CNN-Enhanced Graph Convolutional Network With Pixel- and Superpixel-Level Feature Fusion for Hyperspectral Image Classification

Article *in* IEEE Transactions on Geoscience and Remote Sensing · November 2020

DOI: 10.1109/TGRS.2020.3037361

CITATIONS
374

READS
1,746

4 authors:

Qichao Liu
Nanjing University of Science and Technology
15 PUBLICATIONS **632** CITATIONS
SEE PROFILE

Liang Xiao
Nanjing University of Science and Technology
383 PUBLICATIONS **5,335** CITATIONS
SEE PROFILE

Jingxiang Yang
Nanjing University of Science and Technology
35 PUBLICATIONS **2,436** CITATIONS
SEE PROFILE

Zhihui Wei
Nanjing University of Science and Technology
50 PUBLICATIONS **991** CITATIONS
SEE PROFILE

CNN-Enhanced Graph Convolutional Network with Pixel- and Superpixel-Level Feature Fusion for Hyperspectral Image Classification

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

***Abstract*—Recently, the graph convolutional network (GCN) has drawn increasing attention in hyperspectral image (HSI) classification. Compared with the convolutional neural network (CNN) with fixed square kernels, GCN can explicitly utilize the correlation between adjacent land covers and conduct flexible convolution on arbitrarily irregular image regions; hence, the HSI spatial contextual structure can be better modeled. However, to reduce the computational complexity and promote the semantic structure learning of land covers, GCN usually works on superpixel-based nodes rather than pixel-based nodes; thus, the pixel-level spectral-spatial features cannot be captured. To fully leverage the advantages of the CNN and GCN, we propose a heterogeneous deep network called CNN-enhanced GCN (CEGCN), in which CNN and GCN branches perform feature learning on small-scale regular regions and large-scale irregular regions, and generate complementary spectral-spatial features at pixel and superpixel levels, respectively. To alleviate the structural incompatibility of the data representation between the Euclidean data-oriented CNN and non-Euclidean data-oriented GCN, we propose the graph encoder and decoder to propagate features between image pixels and graph nodes, thus enabling the CNN and GCN to collaborate in a single network. In contrast to other GCN-based methods that encode HSI into a graph during preprocessing, we integrate the graph encoding process into the network and learn edge weights from training data, which can promote the node feature learning and make the graph more adaptive to HSI content. Extensive experiments on three data sets demonstrate that the proposed CEGCN is both qualitatively and quantitatively competitive compared with other state-of-the-art methods.**

***Index Terms*—Convolutional neural network (CNN), graph convolutional network (GCN), hyperspectral image (HSI) classification, graph encoder and decoder, feature fusion**

I. Introduction

HYPERSPECTRAL image (HSI) contains hundreds of bands captured in a continuous wavelength. Given the abundant spectral and spatial information, HSI can distinguish land-cover types at the pixel level. Hence, it has been applied to pollution monitoring, medical diagnosis, agriculture assessment, and geological exploration [1]–[4]. The premise of these applications is to precisely classify each pixel in HSI.

In previous decades, various machine learning based classifiers have been developed for HSI classification. Earlier methods usually assumed that the spectra could be separated in a high-dimensional space. As a result, determining how to establish the mapping function and find the separable hyperplane became the research goal. For example, logistic regression [5] and the extreme learning machine (ELM) [6] have been used to develop pixel-wise HSI classifiers. However, pixel-wise approaches usually produce considerable errors or outliers in the final classification maps. To alleviate this issue, kernel tricks, such as the kernel support vector machine (KSVM) [7] and multiple kernel learning [8], were exploited to improve the linear separability. However, these methods usually focus on designing classifiers while ignoring the representation and learning of features. To make full use of spectral information, typical representation-based methods have been exploited, such as sparse representation [9]–[11], low-rank representation [12], [13], and collaborative representation [14]–[16]. With representation learning, the inherent data structure of spectra could be revealed and the dependence on labeled samples could be reduced. Besides, the spatial structures of HSIs have been also explored to boost the spectral-spatial feature learning, such as graph construction [17], [18], superpixel segmentation [19], and morphological segmentation [20], [21]. By explicitly modeling the spatial structure of HSI, its spatial information could be better exploited. However, limited by manual features and empirical parameters, the above methods cannot learn robust deep feature representations from HSIs.

In contrast to traditional approaches, deep learning methods can learn adaptive and robust deep features from training data automatically. Therefore, deep learning methods have achieved remarkable success in the fields of computer vision [22], natural language processing [23], and intelligent decision [24]. Currently, many classical deep learning methods have been applied to HSI classification and have obtained promising results. For instance, the convolutional neural networks (CNNs) [25]–[27], recurrent neural networks (RNNs) [28], and stacked auto-encoders (SAEs) [29] were proposed to learn the high-dimensional complex features of HSI. Among them, CNNs have become the most widely used tools to extract spectral-spatial features from HSI. Furthermore, different CNNs, such as the variants from the 1D-CNN to the 3D-CNN, and from the single CNN to the hybrid CNN, have been proposed

This work has been supported by the National Natural Science Foundation of China (Grant No.61871226, 62001226), Jiangsu Provincial Social Developing Project (Grant No.BE2018727), the Fundamental Research Funds for the Central Universities (Grant No.30918011104, 30920021134), the National Major Research Plan of China (Grant No.2016YFF0103604). (Corresponding authors: Liang Xiao, and Jingxiang Yang.)

Q. Liu, L. Xiao, J. Yang and Z. Wei are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. L. Xiao is also with the Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing University of Science and Technology, Nanjing 210094, China. (e-mail: qc.l@njust.edu.cn; xiaoliang@mail.njust.edu.cn; yang123jx@njust.edu.cn; gswei@njust.edu.cn)

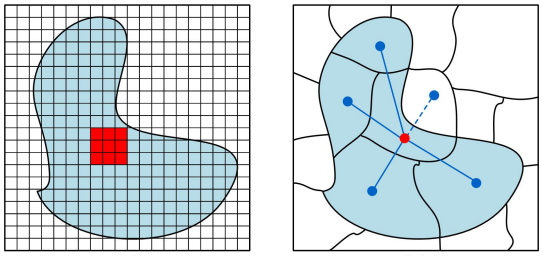

Fig. 1: Illustrations of the CNN and GCN convolution. (a) The CNN performs convolution on small-scale regular regions and generates pixel-level features, where the red square denotes the sampling grid. (b) The superpixel-based GCN performs convolution on large-scale irregular regions and generates superpixel-level features, where dots denote the centroids of superpixels, blue solid lines denote strong edge weights and the blue dashed line denotes a weak or zero edge weight.

to improve the learning ability of spectral-spatial features. For example, in [30], the authors presented a 3D-CNN with 3D convolution for learning spectral-spatial features. While in [26], the two-channel CNN (TCCNN) was proposed to integrate 1D and 2D convolution branches to jointly extract spectral and spatial features. Furthermore, the multichannel CNN (MCCNN) [31] that added an extra 3D convolution branch based on TCCNN has also been proposed. However, as their experiments showed, the gain from adding additional branches was limited, which could be caused by the unsatisfactory compatibility between different network branches. To alleviate this issue, Hao *et al.* [32] proposed to learn the fusion weights of different classes adaptively to balance the features extracted by different branches. Besides, the residual and dense connections have also been introduced to HSI classification [33], [34]. By enhancing the gradient flow, the networks become easier to train, and thus higher performance can be achieved. Moreover, some prior knowledge from other tasks, such as super-resolution [35] and denoising [36], has also been exploited to assist in HSI classification. Other recently advanced CNN architectures have also been explored, such as the spectral and spatial attention networks [37], [38]. Another extension of CNN-based HSI classification is to dynamically adjust the shapes or sampling locations of convolutional kernels according to HSI content [39], [40]; thus, the edges and small regions in classification maps can be better preserved. However, due to the high computational complexity, such deformable CNNs require a higher computing capacity and a longer training time.

Since the previous deep learning models are designed for Euclidean data, they often ignore the inherent correlation between adjacent land covers. Recently, owing to the ability in performing convolutions on arbitrarily structured graphs, the graph convolutional networks (GCNs) have received increasing attention. By encoding HSI into a graph, the correlation between adjacent land covers can be explicitly utilized, and the spatial contextual structure of HSI can be better modeled by GCNs. For instance, Qin *et al.* [41] proposed a semi-supervised GCN method, in which the HSI was first encoded into a graph and then processed by a GCN to propagate information between adjacent pixels according to their spectral similarity and spatial distance. However, due to the large number of pixels in HSI, taking each pixel as a node of a graph was shown to lead to a massive amount of computation and limited its applicability. To cope with this issue, Wan *et al.* [42] proposed to use the superpixels instead of pixels as the nodes, which greatly reduced the size of the graph and made the GCN more practical. Since the shapes and sizes of superpixels can be adaptively adjusted according to HSI content, superpixels could thus provide a good description (e.g., shapes and sizes) of the land covers and facilitate the subsequent graph learning. However, at the superpixel level, the pixels in a superpixel (node) are described with the same features. Hence, local spectral-spatial information at the individual pixel may not be captured by the graph nodes. Additionally, the initial node features obtained by aggregating the raw pixels in a superpixel may be affected by irrelevant information, and as a result, the edge weights calculated based on these rough nodes may not be sufficiently accurate. In this work, we are devoted to resolving the twofold problems mentioned above: 1) modeling the various spatial structures of land covers on graphs and generating pixel-level subtle features and 2) learning better node features and edge weights from training data.

As shown in Fig. 1, the superpixel-based GCN can model the various spatial structures of land covers on graphs, but it cannot generate subtle individual features for each pixel. In contrast, the CNN can learn local spectral-spatial features at the pixel level, but its receptive field is usually limited to a small square window, and thus, the large-scale contextual structure of HSI may be hard to be captured. Inspired by the pixel-level and superpixel-level feature fusion framework [43], we propose to integrate the CNN and GCN into a single network, in which the CNN branch learns features at the pixel level on the small regular regions, while the GCN branch models the large irregular regions of HSI and generates its superpixel-level features. Due to the structural incompatibility of the data between a CNN and GCN, i.e., data in a CNN are arranged in a standard three-dimensional grid, while data in a GCN are represented in the non-Euclidian domain defined by nodes and edges, we propose to remove this obstacle with the graph encoder and decoder to propagate features from image pixels to graph nodes and transform graph features back to the image space, respectively. To this end, we propose the CNN-enhanced GCN (CEGCN) to integrate the complementary advantages of CNN and GCN, as shown in Fig. 2.

Specifically, the input HSI is first transformed by a spectral transformation sub-network (STsN), which reduces the irrelevant information of spectra and enhances their discrimination. Then, a superpixel-level graph sub-network (SGsN) is constructed to model the large-scale spatial structure of HSI on graphs and generate its superpixel-level features. Furthermore, a pixel-level convolutional sub-network (PCsN) is used to extract local spectral-spatial features at the pixel level. Finally, the features extracted by the SGsN and PCsN are fused to boost the classification performance. In contrast to other GCN-based methods that encode HSI into a graph in the prepro-

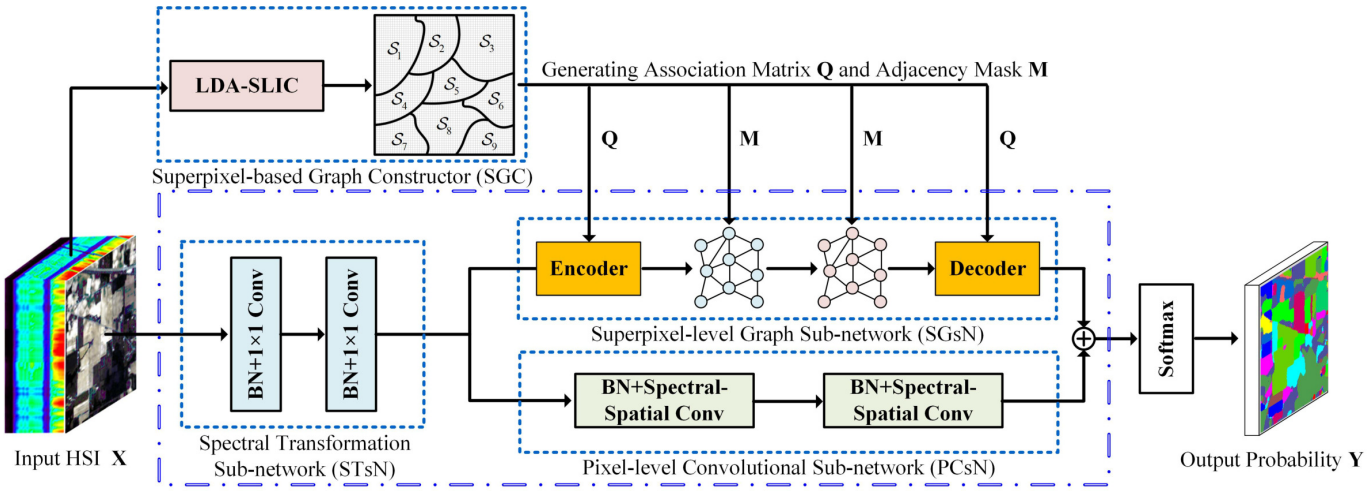

Fig. 2: The architecture of the proposed CEGCN.

cessing stage, in our method, the graph encoding process is integrated into the network so that better discriminative initial nodes are available for the subsequent SGsN. Additionally, different from using a fixed graph calculated from the raw HSI, the edge weights of the graph in our method are automatically learned by the SGsN itself, which can make the graph more adaptive to HSI content. Moreover, to improve the feature learning ability of the CNN, we also propose to replace the standard convolution in PCsN with the spectral-spatial convolution proposed in our previous work [44], in which each 3D convolutional kernel is separated into a 1D kernel used to extract spectral features and a 2D kernel used to exploit spatial features. Such a CEGCN enables SGsN and PCsN to work jointly, such that high performance in classification accuracy and inference speed is achieved.

The main contributions of this paper are as follows.

- We propose a novel hybrid deep learning framework by integrating a CNN with a GCN for pixel- and superpixel-level feature fusion. The graph encoder and decoder are presented to propagate features between image pixels and graph nodes, which alleviates the structural incompatibility of the data between the CNN and GCN and thus enables them to work jointly in a single network.
- Instead of encoding the raw HSI into graph in the preprocessing stage, we propose to first transform spectra by the network and then encode HSI, which is beneficial for the subsequent GCN to learn discriminative features on graphs. Moreover, instead of using the hand-crafted edge weights computed from the raw HSI, we propose to learn the edge weights from the training data, which can promote the adaptivity of graph learning to different HSIs.
- By combining the advantages of the CNN and GCN, the proposed CEGCN can learn features on small-scale regular regions and large-scale irregular regions simultaneously, which forms a better spectral-spatial structured representation for HSI. Furthermore, the lightweight architecture ensures that the CEGCN also achieves a higher training and inference speed.

The rest of this paper is organized as follows. The proposed CEGCN is introduced in Section II. The experiments are illustrated in Section III. The discussions are exhibited in Section IV. The conclusion is presented in Section V.

II. PROPOSED METHOD

The HSI classification task is designed to classify every pixel with a certain label. We denote HSI as $\mathbf{X} \in \mathbb{R}^{H \times W \times B}$, and $\mathcal{D} = \{(\mathbf{X}(\mathbf{p}_t), \mathbf{L}(\mathbf{p}_t))\}_{t=1}^{N}$ is its supervised data set, where $\mathbf{L}(\mathbf{p}_t)$ denotes the label of the pixel $\mathbf{X}(\mathbf{p}_t)$ at the spatial coordinate $\mathbf{p}_t = (x, y)$ in the HSI, and H, W, B, C and N denote the height, width, number of bands, number of classes and number of supervised pixels, respectively. The aim is to predict the labels of all the pixels in the HSI by a classification model given the supervised data set. In this work, we propose the CEGCN, which consists of four modules: 1) the superpixel-based graph constructor (SGC) that defines the graph encoder and decoder, 2) the STsN that transforms spectra and reduces their dimensionality, 3) the SGsN that models the large-scale structure of HSI and generates its superpixel-level features, and 4) the PCsN that extracts local features from HSI at the pixel level. In the following sections, we will describe each module in detail.

A. Superpixel-based Graph Constructor

In general, due to the incompatibility of data representation structures between different network architectures, it is not straightforward to integrate a CNN and a GCN directly. A CNN is designed for data that are arranged in a standard rectangular grid, while a GCN works on arbitrarily structured graphs. Although we can treat image pixels as the nodes in a graph, it would produce a vast graph, and the computation would be intractable. To take full advantage of learning on arbitrary graphs, in image processing, the GCN usually works on superpixel-based nodes but not the pixel-based nodes. To this end, we propose a structure converter to allow the features to be propagated between the image and graph space, as shown in Fig. 3, and we name it the superpixel-based graph constructor.

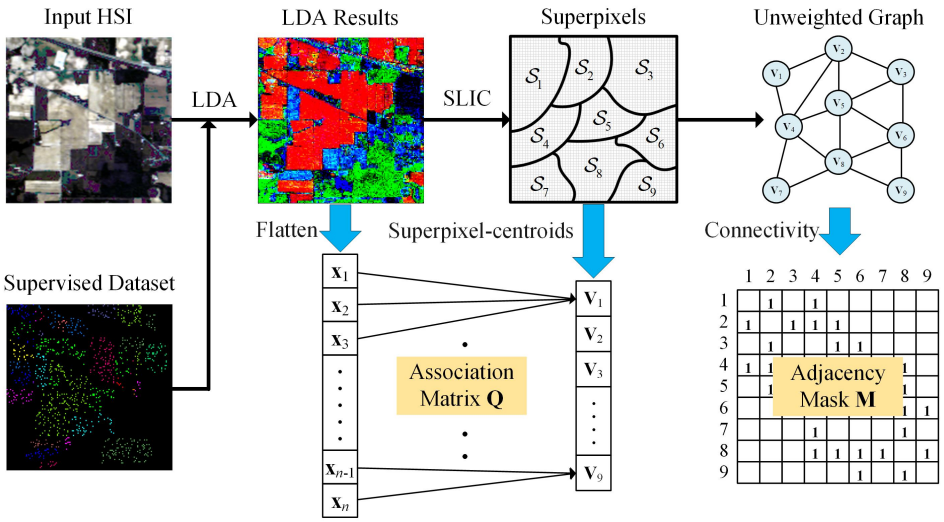

Fig. 3: The flowchart of the SGC module, where $\mathbf{x}_i$ denotes the ith pixel in the flattened HSI, and $\mathbf{V}_j$ denotes the centroid of the jth superpixel $\mathcal{S}_j$.

First, to quickly build a suitable graph structure containing spectral-spatial information, we apply linear discriminant analysis (LDA) to preprocess the HSI for dimensionality reduction. Subsequently, we use the simple linear iterative clustering (SLIC) [45] method to partition HSI into many spatially connected and spectrally similar superpixels, which reveals the spatial structure of HSI. Finally, by establishing the adjacency relations between superpixels, the HSI is converted to an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the nodes and edges, respectively. Specifically, we utilize the LDA and the SLIC to partition the HSI into $Z = \lceil (H \times W)/\lambda \rceil$ superpixels, where $\lambda (1 \leq \lambda)$ denotes the segmentation scale used to control the average size of superpixels. Let $\mathcal{S} = \{\mathcal{S}_i\}_{i=1}^{Z}$ denote the superpixel set, $\mathcal{S}_i = \{\mathbf{x}_j^i\}_{j=1}^{N_i}$ denotes the ith superpixel, $\mathbf{x}_j^i$ denotes the jth pixel in $\mathcal{S}_i$, and N_i is the number of pixels in $\mathcal{S}_i$ ($\mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \forall i \neq j$ and $H \times W = \sum_{i=1}^{Z} N_i$). Analogous to [42], we take the centroid (mean) of each superpixel as a node, i.e.,

$$\begin{aligned} \mathbf{V} &= [\mathbf{V}_1, \mathbf{V}_2, \dots \mathbf{V}_Z]^T \\ &= \left[\frac{1}{N_1} \sum_{j=1}^{N_1} \mathbf{x}_j^1, \frac{1}{N_2} \sum_{j=1}^{N_2} \mathbf{x}_j^2, \dots, \frac{1}{N_Z} \sum_{j=1}^{N_Z} \mathbf{x}_j^Z \right]^T \end{aligned} \tag{1}$$

where $\mathbf{V}_i$ denotes the feature vector of the ith node, and $\mathbf{V}$ is the matrix form of the nodes $\mathcal{V}$.

Because different superpixels contain different numbers of pixels, it is difficult to compute Eq. (1) straightforwardly by the network. To resolve this problem, inspired by [46], we construct a mapping association between pixels and superpixels. Specifically, let $\mathbf{Q} \in \mathbb{R}^{HW \times Z}$ be the association matrix between pixels and superpixels, then we have

$$\mathbf{Q}_{i,j} = \begin{cases} 1 & \text{if } \hat{\mathbf{X}}_i \in \mathcal{S}_j \\ 0 & \text{otherwise} \end{cases}, \hat{\mathbf{X}} = Flatten(\mathbf{X}) \tag{2}$$

where $Flatten(\cdot)$ denotes flattening the HSI data by the spatial dimension, $\hat{\mathbf{X}}_i$ denotes the ith pixel in $\hat{\mathbf{X}}$, $\mathcal{S}_j$ is the jth superpixel, and $\mathbf{Q}_{i,j}$ denotes the value of $\mathbf{Q}$ at location (i, j). Then, Eq. (1) can be rewritten in the form of matrix multiplication, i.e.,

$$\mathbf{V} = Encode(\mathbf{X}; \mathbf{Q}) = \hat{\mathbf{Q}}^{\mathrm{T}} Flatten(\mathbf{X}) \tag{3}$$

where $\hat{\mathbf{Q}}$ denotes the normalized $\mathbf{Q}$ by column, i.e., $\hat{\mathbf{Q}}_{i,j} = \mathbf{Q}_{i,j} / \sum_m \mathbf{Q}_{m,j}$, and $Encode(\cdot; \mathbf{Q})$ is the graph encoder used to encode HSI into the nodes of $\mathcal{G}$ according to $\mathbf{Q}$. By using the transformation in Eq. (3), HSI can be easily and quickly encoded into graph nodes and then processed with the GCN.

Because the task is to classify pixels instead of superpixel-based nodes, the features of nodes should be assigned to pixels before classification. To propagate the node features to pixels, with the association matrix $\mathbf{Q}$, we define the graph decoder $Decoder(\cdot; \mathbf{Q})$, which can be written as

$$\mathbf{X}^* = Decoder(\mathbf{V}; \mathbf{Q}) = Reshape\,(\mathbf{Q}\mathbf{V}) \tag{4}$$

where $Reshape(\cdot)$ denotes restoring the spatial dimension of the flattened data. After the decoding procedure, the graph features can be projected back to the image space.

Then, Eq. (3) will be inserted into the network to encode features in the image space into nodes (graph space). Different from other methods that calculate edge weights from the raw HSI, to promote the adaptability of the graph to HSI content, we propose to learn the edge weights by the GCN itself, which will be described in the subsequent section. In this module, we only construct an unweighted graph, in which the connectivity between nodes is encoded into the adjacency mask. Specifically, we define the adjacency mask $\mathbf{M} \in \mathbb{R}^{Z \times Z}$ as

$$\mathbf{M}_{i,j} = \begin{cases} 1 & \text{if } \mathcal{S}_i \text{ and } \mathcal{S}_j \text{ are adjancent} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\mathbf{M}_{i,j}$ is the value of $\mathbf{M}$ at location (i, j). Note that in our method, $\mathcal{S}_i$ and $\mathcal{S}_j$ are adjacent when they share a common boundary, as shown in Fig. 3.

In summary, based on the superpixel segmentation, we construct an unweighted graph for HSI. The proposed graph

encoder can project arbitrary features from the image space to the graph space. Correspondingly, the graph decoder can assign node features to pixels. Both the graph encoder and decoder can be inserted into the network to integrate the CNN and GCN together. Furthermore, the adjacency mask records the adjacency between land covers, and it will work with the GCN to learn the edge weights.

B. Spectral Transformation Sub-network

Original HSI contains noise and redundant information. To promote a robust and discriminative deep spectral feature learning, we propose the STsN to process the input HSI first. Specifically, let $\mathbf{X}^l$ be the output feature map of the lth spectral convolutional layer; then for each spatial location $\mathbf{p}_0 = (x, y)$ on the output feature map, we have

$$\mathbf{X}_j^l(\mathbf{p}_0) = f\left(\mathbf{W}_j^l \cdot \tilde{\mathbf{X}}^{l-1}(\mathbf{p}_0) + b_j^l\right) \tag{6}$$

where $\tilde{\mathbf{X}}^{l-1} = BN(\mathbf{X}^{l-1})$ denotes the batch-normalized input feature map of the lth convolutional layer, $\mathbf{W}_j^l$ and b_j^l denote the jth kernel with a size of 1×1 and the bias in the lth layer respectively, the operator $\cdot$ denotes the inner product, $\mathbf{X}_j^l(\mathbf{p}_0)$ is the value of the jth output feature channel at the spatial location $\mathbf{p}_0$, and $f(\cdot)$ denotes the activation function.

The STsN is used to reduce the redundant information of spectra and enhance their discrimination. Assuming that the STsN is composed of $L_T (1 \leq L_T)$ spectral convolutional layers, then its input and output data are $\mathbf{X}^0$ and $\mathbf{X}^{L_T}$, respectively. We record the STsN module as $\mathbf{X}^{L_T} = STsN(\mathbf{X}^0)$, which is then used in a later section.

C. Superpixel-level Graph Sub-network

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by the nodes $\mathcal{V}$ and edges $\mathcal{E}$. In practice, the $\mathcal{V}$ and $\mathcal{E}$ are generally encoded into a node matrix $\mathbf{H}$ and adjacency matrix $\mathbf{A}$. Specifically, the ith row of $\mathbf{H}$ denotes the ith node, and $\mathbf{A}_{i,j}$ denotes the edge weight between the ith node and the jth node. Shuman *et al.* [47] extended conventional convolution to the topological domain by performing the spectral decomposition on the Laplacian matrix of the graph. The Laplacian matrix is calculated by $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ is the degree matrix of $\mathbf{A}$, i.e., $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$. Then, the convolution operation on the graph is defined as

$$\mathbf{y} = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^{\mathrm{T}} \mathbf{x} \tag{7}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors defined on the graph $\mathcal{G}$, $\mathbf{U}$ is the orthogonal matrix composed of the feature vectors of the Laplacian matrix $\mathbf{L}$ by column, and $g_\theta(\mathbf{\Lambda}) = diag(\theta_1, \theta_2, \ldots, \theta_n)$ denotes the parameters to be learned. Note that the activation function here is not considered. The graph convolution defined in Eq. (7) contains a large number of parameters to be learned, which could lead to overfitting under limited samples. To overcome this shortcoming, Defferrard *et al.* [48] improved the graph convolution by letting $\theta_i = \sum_{j=0}^{K} \alpha_j \lambda_i^j$, where λ_i^j denotes the j power of the ith eigenvalue of $\mathbf{L}$. Then, Eq. (7) becomes

$$\mathbf{y} = \sum_{j=0}^{K} \alpha_j \mathbf{L}^j \mathbf{x} \tag{8}$$

where α_j is the convolution parameter to be learned, K defines the size of the receptive field, and $\mathbf{L}^j$ denotes the j power of the Laplacian matrix. In most cases, $K = 1$ is sufficient. Then, we have

$$\mathbf{y} = \alpha_0 \mathbf{x} + \alpha_1 \mathbf{L}\mathbf{x} = \beta\left(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{x} \tag{9}$$

by letting $K = 1$ and $\beta = 0.5\alpha_0 = -\alpha_1$. Since the eigenvalues of $\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ are in the range $[0, 2]$, numerical instability may occur when repeating this operation multiple times. To cope with this problem, Kipf *et al.* [49] improved this graph convolution by adding an additional edge from each node to itself, i.e., adding the identity matrix $\mathbf{I}$ to the adjacency matrix $\mathbf{A}$. Moreover, to improve the adaptivity of the graph to HSI content, we propose to calculate the edge weights from the input features of each layer. Specifically, let $\mathbf{H}^l$ be the output of the lth graph convolutional layer; then, the adjacency matrix in this layer is calculated by

$$\mathbf{A}^l = Sigmoid\left(\left(\tilde{\mathbf{H}}^{l-1}\mathbf{W}_\phi^l\right)\left(\tilde{\mathbf{H}}^{l-1}\mathbf{W}_\phi^l\right)^{\mathrm{T}}\right) \odot \mathbf{M} + \mathbf{I} \tag{10}$$

where $\tilde{\mathbf{H}}^{l-1} = BN(\mathbf{H}^{l-1})$ denotes the normalized input feature map of the lth graph convolutional layer, $\mathbf{W}_\phi^l$ is the parameter to be learned, $\mathbf{M}$ is the adjacency mask calculated by Eq. (5), the operator $\odot$ denotes Hadamard product, and $\mathbf{A}^l$ is the learned adjacency matrix in this layer. The sigmoid function compresses the edge weights in the range $(0, 1)$, which can prevent numerical instability. The adjacency mask can cut off the association of non-adjacent nodes while keeping the connectivity between adjacent nodes. As a result, the final graph convolution is

$$\mathbf{H}^l = f\left(\left(\mathbf{D}^l\right)^{-\frac{1}{2}}\mathbf{A}^l\left(\mathbf{D}^l\right)^{-\frac{1}{2}}\tilde{\mathbf{H}}^{l-1}\mathbf{W}^l\right) \tag{11}$$

where $\mathbf{D}^l$ is the degree matrix of $\mathbf{A}^l$, and $\mathbf{W}^l$ is the parameter to be learned.

The SGsN is used to model the large-scale spatial structures of land covers on graphs and generate superpixel-level features. Supposing that the SGsN is composed of $L_S\,(1 \leq L_S)$ graph convolutional layers, then its input and output feature maps are $\mathbf{H}^0$ and $\mathbf{H}^{L_S}$, respectively. We record the SGsN module as a function, i.e., $\mathbf{H}^{L_S} = SGsN(\mathbf{H}^0; \mathbf{M})$.

D. Pixel-level Convolutional Sub-network

Owing to its ability in extracting local spectral-spatial features from HSI, the 2D CNN architecture has been widely used in HSI classification. However, due to the numerous parameters in its convolutional kernels and the limited training samples in the task, the 2D CNN can easily overfit the training data. Therefore, in this work, we utilize the spectral-spatial convolution proposed in our previous work [44] to construct the PCsN. Specifically, in the spectral-spatial convolution, each 3D kernel is separated into a 1D kernel used to extract spectral features and a 2D kernel used to extract spatial features, which can greatly reduce the parameters and enhance the robustness

to overfitting. Let $\mathbf{T}^l$ be the output feature map of the lth spectral-spatial convolutional layer in the PCsN; then, we have

$$\mathbf{T}_j^l(\mathbf{p}_0) = f\left(\sum_{\mathbf{p}_n\in\mathcal{R}} \mathbf{k}_j^l(\mathbf{p}_n)\cdot f\left(\mathbf{W}_j^l\cdot\tilde{\mathbf{T}}^{l-1}(\mathbf{p}_0+\mathbf{p}_n)\right)+b_j^l\right) \quad (12)$$

where $\tilde{\mathbf{T}}^{l-1}=BN(\mathbf{T}^{l-1})$ is the normalized input data of the lth layer, $\mathbf{W}_j^l$, $\mathbf{k}_j^l$, and b_j^l denote the jth spectral 1D kernel, spatial 2D kernel, and bias respectively, $\mathcal{R}$ defines a standard sampling grid (e.g., $\mathcal{R}=(-1,-1),(-1,0),\ldots,(0,1),(1,1)$ defines a 3×3 sampling grid), and $\mathbf{p}_n$ enumerates the locations in $\mathcal{R}$. Owing to the separation of the 3D kernel, the number of parameters necessary to output each feature channel can be reduced from $d^2\times n$ to d^2+n, where d^2 and n denote the spatial size of the kernel and the number of input channels, respectively.

Assuming that the PCsN is composed of $L_P\,(1\le L_P)$ spectral-spatial convolutional layers, then its initial input and final output are $\mathbf{T}^0$ and $\mathbf{T}^{L_P}$, respectively. Similar to the other sub-networks in CEGCN, the PCsN is also recorded as a function, i.e., $\mathbf{T}^{L_P}=PCsN(\mathbf{T}^0)$.

E. Feature Fusion and Classification

Three sub-networks are defined in our method: STsN, SGsN, and PCsN. Each sub-network has its role. Specifically, the STsN is used to reduce the redundant information of spectra and enhance their discrimination, which provides better initial features for the following SGsN and PCsN. The SGsN is used to learn the representation of HSI on graphs and generate its superpixel-level features. Furthermore, the PCsN is used to extract local spectral-spatial features from HSI at the pixel level. Finally, these two different levels of features are fused and then classified. The whole feature extraction process can be expressed by

$$\begin{aligned}\mathbf{F} = Decode\left(SGsN\left(Encode\left(\mathbf{X}_{\mathrm{E}};\mathbf{Q}\right);\mathbf{M}\right);\mathbf{Q}\right)\\ \oplus PCsN\left(\mathbf{X}_{\mathrm{E}}\right)\end{aligned} \quad (13)$$

where $\mathbf{X}_{\mathrm{E}}=STsN\left(\mathbf{X}\right)$ denotes the enhanced HSI data using STsN, the operator $\oplus$ denotes concatenating features along the spectral dimension, and $\mathbf{F}$ is the final fused feature map.

To generate the probability of each class at each pixel, we utilize a softmax classifier to classify the feature map $\mathbf{F}$. Let $\mathbf{Y}$ be the output probability map; then, for each location $\mathbf{p}_0$ on $\mathbf{Y}$, we have

$$\mathbf{Y}(\mathbf{p}_0)=\frac{1}{\sum_m^C e^{\mathbf{k}_m\cdot\mathbf{F}(\mathbf{p}_0)+b_m}}\begin{bmatrix} e^{\mathbf{k}_1\cdot\mathbf{F}(\mathbf{p}_0)+b_1}\\ e^{\mathbf{k}_2\cdot\mathbf{F}(\mathbf{p}_0)+b_2}\\ \vdots\\ e^{\mathbf{k}_C\cdot\mathbf{F}(\mathbf{p}_0)+b_C}\end{bmatrix} \quad (14)$$

where $\mathbf{k}_m$ and $b_m(1\le m\le C)$ denote the mth 1D kernel and bias, and $\mathbf{Y}(\mathbf{p}_0)$ denotes the class probability vector of the pixel at location $\mathbf{p}_0$. Finally, to train the network, we use the cross-entropy loss function as

$$\mathcal{L}=-\sum_{t=1}^{N}\sum_{c=1}^{C}\mathbf{L}_c(\mathbf{p}_t)\log\left(\mathbf{Y}_c\left(\mathbf{p}_t\right)\right) \quad (15)$$

where $\mathbf{L}_c(\mathbf{p}_t)$ denotes the cth element of the label $\mathbf{L}(\mathbf{p}_t)$, and $\mathbf{Y}_c\left(\mathbf{p}_t\right)$ denotes the probability of the pixel $\mathbf{X}(\mathbf{p}_t)$ belonging to the cth class.

III. EXPERIMENTS

In this section, three public HSI data sets are utilized to estimate the proposed CEGCN: they are the Indian Pines, the University of Pavia, and the Salinas data sets. Seven state-of-the-art deep learning methods for HSI classification are compared with our method: they are the deep CNN (DCNN) [25], the two-channel CNN (TCCNN) [26], the multichannel CNN (MCCNN) [31], the spectral-spatial residual network (SSRN) [33], the fast dense spectral-spatial convolutional network (FDSSC) [34], the hybrid spectral network (HybridSN) [27], and the double-branch dual-attention network (DBDA) [38]. All the baseline methods have adopted the hyperparameters recommended in their original papers. The overall accuracy (OA), average accuracy (AA), and kappa statistics (KPP) are employed as the evaluation indices of the classification performance.

A. Configuration for Hyperparameters

The CEGCN is composed of four modules: SGC, STsN, SGsN, and PCsN. Specifically, in the SGC, the segmentation scale λ is fixed to 100. The numbers of layers in the STsN, SGsN, and PCsN are all set to 2, i.e., $L_T=L_S=L_P=2$. The size of $\mathbf{W}_\phi$ in Eq. (10) is fixed to 128×256. The special size of convolutional kernels in the PCsN is fixed to 5×5. All the activation functions in CEGCN are the leaky rectified linear units (Leaky ReLUs) [50]. Other detailed configurations (e.g., numbers of output channels of each layer) are summarized in Table I. Finally, we use the Adam optimizer [51] to train our network with a learning rate equal to 0.0005. The number of training iterations is 600. Our code[1] is completed with Python-3.6 and PyTorch-1.1.0. The experimental environment consists of an i7-8700K CPU and a GTX-1080Ti GPU.

TABLE I: Architecture configuration of the CEGCN.

Module	Layer No.	Size and Number of Kernels
STsN	1	1×1@128
	2	1×1@128
PCsN	1	5×5@128
	2	5×5@64
SGsN	1	@128
	2	@64
SGC	$\lambda=100$	

B. Data set

The first data set is the Indian Pines data set captured over the agricultural test site in the Indian Pines. This HSI contains 220 bands with a size of 145×145. After removing the disturbing bands (e.g., low signal-to-noise ratio and water vapor absorption bands), the remaining 200 bands with 10,366 samples of 16 terrain classes were selected for research and

[1]Code is available at: https://github.com/qichaoliu/CNN_Enhanced_GCN

TABLE II: Category information of the Indiana Pines data set.

No.	Class	Train.	Val.	Test.
1	Alfalfa	5	1	48
2	Corn-notill	143	14	1277
3	Corn-mintill	83	8	743
4	Corn	23	2	209
5	Pasture	49	4	444
6	Trees/Grass	74	7	666
7	Pasture-mowed	2	1	23
8	Hay-windrowed	48	4	437
9	Oats	2	1	17
10	Soybeans-notill	96	9	863
11	Soybean-mintill	246	24	2198
12	Soybean-cleantill	61	6	547
13	Wheat	21	2	189
14	Woods	129	12	1153
15	Building-Grass	38	3	339
16	Stone-steelTowers	9	1	85
	Total	1029	99	9238

TABLE III: Category information of the University of Pavia data set.

No.	Class	Train.	Val.	Test.
1	Asphalt	67	67	6497
2	Meadows	187	187	18275
3	Gravel	21	21	2057
4	Trees/Grass	31	31	3002
5	Metalsheets	14	14	1317
6	Baresoil	51	51	4927
7	Bitumen	14	14	1302
8	Bricks	37	37	3608
9	Shadows	10	10	927
	Total	432	432	41912

analysis. Moreover, 10%, 1%, and 89% of samples per class are randomly selected for training, validation, and testing, respectively. The detailed category information of this HSI is shown in Table II.

The second data set is the University of Pavia data set captured over the University of Pavia. This HSI contains 115 bands with a size of 610×340. After removing the disturbing bands, the remaining 103 bands were used for research. A total of 42,776 samples of 9 terrain classes were labeled in it. Consequently, 1%, 1%, and 98% of samples per class are randomly selected for training, validation, and testing, respectively. The detailed category information of this HSI is shown in Table III.

The third data set is the Salinas data set that was captured over the Salinas Valley. This HSI contains 224 bands with a size of 512×217. After removing the disturbing bands, the remaining 204 bands with 54,129 samples of 16 crops were used for the experiment. Then, 1%, 1%, and 98% of samples per class are randomly selected as the training, validation, and testing data sets, respectively. The detailed category information of this HSI is shown in Table IV.

C. Comparison of Classification Performance

In this section, three data sets are employed as the evaluation data sets, as shown in Table II–Table IV. Each experiment is repeated ten times, in which the mean and the standard deviation of each index (i.e., OA, AA, and KPP) are reported. The classification accuracies of different methods on each data set are detailed in Table V–Table VII, and the classification maps obtained by these methods are illustrated in Figs. 4-6.

TABLE IV: Category information of the Salinas data set.

No	Class	Train.	Val.	Test.
1	Brocoli_green_weeds_1	21	21	1967
2	Brocoli_green_weeds_2	38	38	3650
3	Fallow	20	20	1936
4	Fallow_rough_plow	14	14	1366
5	Fallow_smooth	27	27	2624
6	Stubble	40	40	3879
7	Celery	36	36	3507
8	Grapes_untrained	113	113	11045
9	Soil_vinyard_develop	63	63	6077
10	Corn_senesced_green_weeds	33	33	3212
11	Lettuce_romaine_4wk	11	11	1046
12	Lettuce_romaine_5wk	20	20	1887
13	Lettuce_romaine_6wk	10	10	896
14	Lettuce_romaine_7wk	11	11	1048
15	Vinyard_untrained	73	73	7122
16	Vinyard_vertical_trellis	19	19	1769
	Total	549	549	53031

In the experiment on the Indian Pines data set, our method achieves the best classification results of the evaluated methods. The TCCNN and MCCNN boosted the performance by adding extra CNN branches compared with the DCNN. Other compared methods improved the performance by using advanced network architectures, such as the residual connection, dense connection, and attention mechanism. However, without employing any of these advanced techniques, the CEGCN still achieves the best performance, which is due to the complementarity between the PCsN and SGsN. Specifically, the SGsN learns features on large-scale irregular image regions and encourages piecewise smoothed classification results, which can reduce the classification error caused by abnormal noise. Additionally, the PCsN performs convolution on small-scale regular image regions and generates pixel-level features to supplement the SGsN, which can reduce the classification error in small regions caused by the superpixel segmentation. As a result, the classification map shows a high consistency with the ground truth.

For the University of Pavia and Salinas data sets, our method surpasses all the compared methods. Notably, the accuracies obtained by the DCNN, TCCNN, and MCCNN decrease in turn on both of these data sets, which is contrary to the results from the Indian Pines data set. This phenomenon may have been caused by the fewer training samples in these two data sets compared with Indian Pines, which indicates that more network branches would achieve higher performance with sufficient training samples, but if training samples are insufficient, too many branches may result in performance degradation. By contrast, the fusion of the CNN and GCN in the proposed CEGCN shows more stable results, which shows its robustness and effectiveness.

In summary, the proposed CEGCN encourages smoothed classification results in homogeneous regions, while preserving details in small regions. Compared with the combination of different CNN patterns (e.g., 1D, 2D, and 3D), the complementarity between the PCsN and SGsN is more stable and robust.

TABLE V: Individual class, OA, AA and KPP of all methods on the Indian Pines data set.

Class	DCNN	TCCNN	MCCNN	SSRN	FDSSC	HybridSN	DBDA	CEGCN
1	98.12±3.35	95.65±8.69	98.18±3.63	99.42±1.14	**100±0**	97.16±2.73	**100±0**	97.50±2.04
2	94.72±1.69	95.84±0.92	96.25±0.80	98.40±0.96	98.29±0.62	97.15±0.56	97.79±1.05	**99.08±0.57**
3	94.33±2.30	97.31±1.07	96.84±1.02	97.13±2.29	98.19±0.78	98.25±0.65	98.56±1.08	**99.27±0.41**
4	94.32±4.47	96.85±1.72	97.16±2.46	96.71±4.20	**98.30±1.12**	97.90±2.57	94.89±1.49	97.98±1.46
5	98.08±0.92	97.88±1.13	99.03±0.17	97.20±1.30	98.06±0.17	98.49±0.96	97.54±1.44	**99.14±0.88**
6	98.79±0.74	98.51±1.35	98.61±1.29	99.01±0.89	98.98±0.57	98.92±0.38	98.39±1.57	**99.63±0.32**
7	**100±0**	98.82±2.35	98.03±2.41	80±40	**100±0**	**100±0**	91.81±16.36	99.13±1.73
8	97.75±1.64	98.79±1.54	99.45±0.68	99.45±0.61	**99.95±0.09**	99.67±0.31	99.64±0.71	99.90±0.11
9	90±30	92.51±1.03	97.64±4.70	0±0	60±48.98	92.38±5.25	**98.75±2.50**	97.77±4.44
10	95.53±2.45	95.43±1.92	95.21±1.72	96.38±3.06	98.72±1.08	**98.74±0.81**	96.38±2.01	97.03±1.21
11	96.66±1.18	97.51±0.47	96.68±0.92	98.02±0.90	99.34±0.36	99.16±0.26	98.87±0.32	**99.49±0.17**
12	94.87±3.02	95.12±1.31	96.04±1.43	95.42±4.10	98.19±0.88	97.47±1.17	98.67±0.95	**99.26±0.25**
13	99.37±0.69	98.95±0.32	99.58±0.51	98.95±1.27	**100±0**	98.02±1.64	99.37±0.76	97.43±1.08
14	98.36±0.90	99.01±0.77	99.30±0.36	99.48±0.28	99.75±0.28	99.32±0.44	99.66±0.27	**100±0**
15	93.94±1.85	94.83±2.38	95.71±2.18	96.76±1.89	98.88±0.75	97.64±1.58	97.82±1.74	**99.22±0.78**
16	92.43±5.34	96.16±2.55	95.48±2.20	97.05±2.63	**98.37±2.12**	91.02±4.07	89.54±4.91	96.68±3.21
OA(%)	96.31±0.37	97.14±0.31	97.19±0.45	97.86±0.41	98.93±0.27	98.44±0.16	98.24±0.35	**99.12±0.19**
AA(%)	96.08±2.10	96.82±1.05	97.45±0.53	90.59±2.34	96.56±2.96	97.58±0.82	97.35±1.12	**98.66±0.37**
KPP(×100)	95.80±0.43	96.74±0.36	96.80±0.52	97.56±0.47	98.78±0.31	98.23±0.18	97.99±0.40	**99.01±0.22**

TABLE VI: Individual class, OA, AA and KPP of all methods on the University of Pavia data set.

Class	DCNN	TCCNN	MCCNN	SSRN	FDSSC	HybridSN	DBDA	CEGCN
1	92.72+1.02	89.65±3.13	88.65±2.10	99.66±0.24	99.22±0.67	95.13±1.81	97.48±1.92	**99.91±0.10**
2	97.14+0.59	97.24±0.86	97.37±0.78	98.70±1.02	99.62±0.27	99.16±0.49	99.36±0.61	**100±0**
3	87.91+3.40	74.12±4.36	73.78±5.75	93.95±5.15	97.54±1.96	88.73±4.90	**99.19±1.23**	97.82±1.71
4	99.35+0.90	97.36±1.34	98.54±0.42	99.72±0.27	**99.89±0.09**	98.18±0.77	98.12±0.29	95.57±1.45
5	98.92+1.61	98.94±1.23	98.91±0.91	99.93±0.08	99.29±1.22	98.98±0.93	99.48±0.51	**99.98±0.03**
6	97.41+0.77	95.29±0.83	93.85±2.24	98.52±2.11	99.56±0.48	98.66±0.96	98.86±0.90	**99.99±0.01**
7	91.99+4.80	83.97±3.46	84.47±2.91	96.84±2.22	98.52±2.25	96.64±2.37	97.18±5.46	**99.87±0.10**
8	88.41+1.07	81.84±2.49	82.04±1.68	88.85±5.74	94.29±2.27	90.69±2.72	88.86±5.60	**99.41±0.42**
9	99.41+0.59	93.48±2.21	96.05±5.62	99.53±0.55	**99.72±0.23**	97.21±1.86	97.56±2.88	99.52±0.30
OA(%)	95.35±0.17	93.01±0.73	92.89±1.01	97.54±0.58	98.94±0.21	97.01±0.69	97.74±0.79	**99.49±0.14**
AA(%)	94.81±0.56	90.21±0.73	90.41±1.26	97.30±0.55	98.63±0.47	95.93±0.87	97.34±0.93	**99.12±0.26**
KPP(×100)	93.81±0.23	90.68±0.99	90.55±1.34	96.74±0.77	98.60±0.28	96.02±0.92	97.01±1.05	**99.33±0.19**

TABLE VII: Individual class, OA, AA and KPP of all methods on the Salinas data set.

Class	DCNN	TCCNN	MCCNN	SSRN	FDSSC	HybridSN	DBDA	CEGCN
1	99.97±0.04	99.97±0.04	99.14±1.61	**100±0**	**100±0**	99.79±0.24	99.93±0.12	**100±0**
2	99.68±0.57	98.85±1.04	99.59±0.50	**100±0**	**100±0**	99.97±0.02	99.16±0.91	**100±0**
3	96.23±1.19	95.95±2.87	95.10±3.47	98.54±1.07	99.70±0.37	99.96±0.04	98.31±1.01	**100±0**
4	97.48±2.22	93.07±5.69	96.10±2.41	98.39±2.24	99.19±1.21	98.35±1.05	96.49±2.57	**99.86±0.19**
5	99.48±0.22	98.51±0.68	98.88±0.40	99.74±0.12	99.87±0.12	**99.93±0.07**	99.56±0.32	98.65±0.84
6	99.49±0.73	98.06±1.91	99.03±1.19	99.98±0.01	**100±0**	99.93±0.10	**100±0**	**100±0**
7	98.96±0.94	98.76±1.18	98.49±1.53	99.98±0.02	**100±0**	**100±0**	99.96±0.06	99.97±0.02
8	89.55±0.83	88.52±2.16	84.81±1.82	89.21±5.45	97.62±1.99	**98.81±0.88**	93.58±6.44	98.50±2.24
9	99.66±0.10	99.22±0.34	99.17±0.41	99.78±0.16	99.70±0.15	99.96±0.02	99.37±0.66	**100±0**
10	91.96±4.35	93.17±2.85	92.86±4.49	98.47±0.66	**99.59±0.24**	98.96±1.06	99.40±0.33	97.99±1.70
11	93.93±1.19	91.45±4.78	91.42±5.59	97.95±1.86	98.01±1.84	99.21±1.05	96.25±2.43	**99.88±0.14**
12	98.32±0.60	98.46±1.13	98.11±0.93	98.71±1.55	99.48±0.69	99.81±0.33	99.82±0.35	**100±0**
13	98.74±1.42	99.41±1.17	98.71±2.19	99.12±1.53	**99.86±0.26**	98.77±2.28	99.86±0.16	99.77±0.25
14	99.03±0.56	98.18±0.36	98.55±1.14	**99.65±0.47**	99.58±0.47	99.60±0.45	98.19±1.59	98.87±1.32
15	85.28±1.33	86.81±1.13	79.19±3.03	92.01±5.82	91.03±3.80	97.88±2.67	93.17±6.17	**99.54±0.22**
16	99.56±0.25	99.71±0.19	98.78±1.08	99.93±0.13	**100±0**	**100±0**	99.89±0.20	98.91±1.27
OA(%)	94.72±0.63	94.31±0.75	92.59±0.59	96.03±0.69	98.03±0.52	99.27±0.29	97.01±1.31	**99.36±0.41**
AA(%)	96.71±0.58	96.13±0.53	95.50±0.60	98.22±0.30	98.98±0.18	99.43±0.19	98.31±0.39	**99.49±0.11**
KPP(×100)	94.12±0.70	93.66±0.83	91.75±0.66	95.58±0.77	97.80±0.58	99.19±0.32	96.67±1.46	**99.29±0.46**

IV. DISCUSSIONS

In this section, we design and conduct various experiments to analyze the effectiveness of the proposed CEGCN from different perspectives.

A. Effectiveness of Different Modules in CEGCN

Each module in our method has its role. To evaluate the effectiveness of different modules, in this section, we conduct a set of ablation experiments in which different modules are combined and tested. Due to the necessity of the SGC to the SGsN, we combine them by default. The classification accuracies with the combination of different modules are shown in Table VIII.

It is clear from the results that for the first two data sets, the combination of the SGsN and PCsN achieves a higher performance than when they are used independently, which indicates the complementarity between them. For the Salinas data set, the result of the combination of the SGsN and PCsN is lower than that acquired by using SGsN alone; but after adding

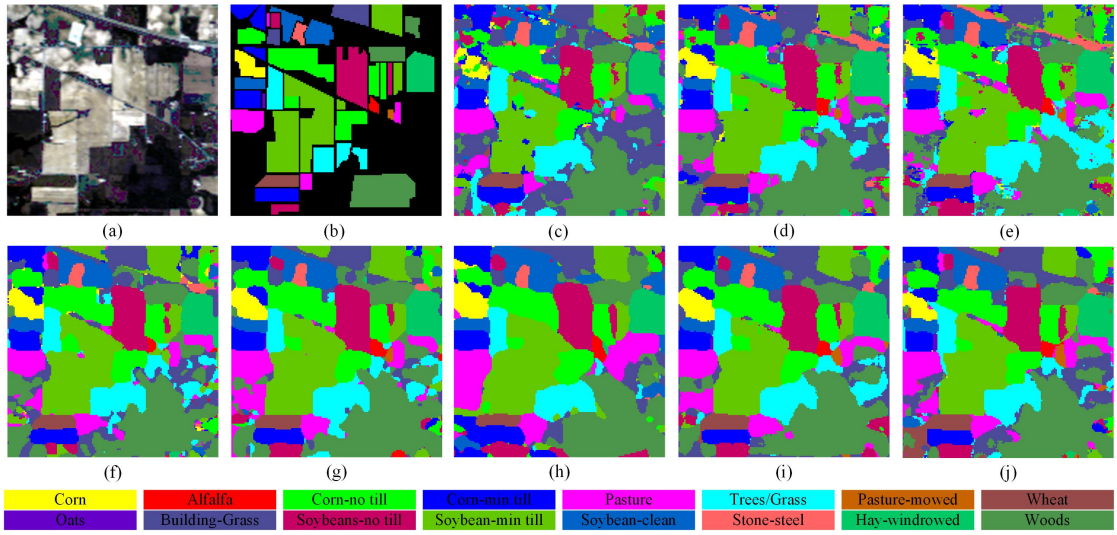

Fig. 4: Classification maps of different methods for the Indian Pines data set. (a) False-color image. (b) Ground truth. (c) DCNN (OA = 96.31%). (d) TC-CNN (OA = 97.14%). (e) MCCNN (OA = 97.19%). (f) SSRN (OA = 97.86%). (g) FDSSC (OA = 98.93%). (h) HybridSN (OA = 98.44%). (i) DBDA (OA = 98.24%). (j) CEGCN (OA = 99.12%).

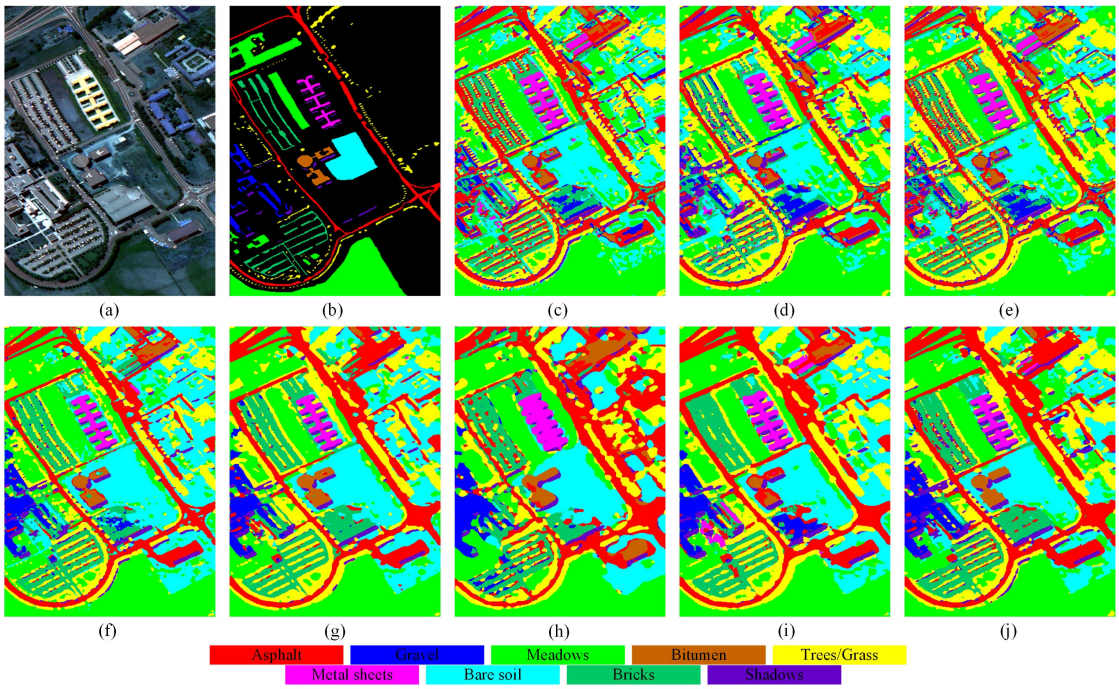

Fig. 5: Classification maps of different methods for the University of Pavia data set. (a) False-color image. (b) Ground truth. (c) DCNN (OA = 95.35%). (d) TC-CNN (OA = 93.01%). (e) MCCNN (OA = 92.89%). (f) SSRN (OA = 97.54%). (g) FDSSC (OA = 98.94%). (h) HybridSN (OA = 97.01%). (i) DBDA (OA = 97.74%). (j) CEGCN (OA = 99.49%).

TABLE VIII: OA(%) indices of the combinations of different modules.

Data set	SGsN	PCsN	SGsN+PCsN	STsN+SGsN+PCsN
Indian Pines	95.49±0.80	98.38±0.21	99.04±0.14	**99.12±0.19**
University of Pavia	96.51±0.59	98.21±0.35	98.75±0.17	**99.49±0.14**
Salinas	98.81±0.48	97.47±0.67	98.55±0.49	**99.36±0.41**

the STsN module, the performance is improved. This finding indicates that the STsN can enhance the discrimination of spectra and optimize the feature pattern so that the subsequent SGsN and PCsN are more compatible, which is also proven by the results of the other two data sets.

The SLIC is used to partition HSI into superpixels to calculate the association matrix and adjacency mask. As the SLIC is unsupervised, applying it to HSI directly may reduce

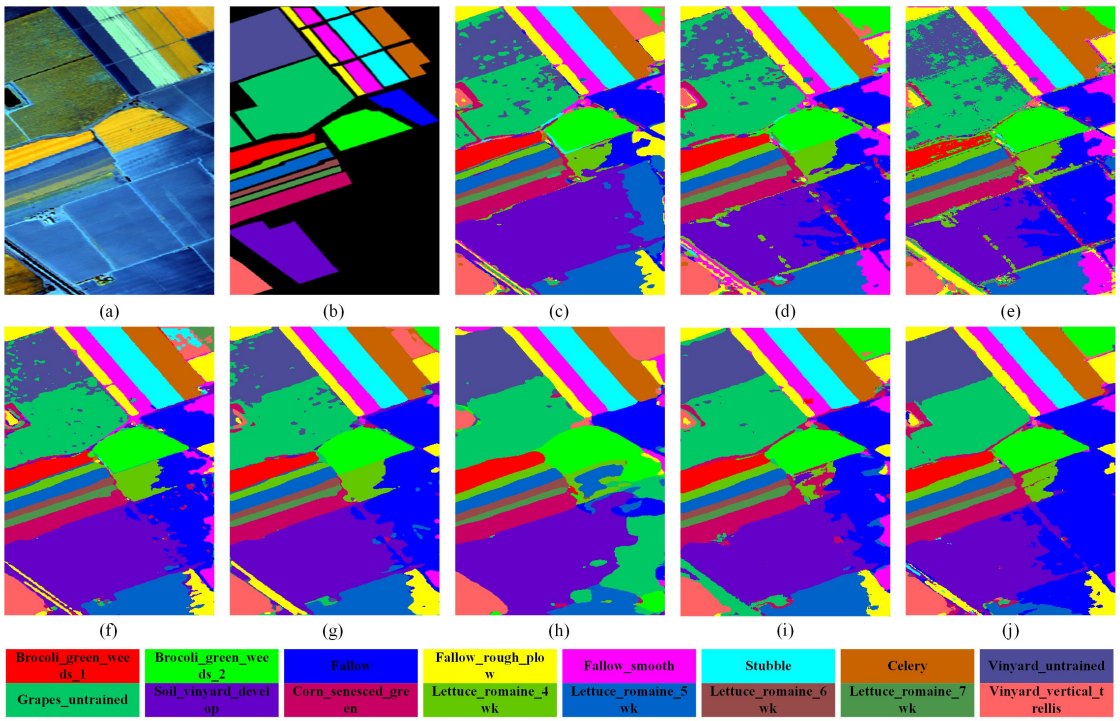

Fig. 6: Classification maps of different methods for the Salinas data set. (a) False-color image. (b) Ground truth. (c) DCNN (OA = 94.72%). (d) TC-CNN (OA = 94.31%). (e) MCCNN (OA = 92.59%). (f) SSRN (OA = 96.03%). (g) FDSSC (OA = 98.03%). (h) HybridSN (OA = 99.27%). (i) DBDA (OA = 97.01%). (j) CEGCN (OA = 99.36%).

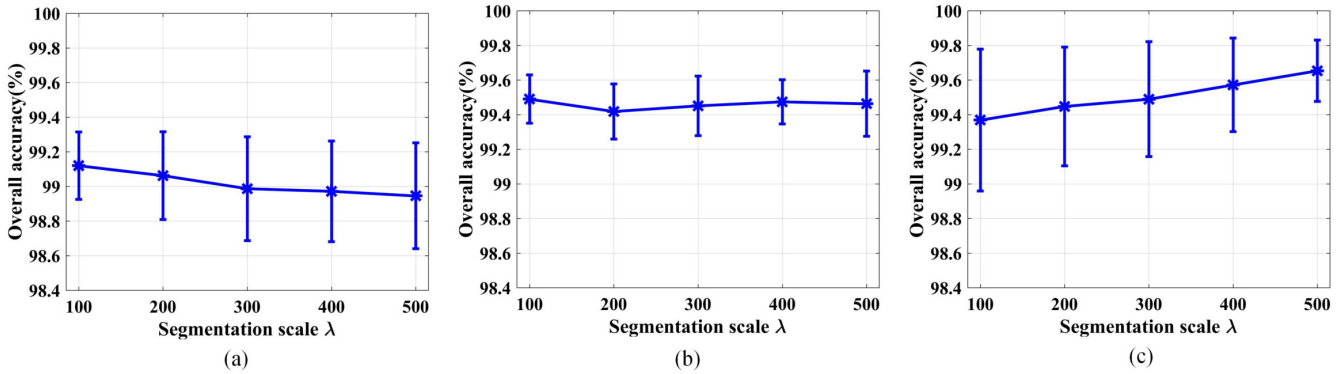

Fig. 7: Classification accuracies of CEGCN with different segmentation scales on each data set. (a) Indian Pines, (b) University of Pavia, and (c) Salinas.

TABLE IX: The OA(%) indices of CEGCN with and without the LDA.

Data set	Without LDA	With LDA
Indian Pines	99.09±0.18	**99.12±0.19**
University of Pavia	99.45±0.24	**99.49±0.14**
Salinas	99.27±0.28	**99.36±0.41**

the accuracy of segmentation. Hence, we first use LDA to pretreat the HSI. To evaluate the effectiveness of LDA in our method, we compare the performance of the CEGCN with and without the LDA. The compared results are shown in Table IX. With the LDA, the CEGCN achieves higher classification accuracies on all the data sets.

B. Influence of Segmentation Scale

The segmentation scale λ could affect the classification results of CEGCN. To analyze its influence on our method, in this section, we set the segmentation scale λ to 100, 200, 300, 400, and 500, and test their classification accuracies on each data set. Each experiment is repeated ten times, and the mean and standard deviation of OA indices are shown in Fig.7.

The segmentation scale determines the size of the constructed graph. A larger scale leads to a smaller graph that preserves larger objects and suppresses more noise. In contrast, a smaller scale results in a larger graph that retains smaller objects while including more noise. As seen, the accuracy of the CEGCN on the Indian Pines data set decreases with the

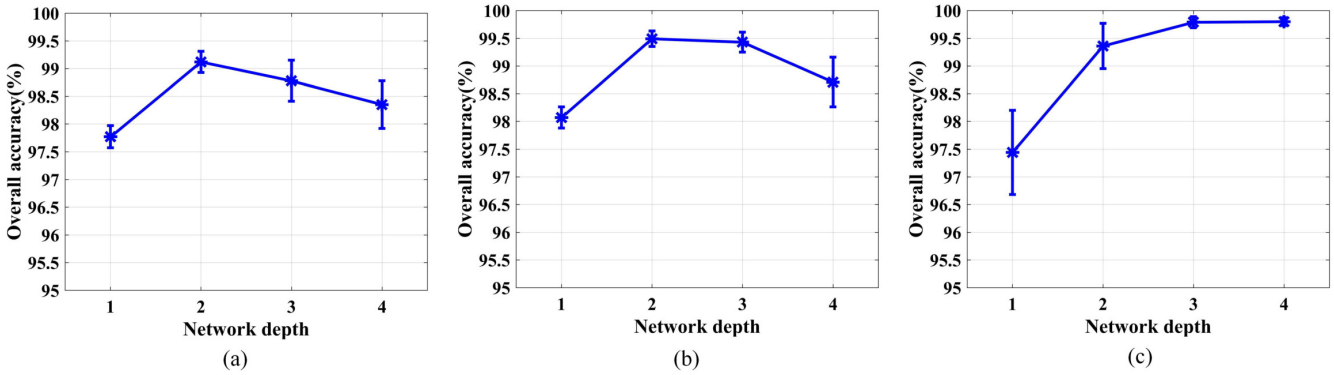

Fig. 8: Classification performance of CEGCN with different network depth on each data set. (a) Indian Pines, (b) University of Pavia, and (c) Salinas.

TABLE X: OA(%) indices of CEGCN with different superpixel segmentation methods.

Data set	SLIC	QS	FH	LSC	SEEDS
Indian Pines	99.12±0.19	99.10±0.29	99.09±0.25	99.14±0.33	**99.25±0.14**
University of Pavia	99.49±0.14	99.47±0.18	**99.54±0.08**	99.51±0.11	99.45±0.17
Salinas	99.36±0.41	**99.73±0.12**	99.51±0.31	99.48±0.21	99.55±0.22

increase of scale, which indicates that there are more small objects existing in this HSI so that smaller scale could preserve more details. In contrast, the accuracy of the CEGCN on the Salinas data sets increases with the increase of scale, which shows that this HSI may consist of more large-scale objects. On the University of Pavia data set, different scales obtain similar performances, which may be because this HSI contains many large and small objects at the same time. To prevent the CEGCN from obtaining over-smoothed classification maps, we fix it to 100 in all experiments.

C. Influence of Segmentation Methods

In our method, the SLIC is adopted as the superpixel segmentation tool due to its simple implementation and easy-using. In practical applications, the SLIC can be replaced with a more adaptive segmentation tool to achieve better performance. In this section, we test several state-of-the-art superpixel segmentation methods on the CEGCN, including the SLIC [45], the quick shift (QS) [52], the Felzenszwalb and Huttenlocher's method (FH) [53], the linear spectral clustering (LSC) [54], and the SEEDS [55]. All the segmentation methods have been tuned to adapt the CEGCN, and their specific hyperparameter setting can be found in the released code (see Section III-A). The experimental configuration is the same as Section III. And the corresponding results are summarized in Tabel X.

As can be seen, with different segmentation methods, the CEGCN still performs well, which proves its robustness and effectiveness. However, the segmentation methods that enable the CEGCN to achieve the best performance on different data sets are different, which indicates that different segmentation methods have different applicability on different HSIs. Considering that the results acquired with these methods have little difference, in practice, we only need to choose an appropriate segmentation method according to experience.

D. Influence of Network Depth

The depth of network, which is a crucial hyperparameter, has a significant impact on the performance of the network. In general, due to the gradient vanishing, deepening a network does not increase the performance invariably. As a result, a suitable network depth could improve its stability and performance. In this section, we let $L_T = L_S = L_P = 1$, 2, 3, and 4 respectively, and then test the performance of CEGCNs with different network depth. Each experiment is repeated ten times, and the OA and standard variance are taken as the evaluation indices.

As shown in Fig. 8, for the Indian Pines data set, the CEGCN achieves the best performance with the network depth equal to 2. For the University of Pavia data set, setting the network depth to 2 or 3 is the best choice. However, for the Salinas data set, the performance of the CEGCN increases with the increase of network depth. This phenomenon may be caused by the inherent smoothness property of the GCN, i.e., the deeper the GCN, the smoother the classification map. Since the Salinas is composed of more large-scale land covers, over-smoothing its classification map could suppress more noise or outliers and then result in higher accuracy, which is consistent with the conclusion in Section IV-B. Nevertheless, this property of GCN is harmful to the HSIs containing small-scale objects, as a result, too deep CEGCN could lead to a performance degeneration on the Indian Pines and University of Pavia data sets. Thus, the network depth of CEGCN is fixed to 2 in all the other experiments.

E. Analysis of Graph Learning

The SGsN module in our method can model the spatial structure of HSI. Different from other methods, the edge weights in the SGsN are automatically learned from the training data set. To analyze the effectiveness of this module,

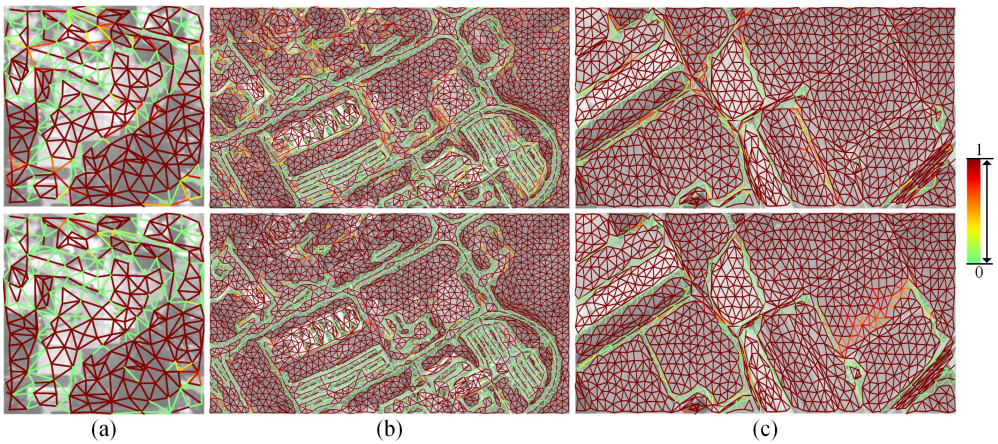

Fig. 9: Visualization of graphs on different data sets, where the two rows denote the graphs in the first and the second graph convolutional layer of the SGsN. (a) Indian Pines, (b) University of Pavia, and (c) Salinas.

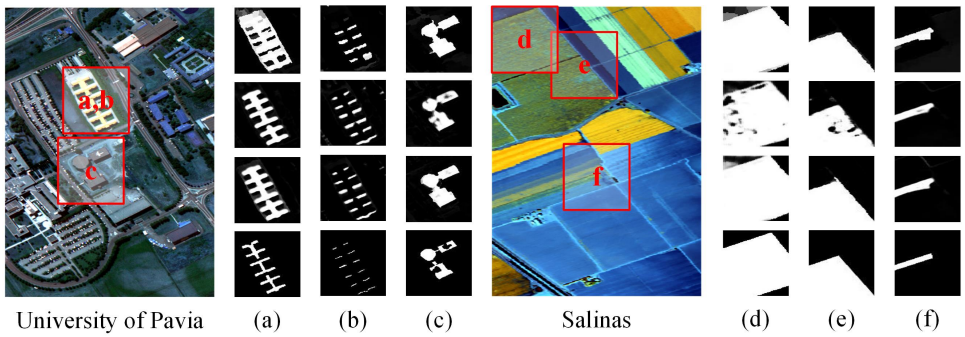

Fig. 10: Visualization of feature fusion, where the four rows denote the local probability maps obtained by SGsN, PCsN, CEGCN, and ground truth. (a-c) The probability maps of the 5th, 9th and 7th classes of the University of Pavia data set. (d-f) The probability maps of the 15th, 8th and 14th classes of the Salinas data set.

TABLE XI: The OA(%) indices of CEGCN with and without the adjacency mask.

Data set	Without Mask	With Mask
Indian Pines	98.91±0.23	**99.12±0.19**
University of Pavia	98.82±0.14	**99.49±0.14**
Salinas	98.02±0.47	**99.36±0.41**

we visualize the learned graphs of each data set, as shown in Fig. 9.

As can be seen, the graphs clearly reveal the spatial structure of each HSI. The edge weights tend to be divided into two opposing extremes, i.e., the maximum value one (red) or the minimum value zero (green). It indicates that, in a homogeneous region, information can be propagated by strong edges, which results in smoothed classification regions. In contrast, the propagation of information is blocked at the borders between different land covers, which could prevent features from being contaminated by different land covers. As a result, such a graph structure with local clusters encourages piecewise smoothed classification results. Additionally, the graphs learned in the two hidden layers are similar, which inspires us to share the same graph in the SGsN and further reduce the parameters in future work.

Instead of learning a fully connected graph, we use the adjacency mask to restrict the information propagation to only adjacent regions. On the one hand, an object is more affected by its neighboring objects but less or not affected by distant objects. On the other hand, it could be hard to learn an exact fully connected graph with limited samples. Hence, as prior information, the adjacency mask can help improve the accuracy of the graph learning. We test the classification performance of the CEGCN with and without the adjacency mask, as shown in Table XI. Obviously, with the adjacency mask, the CEGCN obtains better performance.

F. Analysis of Feature Fusion

In our method, the SGsN and PCsN can work jointly. By fusing the features extracted by the SGsN and PCsN, the CEGCN shows higher classification performance. To analyze the effectiveness of the feature fusion, in this section, we visualize several representative probability maps obtained by SGsN, PCsN, and CEGCN, as shown in Fig. 10.

As seen in Fig. 10-(a, f), the SGsN obtains over-smoothed probability maps compared with the PCsN. In Fig. 10-b, the SGsN fails to preserve small regions while the PCsN obtains better results. It indicates that the SGsN can lose small objects and over-smooth classification maps. However, after fusing them, the probability maps become more consistent with the

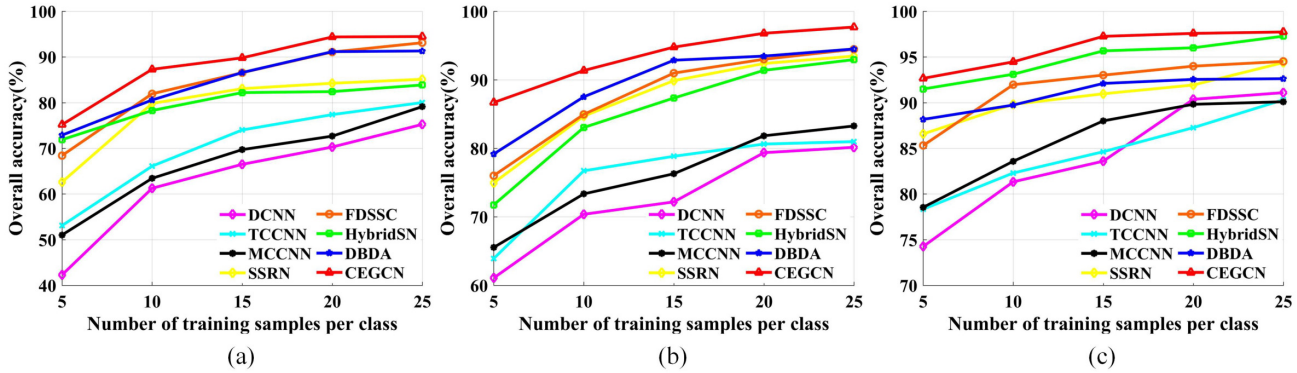

Fig. 11: Classification accuracies of different methods under different scales of training samples. (a) Indian Pines, (b) University of Pavia, and (c) Salinas.

TABLE XII: Running time (s) of different methods on each data set.

Data set	Time	DCNN	TCCNN	MCCNN	SSRN	FDSSC	HybridSN	DBDA	CEGCN
Indian Pines	Train.	40.2	68.3	59.7	185.7	111.6	239.8s	278.3	**16.3**
	Test.	5.7	5.8	6.7	11.6	16.9	9.7s	19.9	**0.6**
University of Pavia	Train.	**22.1**	40.5	36.5	120.8	70.2	97.1s	136.5	163.7
	Test.	30.9	28.8	35.8	61.2	93.1	52.6s	110.7	**5.2**
Salinas	Train.	137.6	209.5	192.1	176.6	142.4	375.1	230.1	**79.6**
	Test.	14.3	17.4	19.9	51.0	89.4	46.4	103.8	**3.6**

ground truth. It indicates that the PCsN could correct and optimize the SGsN in small regions. In contrast, as shown in Fig. 10-(c, d, e), the PCsN fails to obtain the connected probability maps in flat regions, while the SGsN achieves finer results. By fusing them, the CEGCN obtains more accurate results. These results above indicate that the CEGCN tends to rely on the smoothed features of SGsN in flat regions, but in small-object regions, the CEGCN is more likely to use the sophisticated features of the PCsN. By feature fusion, the CEGCN can achieve satisfying classification results in flat and small object regions at the same time.

G. Comparison of Running Time

In this section, the training and testing time of different methods on each data set are reported in Table XII. The experimental conditions are the same as those in Section III. Because HSI classification aims to assign a label to every pixel in this HSI, we take the time of classifying all the pixels as the testing time. Note that the time spent by the SGC in our method is added to the running time. Different from other deep learning methods that take small HSI cubes as input, our method uses the whole HSI as input, which allows the computation to be performed in parallel. As a result, our method has a very fast classification speed. Furthermore, given the lightweight network architecture, the training process of our method is also faster, which shows the promise of the CEGCN for applications in the industrial world.

H. Performance Under Limited Samples

In most cases, labeling the pixels of HSI is usually expensive in terms of human and time resources. As a result, the small-sample learning capacity of a classification model determines its applicability. To evaluate the proposed CEGCN in the case of very limited training samples, we report the OA indices of different methods under different sizes of training samples. Specifically, in each experiment on each data set, we randomly select 5, 10, 15, 20, and 25 samples per class as training samples and 1 sample per class as a validation sample. Each experiment is repeated ten times, and the mean of OA indices are shown in Fig. 11. As can be seen, the proposed CEGCN surpasses all the compared methods on the three data sets, which proves the great small-sample learning ability of our method.

V. Conclusion

Owing to the ability to perform convolutions on arbitrarily structured graphs, the GCN has been used to exploit the correlation between adjacent land covers and model their spatial structures. To reduce the computational complexity and promote the semantic structure learning of land covers, HSI is usually pretreated as superpixels and then encoded into a graph. However, such a superpixel-based GCN cannot classify HSI at the pixel level, which leads to the loss of small regions and makes classification maps over-smoothed. In this work, we propose to utilize CNN to generate local pixel-level features to supplement the superpixel-level features of the GCN. To alleviate the structural incompatibility of the data between the CNN and GCN, we propose the graph encoder and decoder to propagate features between graph nodes and image pixels, which makes the graph features adaptive to the image domain and thus enables the CNN and GCN to work jointly. Furthermore, to make the graph in the CEGCN more adaptive to HSI content, we also propose to learn the edge weights from training data. With the graph encoder and decoder, the CNN and GCN are integrated into the CEGCN to

generate complementary pixel- and superpixel-level features, which achieves a higher classification performance and a faster inference speed. In future works, considering that the proposed encoder and decoder could function as a bridge connecting the CNN and GCN, we will further explore multiscale architectures and attention mechanisms to make full use of their complementary advantages. For the graph learning in different layers, we will further improve the CEGCN by dynamic graph mechanisms to boost the feature extraction. Besides, since the feature fusion in our method is relatively simple, advanced feature fusion mechanisms adapted for the CEGCN will be also explored.

REFERENCES

[1] P. Ghamisi, J. Plaza, Y. Chen, J. Li, and A. J. Plaza, "Advanced spectral classifiers for hyperspectral images: A review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 1, pp. 8–32, March 2017.

[2] 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.

[3] G. Lu and B. Fei, "Medical hyperspectral imaging: a review," *Journal of Biomedical Optics*, vol. 19, no. 1, p. 010901, Jan. 2014.

[4] A. Ghiyamat and H. Z. M. Shafri, "A review on hyperspectral remote sensing for homogeneous and heterogeneous forest biodiversity assessment," *Journal of remote sensing*, vol. 31, no. 7, pp. 1837–1856, Apr. 2010.

[5] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4085–4098, Nov. 2010.

[6] C. Chen, W. Li, H. Su, and K. Liu, "Spectral-spatial classification of hyperspectral image based on kernel extreme learning machine," *Remote Sensing*, vol. 6, no. 6, pp. 5795–5814, Jun. 2014.

[7] B. Kuo, H. Ho, C. Li, C. Hung, and J. Taur, "A kernel-based feature selection method for svm with rbf kernel for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 1, pp. 317–326, Jan. 2014.

[8] Y. Gu, C. Wang, D. You, Y. Zhang, S. Wang, and Y. Zhang, "Representative multiple kernel learning for classification in hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 7, pp. 2852–2865, July 2012.

[9] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 10, pp. 3973–3985, Oct. 2011.

[10] Y. Y. Tang, H. Yuan, and L. Li, "Manifold-based sparse representation for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 12, pp. 7606–7618, Dec. 2014.

[11] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification via kernel sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 1, pp. 217–231, Jan. 2013.

[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 Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2739–2751, June 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 Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 4032–4046, July 2017.

[14] H. Su, B. Zhao, Q. Du, P. Du, and Z. Xue, "Multifeature dictionary learning for collaborative representation classification of hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 4, pp. 2467–2484, April 2018.

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

[16] H. Su, Y. Yu, Q. Du, and P. Du, "Ensemble learning for hyperspectral image classification using tangent collaborative representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 6, pp. 3778–3790, June 2020.

[17] G. Camps-Valls, T. V. Bandos Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 10, pp. 3044–3054, Oct. 2007.

[18] L. Ma, M. M. Crawford, X. Yang, and Y. Guo, "Local-manifold-learning-based graph construction for semisupervised hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 5, pp. 2832–2844, May 2015.

[19] L. Fang, S. Li, X. Kang, and J. A. Benediktsson, "Spectral–spatial classification of hyperspectral images with a superpixel-based discriminative sparse model," *IEEE Transactions on Geoscience and Remote Sensing*, 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 Recognition*, vol. 37, no. 6, pp. 1097–1116, June 2004.

[21] Y. Tarabalka, J. Chanussot, and J. A. Benediktsson, "Segmentation and classification of hyperspectral images using watershed transformation," *Pattern Recognition*, vol. 43, no. 7, pp. 2367–2379, July 2010.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[23] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *north american chapter of the association for computational linguistics (NAACL)*, 2019, pp. 4171–4186.

[24] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, and A. Bolton, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[25] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2015, pp. 4959–4962.

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

[27] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "Hybridsn: Exploring 3-d–2-d cnn feature hierarchy for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 2, pp. 277–281, Feb. 2020.

[28] L. Mou, P. Ghamisi, and X. Z. Xiao, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, July 2017.

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

[30] L. Ying, H. Zhang, and S. Qiang, "Spectral-spatial classification of hyperspectral imagery with 3d convolutional neural network," *Remote Sensing*, vol. 9, no. 1, p. 67, Jan. 2017.

[31] 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 *IEEE International Conference on Intelligent Computing (ICIC)*. Cham, Switzerland: Springer, Jul. 2018, pp. 81–92.

[32] S. Hao, W. Wang, Y. Ye, T. Nie, and L. Bruzzone, "Two-stream deep architecture for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 4, pp. 2349–2361, Dec. 2018.

[33] Z. Zhong, J. Li, Z. Luo, M. Chapman, Z. Zhong, J. Li, Z. Luo, and M. Chapman, "Spectral-spatial residual network for hyperspectral image classification: A 3-d deep learning framework," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 847–858, Feb. 2018.

[34] W. Wang, D. Shuguang, J. Zhongmin, and S. Liujie, "A fast dense spectral-spatial convolution network framework for hyperspectral images classification," *Remote Sensing*, vol. 10, no. 7, p. 1068, Jul. 2018.

[35] S. Hao, W. Wang, Y. Ye, E. Li, and L. Bruzzone, "A deep network architecture for super-resolution-aided hyperspectral image classification with classwise loss," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 8, pp. 4650–4663, June 2018.

[36] C. Xing, L. Ma, and X. Yang, "Stacked denoise autoencoder based feature extraction and classification for hyperspectral images," *Journal of Sensors*, vol. 2016, 2016.
[37] X. Mei, E. Pan, Y. Ma, X. Dai, J. Huang, F. Fan, Q. Du, H. Zheng, and J. Ma, "Spectral-spatial attention networks for hyperspectral image classification," *Remote Sensing*, vol. 11, no. 8, p. 963, Apr. 2019.
[38] R. Li, S. Zheng, C. Duan, Y. Yang, and X. Wang, "Classification of hyperspectral image based on double-branch dual-attention mechanism network," *Remote Sensing*, vol. 12, no. 3, p. 582, Feb. 2020.
[39] Q. Liu, L. Xiao, J. Yang, and J. C. W. Chan, "Content-guided convolutional neural network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 9, pp. 6124–6137, Sept. 2020.
[40] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 764–773.
[41] 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 Geoscience and Remote Sensing Letters*, vol. 16, no. 2, pp. 241–245, Feb. 2019.
[42] S. Wan, C. Gong, P. Zhong, B. Du, L. Zhang, and J. Yang, "Multiscale dynamic graph convolutional network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 5, pp. 3162–3177, May 2020.
[43] T. Lu, S. Li, L. Fang, X. Jia, and J. A. Benediktsson, "From subpixel to superpixel: A novel fusion framework for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 8, pp. 4398–4411, Aug. 2017.
[44] Q. Liu, L. Xiao, F. Liu, and J. Huan, "SSCDenseNet:a spectral-spatial convolutional dense network for hyperspectral image classification," *Acta Electronica Sinica (in Chinese)*, vol. 48, no. 4, pp. 751–762, Nov. 2020.
[45] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.
[46] V. Jampani, D. Sun, M.-Y. Liu, M.-H. Yang, and J. Kautz, "Superpixel sampling networks," in *European Conference on Computer Vision (ECCV)*, Sep. 2018.
[47] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, May 2013.
[48] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems (NIPS)*, 06 2016, pp. 3844–3852.
[49] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International conference on learning representations (ICLR)*, 2017.
[50] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning (ICML)*, vol. 30, no. 1, 2013, p. 3.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International conference on learning representations (ICLR)*, May 2015.
[52] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *European Conference on Computer Vision (ECCV)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 705–718.
[53] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, Sep. 2004.
[54] Zhengqin Li and Jiansheng Chen, "Superpixel segmentation using linear spectral clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1356–1363.
[55] V. D. B. Michael, X. Boix, G. Roig, and L. Van Gool, "SEEDS: Superpixels extracted via energy-driven sampling," *International Journal of Computer Vision*, vol. 111, no. 3, pp. 298–314, Feb. 2015.

Qichao Liu (S'18) received the B.S. degree in computer science and technology from the Nanjing University of Science and Technology, Nanjing, China, in 2015.

He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering in Nanjing University of Science and Technology. His fields of interest include deep learning and remote sensing image processing.

Liang Xiao (M'11) received the B.S. degree in applied mathematics and the Ph.D. degree in computer science from the Nanjing University of Science and Technology (NJUST), Nanjing, 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, China. 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 (S' 14, M' 19) received the joint Ph.D degrees in control theory and control engineering and Engineering Science from Northwestern Polytechnical University, China, and Vrije Universiteit Brussel, Belgium, in 2019.

Currently he is a Lecturer in Nanjing University of Science and Technology. 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 South East University, Nanjing, China, in 1983, 1986, and 2003, respectively.

He is currently the Professor and Doctoral Supervisor of the Nanjing University of Science and Technology. His current research interests include mathematical image modeling, multi-scale 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.