer parameters can be combined well with the graph convolution to extract multiscale features for each pixel. Moreover, the fashion of progressively fusing different-scale features from coarse to fine can generate subtle fusion results for each pixel and show great fitness to the pixelwise HSI classification task. Large quantities of experiments have shown the outstanding performance of the proposed method. Considering that the construction of multilevel graphs is an unsupervised process, in future works, we would further integrate the graph construction into networks to learn more accurate graph structures under supervised learning.

References

[1] M. Govender, K. Chetty, and H. Bulcock, "A review of hyperspectral remote sensing and its application in vegetation and water resource studies," *Water SA*, vol. 33, no. 2, pp. 145–151, Dec. 2009.

[2] D. Ramakrishnan and R. Bharti, "Hyperspectral remote sensing and geological applications," *Current Sci.*, vol. 108, no. 5, pp. 879–891, 2015.

[3] G. Lu and B. Fei, "Medical hyperspectral imaging: A review," *J. Biomed. Opt.*, vol. 19, no. 1, Jan. 2014, Art. no. 010901.

[4] H. Yuan and Y. Y. Tang, "Spectral–spatial shared linear regression for hyperspectral image classification," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 934–945, Apr. 2017.

[5] G. T. Kaya, "A hybrid model for classification of remote sensing images with linear SVM and support vector selection and adaptation," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 4, pp. 1988–1997, Aug. 2013.

[6] J. Chen and R. Wang, "A pairwise decision tree framework for hyperspectral classification," *Int. J. Remote Sens.*, vol. 28, no. 12, pp. 2821–2830, Jun. 2007.

[7] Y. Y. Tang, H. Yuan, and L. Li, "Manifold-based sparse representation for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 12, pp. 7606–7618, Dec. 2014.

[8] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification via kernel sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 217–231, Jan. 2013.

[9] S. Jia, Y. Xie, G. Tang, and J. Zhu, "Spatial–spectral-combined sparse representation-based classification for hyperspectral imagery," *Soft Comput.*, vol. 20, no. 12, pp. 4659–4668, Dec. 2016.

[10] H. Su, B. Zhao, Q. Du, and P. Du, "Kernel collaborative representation with local correlation features for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 1230–1241, Feb. 2019.

[11] H. Su, Y. Yu, Q. Du, and P. Du, "Ensemble learning for hyperspectral image classification using tangent collaborative representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 6, pp. 3778–3790, Jun. 2020.

[12] Y. Gu, Q. Wang, H. Wang, D. You, and Y. Zhang, "Multiple kernel learning via low-rank nonnegative matrix factorization for classification of hyperspectral imagery," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2739–2751, Jun. 2015.

[13] W. Sun, G. Yang, B. Du, L. Zhang, and L. Zhang, "A sparse and low-rank near-isometric linear embedding method for feature extraction in hyperspectral imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 4032–4046, Jul. 2017.

[14] C. Xing, L. Ma, and X. Yang, "Stacked denoise autoencoder based feature extraction and classification for hyperspectral images," *J. Sensors*, vol. 2016, pp. 1–10, Jun. 2016.

[15] B. Demir and S. Erturk, "Wavelet denoising before support vector classification of hyperspectral images," in *Proc. IEEE Signal Process. Commun. Appl.*, Jun. 2007, pp. 1–4.

[16] Q. Yuan, L. Zhang, and H. Shen, "Hyperspectral image denoising employing a spectral–spatial adaptive total variation model," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 10, pp. 3660–3677, Oct. 2012.

[17] Y. Wang, H. Song, and Y. Zhang, "Spectral–spatial classification of hyperspectral images using joint bilateral filter and graph cut based model," *Remote Sens.*, vol. 8, no. 9, p. 748. Sep. 2016.

[18] L. Ma, M. M. Crawford, X. Yang, and Y. Guo, "Local-manifold-learning-based graph construction for semisupervised hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2832–2844, May 015.

[19] L. Y. Fang, S. T. Li, X. D. Kang, and J. A. Benediktsson, "Spectral–spatial classification of hyperspectral images with a superpixel-based discriminative sparse model," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 8, pp. 4186–4201, Aug. 2015.

[20] A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A new approach to mixed pixel classification of hyperspectral imagery based on extended morphological profiles," *Pattern Recognit.*, vol. 37, no. 6, pp. 1097–1116, 2004.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[22] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Assoc. Comput. Linguistics, Hum. Lang. Technol. (NAACL-HLT)*, Minneapolis, MI, USA. ACL, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423, doi: 10.18653/v1/N19-1423.

[23] 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.

[24] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 4959–4962.

[25] Y. Li, H. Zhang, and Q. Shen, "Spectral–spatial classification of hyperspectral imagery with 3D convolutional neural network," *Remote Sens.*, vol. 9, no. 1, p. 67, 2017.

[26] J. Yang, Y.-Q. Zhao, and J. C.-W. Chan, "Learning and transferring deep joint spectral–spatial features for hyperspectral classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4729–4742, Aug. 2017.

[27] C. Chen, J.-J. Zhang, C.-H. Zheng, Q. Yan, and L.-N. Xun, "Classification of hyperspectral data using a multi-channel convolutional neural network," in *Proc. Int. Conf. Intell. Comput.* Cham, Switzerland: Springer, Jul. 2018, pp. 81–92.

[28] 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.
[29] W. Wang, D. Shuguang, J. Zhongmin, and S. Liujie, "A fast dense spectral–spatial convolution network framework for hyperspectral images classification," *Remote Sens.*, vol. 10, no. 7, p. 1068, Jul. 2018.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[31] G. Huang, Z. Liu, and L. van der Maaten, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[32] Q. Liu, L. Xiao, J. Yang, and J. C.-W. Chan, "Content-guided convolutional neural network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 9, pp. 6124–6137, Sep. 2020.
[33] J. Zhu, L. Fang, and P. Ghamisi, "Deformable convolutional neural networks for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 8, pp. 1254–1258, Aug. 2018.
[34] R. Li, S. Zheng, C. Duan, Y. Yang, and X. Wang, "Classification of hyperspectral image based on double-branch dual-attention mechanism network," *Remote Sens.*, vol. 12, no. 3, p. 582, Feb. 2020.
[35] X. Mei *et al.*, "Spectral–spatial attention networks for hyperspectral image classification," *Remote Sens.*, vol. 11, no. 8, p. 963, Apr. 2019.
[36] L. Mou, X. Lu, X. Li, and X. X. Zhu, "Nonlocal graph convolutional networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 12, pp. 8246–8257, Dec. 2020.
[37] A. Qin, Z. Shang, J. Tian, Y. Wang, T. Zhang, and Y. Y. Tang, "Spectral–spatial graph convolutional networks for semisupervised hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 2, pp. 241–245, Sep. 2019.
[38] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 7, pp. 5966–5978, Jul. 2021, doi: 10.1109/TGRS.2020.3015157.
[39] S. Wan, C. Gong, P. Zhong, B. Du, L. Zhang, and J. Yang, "Multiscale dynamic graph convolutional network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 5, pp. 3162–3177, May 2020.
[40] Q. Liu, L. Xiao, J. Yang, and Z. Wei, "CNN-enhanced graph convolutional network with pixel- and superpixel-level feature fusion for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, pp. 1–15, 2020, doi: 10.1109/TGRS.2020.3037361.
[41] X. Wei, Q. Yang, Y. Gong, N. Ahuja, and M.-H. Yang, "Superpixel hierarchy," *IEEE Trans. Image Process.*, vol. 27, pp. 4838–4849, 2018.
[42] D. B. West, *Introduction to Graph Theory*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.
[43] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Jun. 2016, pp. 3844–3852.
[44] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2017, pp. 1–14.
[45] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Int. Conf. Mach. Learn. (ICML)*, vol. 30, no. 1, 2013, p. 3.
[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2015, pp. 1–15.

Qichao Liu (Student Member, IEEE) received the B.S. degree in computer science and technology from Nanjing University of Science and Technology, Nanjing, China, in 2015, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering.

His fields of interest include deep learning and remote sensing image processing.

Liang Xiao (Member, IEEE) received the B.S. degree in applied mathematics and the Ph.D. degree in computer science from Nanjing University of Science and Technology (NJUST), Nanjing, Jiangsu, China, in 1999 and 2004, respectively.

From 2006 to 2008, he was a Post-Doctoral Research Fellow with the Pattern Recognition Laboratory, NJUST. From 2009 to 2010, he was a Post-Doctoral Fellow with the Rensselaer Polytechnic Institute, Troy, NY, USA. Since 2013, he has been the Deputy Director of the Jiangsu Key Laboratory of Spectral Imaging Intelligent Perception, Nanjing. Since 2014, he has been the Second Director of the Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, NJUST, where he is currently a Professor with the School of Computer Science and Engineering. His fields of interest include remote sensing image processing, image modeling, computer vision, machine learning, and pattern recognition.

Jingxiang Yang (Member, IEEE) received the joint Ph.D. degree in control theory and control engineering and engineering science from Northwestern Polytechnical University, Xi'an, China, and Vrije Universiteit Brussel, Ixelles, Belgium, in 2019.

He is currently a Lecturer with Nanjing University of Science and Technology, Nanjing, China. His research interests include deep learning and its applications in hyperspectral image processing.

Zhihui Wei received the B.Sc., M.Sc., and Ph.D. degrees from Southeast University, Nanjing, China, in 1983, 1986, and 2003, respectively.

He is currently the Professor and a Doctoral Supervisor with Nanjing University of Science and Technology, Nanjing. His current research interests include mathematical image modeling, multiscale analysis, video and image coding and compressing, watermarking and steganography, speech and audio processing, mathematical methods, and machine learning techniques in data science, such as multiscale geometrical analysis, manifold learning and regularization, optimization method for big data processing, and deep learning.