

## **Dual Graph Convolutional Network for Hyperspectral Image Classification With Limited Training Samples**

He, X.; Chen, Y.; Ghamisi, P.;

Originally published:

March 2021

**IEEE Transactions on Geoscience and Remote Sensing 60(2021), 5502418**

DOI: <https://doi.org/10.1109/TGRS.2021.3061088>

Perma-Link to Publication Repository of HZDR:

<https://www.hzdr.de/publications/Publ-33607>

Release of the secondary publication  
on the basis of the German Copyright Law § 38 Section 4.

# Dual Graph Convolutional Network for Hyperspectral Image Classification With Limited Training Samples

Xin He<sup>ID</sup>, Yushi Chen<sup>ID</sup>, *Member, IEEE*, and Pedram Ghamisi<sup>ID</sup>, *Senior Member, IEEE*

**Abstract**—Due to powerful feature extraction capability, convolutional neural networks (CNNs) have been widely used for hyperspectral image (HSI) classification. However, because of a large number of parameters that need to be trained, sufficient training samples are usually required for deep CNN-based methods. Unfortunately, limited training samples are a common issue in the remote sensing community. In this study, a dual graph convolutional network (DGCN) is proposed for the supervised classification of HSI with limited training samples. The first GCN fully extracts features existing in and among HSI samples, while the second GCN utilizes label distribution learning, and thus, it potentially reduces the number of required training samples. The two GCNs are integrated through several iterations to decrease interclass distances, which leads to a more accurate classification step. Moreover, a new idea entitled multiscale feature cutout is proposed as a regularization technique for HSI classification (DGCN-M). Different from the regularization methods (e.g., dropout and DropBlock), the proposed multiscale feature cutout could randomly mask out multiscale region sizes in a feature map, which further reduces the overfitting problem and yields consistent improvement. Experimental results on the four popular hyperspectral data sets (i.e., Salinas, Indian Pines, Pavia, and Houston) indicate that the proposed method obtains good classification performance compared to state-of-the-art methods, which shows the potential of GCN for HSI classification.

**Index Terms**—Classification, convolutional neural network (CNN), dual graph convolutional network (DGCN), hyperspectral image (HSI), label distribution learning.

## I. INTRODUCTION

**H**YPERSPECTRAL images (HSIs) often contain rich spectral and spatial information [1], which makes such data useful for a variety of applications, such as urban planning, vegetation monitoring, and target detection [2]. A basic and important technique to process HSIs is classification, which assigns a particular category to each pixel in the

HSI. A large number of HSI classification methods have been proposed, which considers spatial and spectral information [3], [4]. Due to high data complexity and the limited availability of training samples, the accurate classification of HSI is still challenging.

In the early stage of HSI spectral–spatial classification, the proposed methods obtained the spectral and spatial information separately [5]. For example, approaches based on spatial filters (e.g., morphological operators and low-rank representation) have been proven to be promising in extracting spatial information from HSIs [6]. Among these filtering approaches, one can mention, the morphological profiles (MPs) [7], extended MPs (EMPs) [8], and extended multiattribute profiles (EMAP) [9]. Then, the features extracted by those approaches were combined with the results of spectral methods; among them, the kernel-based methods have been widely used to classify HSI with limited training samples [10], [11]. Moreover, other low-rank representation methods were introduced to utilize a subspace learning technique, which aims to study the underlying low-dimensional subspace structures, and, hence, removed the redundancy of the image [12], [13]. For instance, in [14], a low-rank structured prior method was exploited to obtain the spatial dependences of the neighboring pixels. Then, the spatial information was integrated with spectral information extracted by a classifier to shape the final classification map. However, due to the fact that HSIs have 3-D structures, these separated frameworks cannot fully exploit the spectral and spatial relations.

Many studies have focused on extracting joint spectral–spatial information for HSI classification, mostly based on 3-D structure-based methods to fully exploit the 3-D nature of hyperspectral data cubes [15]. A series of methods have been developed based on the extension of classical 2-D algorithms, such as the 3-D Gabor [16], 3-D MP [17], and 3-D local binary pattern [18], as well as feature fusion frameworks with the typical 3-D methods [19]. These methods focus on designing different 3-D feature extraction operators to extract spectral and spatial information from different angles (i.e., morphology, local dependence, and shape smoothness).

Most of the aforementioned HSI classification methods do not deeply and automatically extract features. Recently, deep learning-based methods have attracted lots of attention and led to accurate classification, which can progressively

Manuscript received November 5, 2020; revised December 20, 2020 and January 26, 2021; accepted February 16, 2021. This work was supported by the Natural Science Foundation of China under Grant 61971164. (Corresponding author: Yushi Chen.)

Xin He and Yushi Chen are with the School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China (e-mail: 19B905007@stu.hit.edu.cn; cheniyushi@hit.edu.cn).

Pedram Ghamisi is with the Helmholtz Institute Freiberg for Resource Technology, Helmholtz-Zentrum Dresden-Rossendorf, 01328 Dresden, Germany, and also with the Institute of Advanced Research in Artificial Intelligence (IARAI), 1030 Vienna, Austria (e-mail: p.ghamisi@gmail.com).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TGRS.2021.3061088>.

Digital Object Identifier 10.1109/TGRS.2021.3061088

0196-2892 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

See <https://www.ieee.org/publications/rights/index.html> for more information.

and automatically extract discriminate and abstract features. Among existing deep models, the convolutional neural network (CNN) has been extensively explored for HSI classification [20]. The existing CNN-based HSI classification approaches can be roughly divided into three sets of approaches: 1) methods that are based on the combination of handcrafted features and the CNN-based extracted features; 2) methods that are purely based on the features extracted by CNN; and 3) the CNN-related techniques.

At the beginning of CNN-based HSI classification, due to the fact that the high dimensionality of HSIs usually causes the problem of overfitting, handcrafted feature extraction and CNN were combined to classify such data [21]. The first article that applied CNN in HSI classification utilized the principal component analysis as a preprocessing step to reduce the number of features, and then, these features were fed into convolutional layers to extract discriminative features [22]. A similar work introduced the local discriminant embedding algorithm to reduce the dimensions of HSI and generate spectral features, and then, CNN extracted the spatial features [23]. Besides, other handcrafted features based on multiscale covariance maps [24], attribute profiles [25], and Gabor features [26] were combined with deep features extracted by CNN to train the model to produce final classification results.

With the development of machine learning techniques, CNN can be used solely to extract features and classify HSI without a need for other handcrafted feature extraction methods. For example, in [27], deep CNN was utilized to extract pixel-pair features and, hence, jointly exploited spatial-spectral information to achieve better performance. Zhang *et al.* [28] proposed a dual-channel CNN, which combined spectral features extracted by 1-D-CNN with spatial features extracted by 2-D-CNN. Xu *et al.* [29] presented a unified spectral-spatial network to obtain discriminative features for HSI classification. In addition, due to the fact that the input of CNN is a 3-D patch, the straightforward way to classify HSI is to utilize 3-D kernels to obtain features [30]. He *et al.* [31] proposed a 3-D deep CNN, which is able to jointly extract spatial and spectral information by computing multiscale features. Besides, the spectral-spatial residual network (SSRN) explored by Zhong *et al.* [32] utilized consecutive spectral and spatial residual blocks with different sizes of 3-D convolutional kernels to extract spectral and spatial features separately. However, the complexity of the models, which involves 3-D convolutional kernels, is very high. To partially address this issue, another extension of the CNN was introduced based on the separable convolutions or the separable filter learning [33], [34]. For example, the large 3-D convolutional kernels were decomposed into small convolutional kernels to reduce computational complexity, while it also achieved better performance [35].

Moreover, some methods related to original CNNs were also proposed for HSI classification. For example, residual networks and densely connected CNN were used to enhance classification performance [36]–[39]. In addition, the cascaded recurrent neural network (RNN) was proposed for the pixel-level HSI classification task [40]. Moreover, the capsule

network was introduced to achieve better results by utilizing capsule units [41], [42].

Recently, several GCN-based methods have been proposed for HSI supervised and semisupervised classification. Limited labeled samples are a common issue in the practice of HSI classification, and GCN-based semisupervised learning was found to be a good solution [43]. Furthermore, in [44], a graph attention network, which is a modification of GCN, was proposed for HSI semisupervised classification. To explore the spatial information of HSI, context-aware GCN and multiscale GCN were proposed in [45] and [46], respectively. Besides, different from the above semisupervised GCN strategies for HSI classification, a nonlocal graph convolutional network was explored, which took the whole HSI as input to learn graph representations [47]. Very recently, a minibatch GCN was proposed for HSI classification with low computational cost [48]. In general, GCN-based methods have shown their potential in HSI classification, but many practical issues, such as limited training samples, still need to be addressed.

Due to the fact that the manual annotation of a large number of training samples is laborious and time-demanding in real scenarios, the number of training samples is usually very limited. Especially, this issue becomes more serious when it comes to HSIs. To address this problem, one possible solution aims to incorporate unlabeled samples in semisupervised hyperspectral classification [49]. Other ways including feature extraction, feature selection, or sparse learning have also been investigated to overcome the limitation of labeled samples for HSI classification [50], [51]. Therefore, we address this major challenge in HSI classification by introducing dual graph methods with only five training samples per class.

Different from the above GCN methods, in this article, the dual graph convolutional network (DGCN) is proposed, which explicitly calculates the similarities among samples and integrates label distribution learning into graph learning. Especially, this article not only captures relations among samples but also obtains label distribution relations among samples. In addition, the style of capturing the relations among samples is more intuitive for DGCN. For example, edges in the proposed DGCN are defined and updated by calculating similarities between nodes, which focuses more on the relations in the form of similarities between samples. Specifically, the spectral-spatial information is extracted by CNN without any preprocessing step to preserve discriminative features. Since CNN has a powerful capability for feature extraction, it is widely used in HSI processing. However, CNN extracts the features of an individual sample. Then, the learned features are fed into the proposed DGCN to capture the relationships among samples. Moreover, drop edge is used in the proposed DGCN (i.e., DGCN-D) for alleviating the overfitting problem in the case of the limited sample size [52]. Besides, multiscale feature cutout is proposed as a regularization method for DGCN (i.e., DGCN-M) to further improve the performance of the proposed classifier. It should be noted that this work is not a simple adaption of GCN for HSI classification. Instead, this work takes a step further to develop a novel classifier to accurately address the challenge

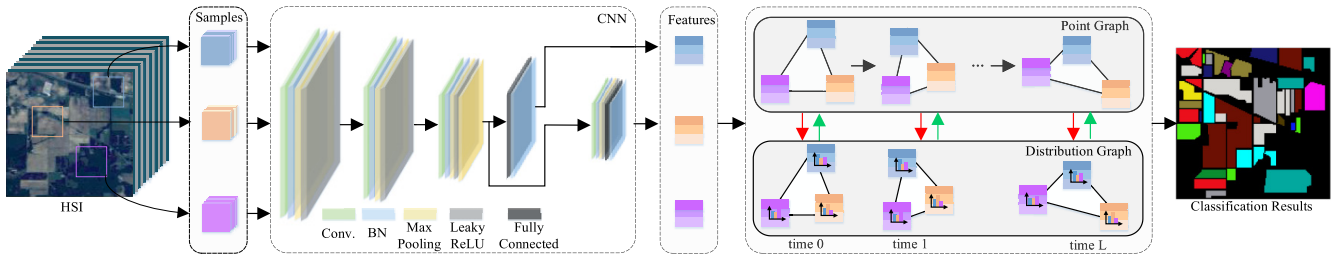


Fig. 1. Framework of the proposed DGCN, which integrates the features of the point graph and distribution graph. As an example, three samples of HSI are illustrated. The sample features are extracted by CNN. Then, these features are fed into the dual graph to further learn discriminative features finalize the classification task. The green arrow and red arrow represent the process of interacting features at each time  $L$ .

of the imbalance between the limited training samples and high dimensionality for HSI classification. In summary, the main contributions of this study are listed as follows.

- 1) A DGCN, which is a hybrid network of CNN and two graph networks (i.e., the point graph and the distribution graph), is proposed for HSI classification with limited training samples.
- 2) Instead of simply applying GCN to classify HSI, the point graph aims to fully explore the relationships among samples, and the distribution graph utilizes label distribution learning to obtain high correlation features among samples with the same label. The two graphs are integrated with each other to fully extract features among training samples; thus, DGCN reduces the number of needed training samples.
- 3) In order to mitigate the overfitting issue caused by limited HSI training samples, the drop edge is investigated in the proposed DGCN.
- 4) To further improve the HSI classification results, motivated by the regularization method called cutout, we propose a novel technique by improving the cutout with the multiscale operation to feature maps in CNN; this method increases the generalization capability of the proposed DGCN.

The rest of this article is organized as follows. Section II introduces the proposed methods, including DGCN, DGCN-D, and DGCN-M in detail. Section III presents comprehensive experiments, including data description, results, and analysis of the proposed methods. Finally, Section IV summarizes the main concluding remarks.

## II. PROPOSED METHODS

This section provides detailed information about the proposed methods termed DGCN, DGCN with drop edge (DGCN-D), and DGCN with multiscale feature cutout (DGCN-M).

### A. DGCN for HSI Classification

The proposed DGCN is composed of two graphs, named point graph and distribution graph. The two graphs aim to capture the relationships among samples. Fig. 1 shows an overview of DGCN for HSI classification, which includes three main parts (i.e., data preprocessing, CNN feature extraction, and dual graph learning). Detailed information on these three

parts is described below. Algorithm 1 describes the whole process of DGCN for HSI classification.

Let  $\mathbf{I} \in \mathbb{R}^{H \times W \times B}$  be the HSI, where  $H$ ,  $W$ , and  $B$  indicate the spatial height, spatial width, and band number of spectral signatures, respectively. To extract the spectral-spatial information from  $\mathbf{I}$ ,  $\mathbf{I}$  is first processed for data preparation. Each pixel  $i$  in  $\mathbf{I}$  forms a fixed square box, in which  $i$  is the center pixel and a fixed number of pixels around  $i$  are regarded as the adjacent pixels. Thus, the HSI eventually generates a total number of  $HW$  cubes, which represents a sample set  $\mathbf{I}' = \{\mathbf{I}'_1, \mathbf{I}'_2, \dots, \mathbf{I}'_{HW}\}$ . Here,  $\mathbf{I}'_i \in \mathbb{R}^{S \times S \times B}$  corresponds to the sample of the pixel  $i$ , and  $S \times S$  indicates the spatial size. The label set  $\mathbf{y} = \{y_1, y_2, \dots, y_{HW}\}$  represents the corresponding labels of  $\mathbf{I}'$ , where  $y_i \in \{1, 2, \dots, n\}$  is the center pixel of each sample, and  $n$  is the total classes of the ground truth.

Then, the samples are fed into CNN to extract spectral-spatial features in the second part. The CNN is composed of multiple layers, including the convolutional layer, the max-pooling layer, the Leaky-ReLU operation, and the fully connected (FC) layer. Among them, the convolutional layer is the main element, which can be described as follows:

$$f_{ij}^{xy} = \sum_o \sum_{p=1}^r \sum_{q=1}^s w_{ijm}^{qt} f_{(i-1)m}^{(x+q)(y+t)} + b_{ij} \quad (1)$$

where  $f_{ij}^{xy}$  indicates the output variable in the  $j$ th feature map at the  $i$ th layer,  $x$  and  $y$  represent the corresponding positions in the feature map,  $r$  and  $s$  are the kernel size,  $q$  and  $t$  are the kernel indices,  $m$  indicates the feature map index, and  $b$  is the bias.

CNN extracts features from different HSI samples independently without considering the relationships of different samples, while the proposed DGCN, including the point graph and the distribution graph, solves the disadvantages of CNN by exploring the relationships (i.e., label information and feature information) among samples. Third, DGCN passes the spectral-spatial features extracted by CNN to the subsequent dual graph learning. In the graph learning, the important elements for constructing each graph can be defined as  $G(V, E)$ , where  $V = \{v_k | \forall k \in \{1, \dots, N\}\}$  is the set of nodes,  $E = \{e_{k,v} | \forall k, v \in \{1, \dots, N\}\}$  is the set of edges, and  $N$  is the number of nodes in graph  $G$ . Besides, the core element of GCN is convolution; different from CNN, the convolution of GCN is represented in the form of nodes, which is defined in the spectral domain [53], [54]. The updated node representation



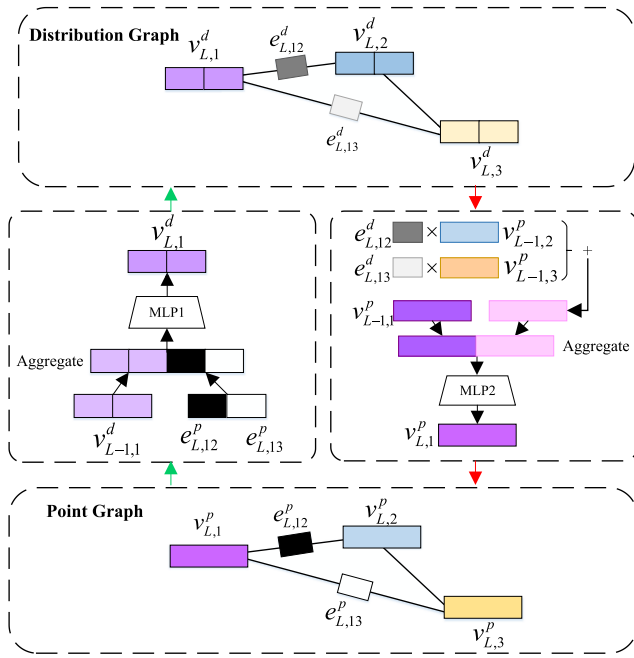


Fig. 2. Detailed process of the proposed DGCN that contains four stages.

by the convolution in GCN is defined as follows:

$$\hat{h}_k^{c+1} = \frac{1}{\deg_k} \sum_{v \in N_k} h_v^c \quad (2)$$

$$h_k^{c+1} = \sigma(U^c \hat{h}_k^{c+1}) \quad (3)$$

where  $h_k^{c+1}$  indicates the feature of node  $k$  at layer  $c+1$ . Node  $k$  connected to other nodes in the graph can be represented as  $N_k$ , degree of the node  $k$  forms  $\deg_k = |N_k|$ ,  $\sigma$  indicates the nonlinearity operation, and  $U^c \in R$  indicates the learned parameters.

The proposed DGCN contains two important graphs named point graph ( $G_L^p$ ) and distribution graph ( $G_L^d$ ). Specially,  $G_L^p$  calculates the similarities among the output features of CNN, and  $G_L^d$  aims to process the relationships of label distribution among  $I'$ . To integrate the output features (i.e., nodes and edges) of  $G_L^p$  and  $G_L^d$ , at each time  $L$  ( $L > 0$ ),  $G_L^p$  generates  $G_L^d$  by inserting the relationships of different samples into  $G_L^d$ , and  $G_L^d$  refines  $G_L^p$  by delivering the relationships of label distribution from  $I'$ , where  $L$  is the number of cycles. When  $L = 0$ , the two graphs initialize the nodes and edges, and when  $L > 0$ , the two graphs fuse their extracted features (i.e., nodes and edges) with each other. The detailed introduction is shown in Fig. 2. The dual graph learning in the proposed DGCN contains four stages, which can be summarized as follows four steps.

First, the point graph can be described as  $G_L^p = (V_L^p, E_L^p)$ , which consists of the node feature set  $V_L^p = \{v_{L,i}^p | i \in \{1, \dots, N\}\}$  and the edge feature set  $E_L^p = \{e_{L,ij}^p | i, j \in \{1, \dots, N\}\}$ . When  $L = 0$ , the output features of the second FC layer in CNN are used to initialize the node feature set  $V_L^p$  in  $G_L^p$ . The output features of the first FC layer in CNN are utilized to initialize the edge feature set  $E_L^p$  by computing samples' similarities among node features. Edge

#### Algorithm 1 DGCN for HSI Classification

1. **begin**
2. **initialize** the number of training samples  $n_{\text{train}}$ , fixed square length  $S$ , cyclic time  $L$ .
3. **for each pixel:**
4.   select the spatial size  $S \times S$  of each pixel, given the labels of each pixel, add them to the sample set  $I'$  and the label set  $y$ , respectively.
5.   split  $I'$  and  $y$  into training set according to  $n_{\text{train}}$ .
6. **Construct DGCN (If  $L = 0$ ):**
7.   **initialize** the point graph ( $G_L^p$ ):
8.   **for each sample  $I'_i$  in the training set:**
9.    feed  $I'_i$  into CNN with learned parameters  $w$  and  $b$ .
10.    initialize node features  $V_L^p$  of  $G_L^p$  according to the output features of CNN from  $I'$ .
11.    initialize edge features  $E_L^p$  of  $G_L^p$  among  $I'$ , for each edge feature  $e_{L,ij}^p$  between  $I'_i$  and  $I'_j$  in  $E_L^p$ :
12.       $e_{L,ij}^p = f_{e_L^p}((v_{L,i}^p - v_{L,j}^p)^2)$
13.   **initialize** the distribution graph ( $G_L^d$ ):
14.    initialize node features  $V_L^d$  of  $G_L^d$  according to the label distribution information among samples.
15.    initialize edge features  $E_L^d$  of  $G_L^d$  according to the  $V_L^d$ , for each edge feature  $e_{L,ij}^d$  between  $I'_i$  and  $I'_j$  in  $E_L^d$ :
16.       $e_{L,ij}^d = f_{e_L^d}((v_{L,i}^d - v_{L,j}^d)^2)$
17.   **Integrate  $G_L^p$  with  $G_L^d$  (If  $L > 0$ ):**
18.    **for each cyclic time  $L$ :**
19.    combine each node  $V_L^p$  in  $G_L^p$  with  $E_L^d$  in  $G_L^d$ .
20.    compute each edge in  $E_L^p$  according to the previous  $E_{L-1}^p$  in  $G_L^p$ .
21.    combine each node  $V_L^d$  in  $G_L^d$  with  $E_L^p$  in  $G_L^p$ .
22.    compute each edge in  $E_L^d$  according to the previous  $E_{L-1}^d$  in  $G_L^d$ .
23.    predict each node in  $V_L^p$  in  $G_L^p$ .
24. **end**

features  $E_L^p = \{e_{L,ij}^p | i, j \in \{1, \dots, N\}\}$  are constructed using neighboring nodes. Here, each edge feature stands for the similarities between different HSI samples, which can be defined as follows:

$$e_{L,ij}^p = \begin{cases} f_{e_L^p}((v_{L,i}^p - v_{L,j}^p)^2), & \text{if } L = 0, \\ f_{e_L^p}((v_{L,i}^p - v_{L,j}^p)^2) \cdot e_{L-1,ij}^p, & \text{if } L > 0 \end{cases} \quad (4)$$

where  $v_{L,i}^p$  in  $V_L^p$  represents the output feature of CNN from different  $I'_i$ 's. Especially, when  $L = 0$ ,  $v_{L,i}^p$  represents the output feature of the first FC layer, while, when  $L > 0$ ,  $v_{L,i}^p$  is the output feature of the second FC layer in CNN.  $f_{e_L^p}$  indicates a transform network, which includes two blocks and a convolution layer, and each block contains a convolution layer, a batch normalization (BN) layer, and a Leaky-ReLU operation. Here,  $e_{L,ij}^p$  is updated not only by the current feature but also the previous edge feature  $e_{L-1,ij}^p$ .

Second, the distribution graph is next to be constructed, which is described as  $G_L^d = (V_L^d, E_L^d)$ . Here,  $V_L^d = v_{L,i}^d | i \in$

$\{1, \dots, N\}$  and  $E_L^d = e_{L,ij}^d | \forall i, j \in \{1, \dots, N\}$  indicate the set of node features and edge features in  $G_L^d$ , respectively. This process aims to obtain  $V_L^d$ , where the order of node features  $V_L^d$  follows the position order in  $G_L^p$ , which can be described through (5). When  $L = 0$ ,  $y_i$  and  $y_j$  are the labels of samples  $I_i'$  and  $I_j'$ .  $\delta(\cdot)$  and  $\parallel$  indicate the concatenation operator and the Kronecker delta function, receptivity

$$v_{L,i}^d = \begin{cases} \parallel_{j=1}^N \delta(y_i, y_j), & \text{if } I_i' \text{ is training sample} \\ & \text{and } L = 0, \\ \left[ \frac{1}{NK}, \dots, \frac{1}{NK} \right], & \text{if } I_i' \text{ is not training sample} \\ & \text{and } L = 0, \\ f_{v_L^d}(\parallel_{j=1}^N e_{L,ij}^p, v_{L-1,i}^d), & \text{if } L > 0. \end{cases} \quad (5)$$

When  $L > 0$ ,  $e_{L,ij}^p$  in  $E_L^p$  is obtained by the last step, and  $f_{v_L^d}$  is the aggregated network MLP1, which contains an FC layer and a Leaky-ReLU operation.

Third,  $E_L^d$  in  $G_L^d$  is calculated using (6), and  $E_L^d$  represents the distribution similarities among  $V_L^d$ . Each edge feature represents the relationship of the current node to all other nodes in a one-versus- $N$  manner. Here,  $f_{e_L^d}$  is an MLP2 composed of two convolutional blocks and a sigmoid layer for computing  $e_{L,ij}^d$  in  $E_L^d$

$$e_{L,ij}^d = \begin{cases} f_{e_L^d}((v_{L,i}^d - v_{L,j}^d)^2), & \text{if } L = 0, \\ f_{e_L^d}((v_{L-1,i}^d - v_{L-1,j}^d)^2) \cdot e_{L-1,ij}^d, & \text{if } L > 0. \end{cases} \quad (6)$$

Finally, at the end of each time, in order to merge all the extracted features (i.e., nodes and edge information) of the two graphs, the generated  $E_L^d$  in  $G_L^d$  flows back into  $G_L^p$  to generate a new node set  $V_L^d$  in  $G_L^d$ , where each  $v_{L,i}^d$  in  $V_L^d$  represents a new node.  $v_{L,i}^p$  is obtained by aggregating all the node features through the last step with  $e_{L,ij}^d$ , which is according to the following formula:

$$v_{L,i}^p = f_{v_L^p} \left( \sum_{j=1}^N (e_{L,ij}^d \cdot v_{L-1,j}^p), v_{L-1,i}^p \right) \quad (7)$$

where  $f_{v_L^p}$  indicates a concatenated network, which includes two blocks to update  $v_{L,i}^p$ . Once  $L$  is finalized at this step,  $v_{L,i}^p$  could combine the features (i.e., nodes and edges) extracted by the point graph with the information obtained by the distribution graph. Thus, the above four stages are repeated several times  $L$  to fuse  $G_L^p$  with  $G_L^d$  sufficiently.

The loss function of the proposed DGCN contains two important elements, named the point loss  $\text{Loss}_L^p$  and the distribution loss  $\text{Loss}_L^d$ , which are defined as follows:

$$\text{Loss}_L^p = \text{Loss}_{\text{CE}}(P(\hat{y}_i | I_i), y_i) \quad (8)$$

$$\text{Loss}_L^d = \text{Loss}_{\text{CE}} \left( \text{Softmax} \left( \sum_{j=1}^N e_{L,ij}^d \cdot \text{one-hot}(y_j) \right), y_i \right) \quad (9)$$

where  $\text{Loss}_{\text{CE}}$  represents the cross-entropy loss function [55],  $P(\hat{y}_i | x_i)$  is the model probability predictions of the sample  $I_i$ , and  $y_i$  is the corresponding label.  $e_{L,ij}^d$  indicates the edge feature in  $G_L^d$  at time  $L$ .

## B. DGCN-D for HSI Classification

For the classification of HSIs using deep-learning-based approaches, there exist a huge number of parameters. When the number of training samples is limited, the performance of the network can be downgraded due to the overfitting problem. Here, in order to alleviate this issue and strengthen the generalization ability using small sample size, DGCN-D is introduced for the proposed DGCN, which aims to randomly drop a number of edges from the input graph at each training time. The whole process of DGCN for HSI classification is described in Algorithm 2.

DGCN-D is a flexible and effective method, which randomly drops out a certain rate of edges in the graph during the training process. Moreover, DGCN-D can be considered as the message passing reducer, which makes node connections sparser by dropping a certain number of edges, and thus, DGCN could go deeper than CNN by addressing the overfitting issue for HSI classification.

---

### Algorithm 2 DGCN-D for HSI Classification

---

**input:**

sample set  $I'$  and corresponding label set  $y$  of HSI.

**output:**

predict labels of test samples

1. **initialize** cyclic time  $L$  and drop edges with probability  $p$ .
  2. **Construct DGCN (If  $L = 0$ ):**
  3. **initialize** the point graph ( $G_L^p$ ):
  4. for each sample  $I_i'$  in the training set:
  5. feed  $I_i'$  into CNN with learned parameters  $w$  and  $b$ .
  6. initialize node features  $V_L^p$  of  $G_L^p$  according to the output features of CNN from  $I'$ .
  7. initialize edge features  $E_L^p$  of  $G_L^p$  among  $I'$ , for each edge feature  $e_{L,ij}^p$  between  $I_i'$  and  $I_j'$  in  $E_L^p$ :
  8.  $e_{L,ij}^p = f_{e_L^p}((v_{L,i}^p - v_{L,j}^p)^2)$
  9. for each edge feature  $e_{L,ij}^p$  in  $E_L^p$ :
  10. randomly set  $E_L^p$  to be zeros with  $p$
  11. **initialize** the distribution graph ( $G_L^d$ ):
  12. initialize node features  $V_L^d$  of  $G_L^d$  according to the label distribution information among samples.
  13. initialize edge features  $E_L^d$  of  $G_L^d$  according to the  $V_L^d$ , for each edge feature  $e_{L,ij}^d$  between  $I_i'$  and  $I_j'$  in  $E_L^d$ :
  14.  $e_{L,ij}^d = f_{e_L^d}((v_{L,i}^d - v_{L,j}^d)^2)$
  15. for each edge feature  $e_{L,ij}^d$  in  $E_L^d$ :
  16. randomly set  $E_L^d$  to be zeros with  $p$
  17. **Integrate  $G_L^p$  with  $G_L^d$  (If  $L > 0$ ):**
  18. for each cyclic time  $L$ :
  19. combine each node  $V_L^p$  in  $G_L^p$  with  $E_L^d$  in  $G_L^d$ .
  20. compute each edge in  $E_L^p$  according to the previous  $E_{L-1}^p$  in  $G_L^p$ .
  21. combine each node  $V_L^d$  in  $G_L^d$  with  $E_L^p$  in  $G_L^p$ .
  22. compute each edge in  $E_L^d$  according to the previous  $E_{L-1}^d$  in  $G_L^d$ .
  23. predict each node in  $V_L^p$  in  $G_L^p$ .
-

Here, the proposed DGCN-D can be defined by (10).  $A \in R^{M \times M}$  represents the adjacency matrix in the graph;  $A_{i,j}$  is equal to the weight of the edge  $e_{i,j}$  between nodes  $i$  and  $j$  if  $v_i$  and  $v_j$  are connected or  $A_{i,j} = 0$  otherwise. The adjacency matrix of  $A$  with drop edge can be represented as

$$A_{\text{drop}} = A - A' \quad (10)$$

where  $V$  and  $p$  indicate the number of edges and dropping rate, respectively.  $A'$  stands for a sparse matrix expanded by a random subset of size  $V_p$  from  $E$ . In order to validate the effectiveness of DGCN-D for HSI classification, the detailed discussion of  $p$  is shown in Section III.

The updated node representations by the convolution in DGCN-D are defined as follows:

$$h_k^{c+1} = \sigma \left( \hat{A}_{\text{drop}} W^{(c)} \sum_{v \in N_k} h_v^c \right) \quad (11)$$

$$\hat{A}_{\text{drop}} = \hat{D}^{-1/2} (A_{\text{drop}} + I) \hat{D}^{-1/2} \quad (12)$$

where  $h_k^{c+1}$  is the feature of the node  $k$  at layer  $c + 1$ ,  $\hat{D}$  represents the corresponding degree matrix of  $A + I$ ,  $\hat{A}_{\text{drop}}$  indicates the renormalization of the adjacency matrix with drop edge, and  $W^{(l)}$  stands for the filter matrix at the  $c$ th layer.

### C. DGCN-M for HSI Classification

In order to address the overfitting issue and further improve the classification performance, a new regularization technique called multiscale feature cutout with DGCN (DGCN-M) is proposed, which is inspired by the cutout.

The cutout is first introduced by DeVriesl and Taylor [56]. The core idea of the cutout is to randomly drop out contiguous square regions of the input. These regions could propagate through the back-propagation algorithm. Inspired by the cutout, this article proposes a new regularization method to generalize the model well for HSI classification without increasing the complexity of the model. Algorithm 3 describes the whole process of DGCN-M for HSI classification.

Multiscale feature cutout improves model performance by producing multiscale zero masks of feature maps at each training time. Multiscale feature cutout is based on applying cutout to feature maps, which is named feature cutout with DGCN (DGCN-C). The difference between the cutout and multiscale feature cutout is that the proposed multiscale feature cutout focuses on the feature maps of the convolutional layer since the activation units in convolutional layers are spatially correlated and the former convolutional layer is important for the subsequent steps of feature extraction. Therefore, this article mainly focuses on the feature maps produced by the first convolutional layer. In addition, compared to another frequently used regularization method (i.e., dropout), the proposed regularization method randomly masks out a regular square from feature maps instead of individual pixels. Besides, compared to DropBlock [57], multiscale feature cutout produces feature maps with multiscale cutout region sizes, and it is easy to implement. The cutout region size is set to be more than half of the input that, sometimes, the region size

### Algorithm 3 DGCN-M for HSI Classification

#### input:

sample set  $I'$  and corresponding label set  $y$  of HSI.

#### output:

predict labels of test samples

1. **initialize** cyclic time  $L$ , the size of the feature cutout region  $r$ , and feature map cutting out rate  $m$
2. **Construct DGCN** (If  $L = 0$ ):
3. **initialize** the point graph ( $G_L^p$ ):
4. for each sample  $I'_i$  in the training set:
5. feed  $I'_i$  into CNN with learned parameters  $w$  and  $b$ .
6. for each convolutional layer in CNN:
7. randomly select feature maps with probability  $m$ .
8. for each selected feature
9. apply twice zero mask with size  $r$  to a random location
10. initialize node features  $V_L^p$  of  $G_L^p$  according to the output features of CNN from  $I'$ .
11. initialize edge features  $E_L^p$  of  $G_L^p$  among  $I'$ , for each edge feature  $e_{L,ij}^p$  between  $I'_i$  and  $I'_j$  in  $E_L^p$ :
12.  $e_{L,ij}^p = f_{e^p}((v_{L,i}^p - v_{L,j}^p)^2)$
13. **initialize** the distribution graph ( $G_L^d$ ):
14. initialize node features  $V_L^d$  of  $G_L^d$  according to the label distribution information among samples.
15. initialize edge features  $E_L^d$  of  $G_L^d$  according to the  $V_L^d$ , for each edge feature  $e_{L,ij}^d$  between  $I'_i$  and  $I'_j$  in  $E_L^d$ :
16.  $e_{L,ij}^d = f_{e^d}((v_{L,i}^d - v_{L,j}^d)^2)$
17. **Integrate  $G_L^p$  with  $G_L^d$**  (If  $L > 0$ ):
18. for each cyclic time  $L$ :
19. combine each node  $V_L^p$  in  $G_L^p$  with  $E_L^d$  in  $G_L^d$ .
20. compute each edge in  $E_L^p$  according to the previous  $E_{L-1}^p$  in  $G_L^p$ .
21. combine each node  $V_L^d$  in  $G_L^d$  with  $E_L^p$  in  $G_L^p$ .
22. compute each edge in  $E_L^d$  according to the previous  $E_{L-1}^d$  in  $G_L^d$ .
23. predict each node in  $V_L^p$  in  $G_L^p$ .

might not be fully contained inside the feature map. Thus, feature maps with various levels of region sizes are generated, and multiscale feature cutout further reduces the overfitting problem. Fig. 3 illustrates the differences between several methods intuitively (i.e., DGCN with DropBlock, DGCN-C, and DGCN-M).

There are three main parameters involved in multiscale feature cutout, including the cutout region size  $r$ , mask probability  $m$ , and the number of scales  $s$ . Especially, the region size  $r$  controls the shape of the zero masks of the input feature map.

The choice of  $m$  defines the number of feature maps on which cutout should be applied. The value of  $s$  is the number of scales. First, in the process of multiscale feature cutout,  $s$  pixel coordinates are randomly selected in the feature map. Second, the region size  $r$  is placed around each pixel coordinate location. Instead of applying cutout to each feature map, the mask probability  $m$  makes the model receive

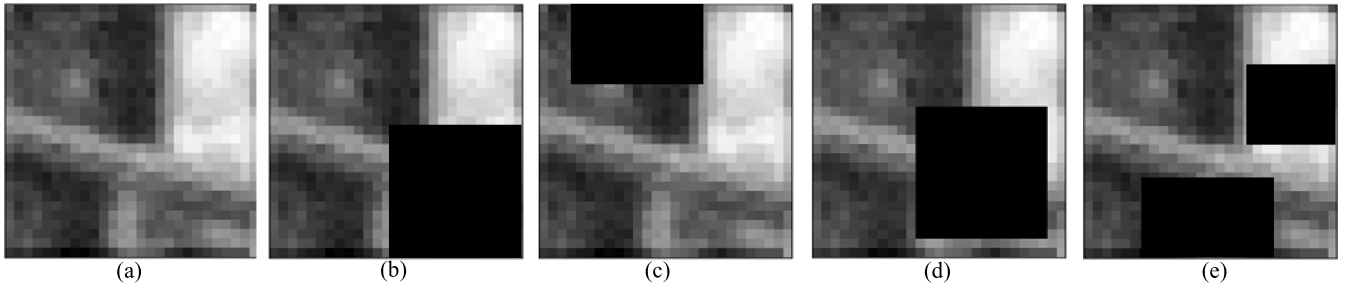


Fig. 3. Comparison of feature maps with different methods on the Indian Pines data set. (a) Original feature map. (b) Corresponding features map with Dropblock. (c) First sample of DGCN with feature cutout. (d) Second sample of DGCN with feature cutout. (e) Corresponding features map with multiscale feature cutout.

unmodified feature maps sometimes to make the model more robust. Detailed discussions on  $r$ ,  $m$ , and  $s$  can be found in Section III-C. These parameters can substantially influence the final classification performance of DGCN-M for HSI classification.

### III. EXPERIMENTS

In this section, the performance of the proposed methods is verified on the four benchmark data sets.

#### A. Experimental Data Description

The four widely used data sets, including the Salinas, Indian Pines, Pavia, and Houston data sets, are introduced in the following. The color composite image, ground-truth classification map, and detailed information on the number of each class are presented in Fig. 4.

1) *Salinas Data Set*: This HSI data set was captured by the AVIRIS sensor over Salinas Valley. It contains  $512 \times 217$  pixels, and the spatial resolution is 3.7 m. The 204 spectral bands are preserved in the range of 0.2–2.4  $\mu\text{m}$  after the removal of several spectral bands (108–112, 154–167, and 224) due to the noise effect. The ground truth contains 16 land-cover types.

2) *Indian Pines Data Set*: This data set was captured by the Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) in June 1992. The data set consists of  $145 \times 145$  pixels with 220 spectral bands in the wavelength ranging from 0.2 to 2.4  $\mu\text{m}$ . After removing water absorption and low signal-to-noise ratio bands (bands 104–108, 150–163, and 220), 200 bands are used for experiments. The corresponding ground truth contains 16 classes of interest.

3) *Pavia Data Set*: It was acquired by the Reflective Optics Spectrographic Image System (ROSIS-3) sensor, which consists of 115 spectral bands with wavelength ranging from 0.43 to 0.86  $\mu\text{m}$ . Its spatial resolution is 1.3 m, and its image size is  $610 \times 340$ . The 12 noisy spectral bands have been removed due to the low signal-to-noise ratio, leading to 103 bands. Nine classes of land covers are used in the experiments.

4) *Houston Data Set*: It was acquired on February 16, 2017, by the CASI-1500 over the area of the University of Houston, Houston, TX, USA, which is available from the 2018 IEEE GRSS Data Fusion Contest. The training portion of the data set consists of  $601 \times 2384$  pixels with 50 bands in the range

TABLE I  
ARCHITECTURE OF THE CONVOLUTION NEURAL NETWORK

Order	Input	Conv. Output	Pad	BN	Leaky-ReLU	Pool Output	FC Output
1	$27 \times 27 \times n\text{Band}$	$3 \times 3$ $27 \times 27 \times 128$	1	Yes	0.2	$2 \times 2$ $13 \times 13 \times 128$	-
1→2	$13 \times 13 \times 128$	$3 \times 3$ $11 \times 11 \times 192$	0	Yes	0.2	$2 \times 2$ $5 \times 5 \times 192$	-
2→3	$5 \times 5 \times 192$	$3 \times 3$ $5 \times 5 \times 256$	1	Yes	0.2	$2 \times 2$ $2 \times 2 \times 256$	-
3→4	$2 \times 2 \times 256$	-	-	-	-	$2 \times 2$ $1 \times 1 \times 256$	-
4→5	$1 \times 1 \times 256$	-	-	Yes	-	-	$1 \times 128$
3→6	$2 \times 2 \times 256$	$3 \times 3$ $2 \times 2 \times 512$	1	Yes	0.2	$2 \times 2$ $1 \times 1 \times 512$	-
6→7	$1 \times 1 \times 512$	-	-	Yes	-	-	$1 \times 128$

of 0.38–1.05  $\mu\text{m}$  and a ground-sampling distance of 1 m. The ground truth contains 20 land cover types.

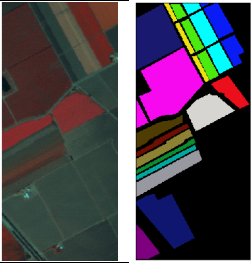

#### B. Experimental Design

In the experiments, for all the data sets, the input data are normalized into  $[-0.5, 0.5]$ . The  $27 \times 27$  neighbors of each pixel are used as the input to the models. In addition, the training samples are randomly chosen from all labeled samples (i.e., five, six, and seven training samples per each class).

To evaluate the effectiveness of the proposed method DGCN, several existing methods are used for comparisons, including SVM with EMPs (EMPs-SVM) [8], CNN, CNN with EMPs (EMP-CNN) [58], hybrid spectral CNN (HySN) [59], and GCN [48]. Besides, since this article focuses on exploiting the issue of a limited number of training samples for HSI classification, Siamese CNN (SCNN), which is proven to be an effective approach to handle such an issue, is also considered in the experiments [60].

The CNN used in the proposed network contains four stacked blocks; each block consists of a  $3 \times 3$  convolutional layer, BN layer, Leaky-ReLU operation with the size of 0.2, and max-pooling layer. The third and fourth blocks are followed by a BN layer and an FC layer. The output of the first FC layer is used to initialize edge features in the point graph, and the second FC layer is used to initialize the node features in the point graph. The architecture of CNN is shown in Table I.



Salinas				Indian Pines			
							
Class		Sample		Class		Sample	
No.	Color	Name	Number	No.	Color	Name	Number
1		Brocoli green weeds 1	1977	1		Alfalfa	46
2		Brocoli green weeds 2	3726	2		Corn-notill	1428
3		Fallow	1976	3		Corn-min	830
4		Fallow_rough_plow	1394	4		Corn	237
5		Fallow_smooth	2678	5		Grass-pasture	483
6		Stubble	3959	6		Grass-trees	730
7		Celery	3579	7		Grass-pasture-mowed	28
8		Grapes_untrained	11213	8		Hay-windrowed	478
9		Soil_vinyard_develop	6197	9		Oats	20
10		Corn_senesced_green_weeds	3249	10		Soybean-notill	972
11		Lettuce_roumaine_4wk	1058	11		Soybean-mintill	2455
12		Lettuce_roumaine_5wk	1908	12		Soybean-clean	593
13		Lettuce_roumaine_6wk	909	13		Wheat	205
14		Lettuce_roumaine_7wk	1061	14		Woods	1265
15		Vinyard_untrained	7164	15		Buildings-Grass-Trees	386
16		Vinyard_vertical_trellis	1737	16		Stone-Steel-Towers	93
Total		53785		Total		10249	

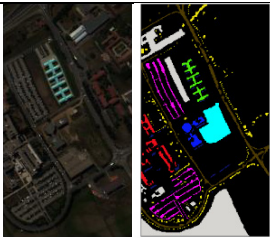
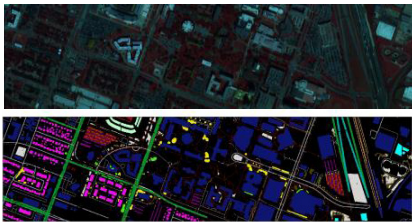
Pavia				Houston			
							
Class		Sample		Class		Sample	
No.	Color	Name	Number	No.	Color	Name	Number
1		Asphalt	6852	1		Healthy grass	9799
2		Meadows	18686	2		Stressed grass	32502
3		Gravel	2207	3		Synthetic grass	684
4		Trees	3436	4		Evergreen Trees	13588
5		Metal sheets	1378	5		Deciduous Trees	5048
6		Bare soil	5104	6		Soil	4516
7		Bitumen	1356	7		Water	266
8		Bricks	3878	8		Residential	39762
9		Shadow	1026	9		Commercial	223684
				10		Road	45810
				11		Sidewalk	34002
				12		Crosswalk	1516
				13		Major Thoroughfares	46358
				14		Highway	9849
				15		Railway	6937
				16		Paved Parking Lot	11475
				17		Gravel Parking Lot	149
				18		Cars	6578
				19		Trains	5365
				20		Seats	6824
Total		43923		Total		504712	

Fig. 4. Color composite image, ground truth, and detailed information on the number of classes for four publicly available data sets (Salinas, Indian Pines, Pavia, and Houston).

During the training procedure, the total number of iterations is set to 500, 500, 300, and 500 for the Salinas, Indian Pines, Pavia, and Houston data sets, respectively. Besides, the method of minibatch was adopted, which is set to 5, 5, 10, and 5

TABLE II

PARAMETERS OF DIFFERENT METHODS TAKEN ON THE FOUR DATA SETS

Datasets Methods	Parameters	Salinas	Indian Pines	Pavia	Houston
DGCN	Learning rate	0.001	0.001	0.001	0.001
	Mini-batch	5	5	10	5
	Iteration epochs	500	500	300	500
	$L$	6	6	6	6
DGCN-D	Learning rate	0.001	0.001	0.001	0.001
	Mini-batch	5	5	10	5
	Iteration epochs	400	300	200	500
	$L$	6	6	6	6
	Dropping rate	0.2	0.5	0.3	0.7
DGCN-M	Learning rate	0.001	0.001	0.001	0.005
	Mini-batch	5	5	10	5
	Iteration epochs	600	500	500	600
	$L$	6	6	6	6
	Cut region	18	15	15	15
	Mask probability	0.5	0.7	0.5	0.5

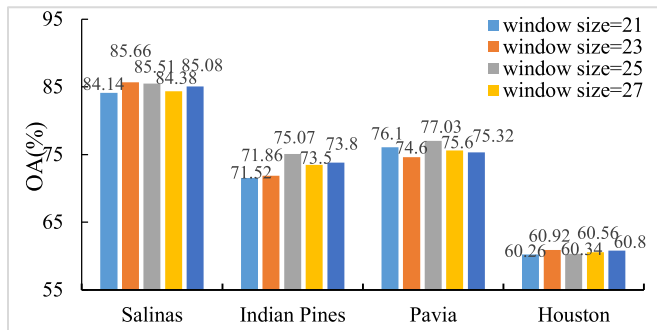


Fig. 5. Results of DGCN with different window sizes.

for the Salinas, Indian Pines, Pavia, and Houston data sets, respectively. The detailed information of parameters is shown in Table II.

To quantitatively evaluate all the methods, overall accuracy (OA), average accuracy (AA), kappa coefficient ( $K$ ), and class-specific accuracy are utilized as the evaluation metrics.

### C. Parameter Settings

There are several important parameters that influence the final classification accuracies. The parameters include the drop edges with probability  $p$  for DGCN-D and the cutout size  $r$  and mask probability  $m$  for the DGCN-M. For other parameters, such as the window size, we only provide empirical settings.

1) *Influence of Window Size*: To validate the influences of DGCN with different spatial window sizes (i.e.,  $21 \times 21$ ,  $23 \times 23$ ,  $25 \times 25$ ,  $27 \times 27$ , and  $29 \times 29$ ), the results of DGCN on the four data sets are shown in Fig. 5. As shown in Fig. 5, DGCN obtains the best performance concerning the  $23 \times 23$  window size on the Salinas and Houston data sets and  $25 \times 25$  window size on the Indian Pines and Pavia data sets. In the experiments, to give a fair comparison with other methods, the window size is set to  $27 \times 27$  for all the experiments, which follows the same settings described in [58].

2) *Analysis of Dropping Rates  $p$* : For DGCN-D, the probability  $p$  controls the certain rate of edges in the graph for each iteration. The performance sensitivity of DGCN-D with

respect to different values of probability  $p$  is investigated. The grid search strategy is chosen to find the optimal value of  $p$ , which varies from 0.1 to 0.9. The related experimental results are provided in Fig. 6 for all four data sets with five training samples per each class. Compared to DGCN, it can be observed that DGCN-D consistently promotes the performance with small values of  $p$  on all the data sets, but, when the value of  $p$  reaches a certain value, it will decrease the final testing accuracy. For example, the values of  $p$  ranging from 0.1 to 0.3 achieve better performance on the Salinas data set compared to DGCN. It proves that DGCN-D can alleviate the risk of overfitting problems with the proper value. According to Fig. 6, with reference to the OA results, the value of  $p$  is set to 0.2, 0.5, 0.3, and 0.7 for the Salinas, Indian Pines, Pavia, and Houston data sets, respectively.

3) *Analysis of the Cutout Region  $r$ , Mask Probability  $m$ , and Number of Scales  $s$* : Experiments are conducted on the four data sets, and multiscale feature cutout is applied after the first convolutional layer. Here, there are three important hyperparameters in DGCN-M, including cutout region size  $r$ , mask probability  $m$ , and number of scales  $s$ . Especially,  $r$  defines how many square regions of the feature maps should randomly be masked out and set to 0 during training. Here, the square patches (i.e.,  $3 \times 3$ ,  $6 \times 6$ ,  $9 \times 9$ ,  $15 \times 15$ ,  $18 \times 18$ , and  $24 \times 24$ ) are selected to search for the best region  $r$  for DGCN-M. Besides,  $m$  is the probability, which defines the number of feature maps on which cutout should be applied. The value of  $m$  is selected from  $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ . A larger value of  $r$  implies that more locations in the feature maps are set to 0. In addition, the value of  $s$  is the number of scales. In the experiment, to analyze the influences of  $r$  and  $m$ ,  $s$  is fixed. All the OA results as a function of  $r$  and  $m$  with different data sets are shown in Fig. 7. For the case of  $s$ , the values of  $r$  and  $m$  are set to the optimal values to verify the influence of the number of  $s$ . Fig. 8 displays the classification results with respect to different values of  $s$  with 200 training samples for all the data sets. From Fig. 7, it can be seen that OA follows a parabolic trend; when the value of  $r$  is about half of the input, accuracy reaches an optimal point, after which OA decreases with larger values of  $r$  and  $m$ . It indicates that the proper values of  $r$  and  $m$  are important for HSI classification. Therefore,  $r$  and  $m$  are set to the optimal value for different data sets (i.e., the values of  $r$  and  $m$  are 18 and 0.5 for the Salinas data set,  $r$  is 15 and  $m$  is 0.7 for the Indian Pines data set,  $r$  is set to 15 and  $m$  is set to 0.5 for the Pavia data set, and  $r$  is set to 15 and  $m$  is set to 0.5 for the Houston data set). From Fig. 8, the multiscale feature cutout obtains the best result when the value of  $s$  is set to two for all the data sets. Thus, the value of  $s$  is set to two in the experiments to achieve the best performance. It shows that the proposed DGCN-M can address the overfitting problem while preserving the power of the model, leading to better classification results.

### D. Experimental Results and Analysis

1) *Comparison of DGCN With the State of the Arts*: The proposed DGCN is compared with a traditional supervised method: EMP combined with SVM and five deep

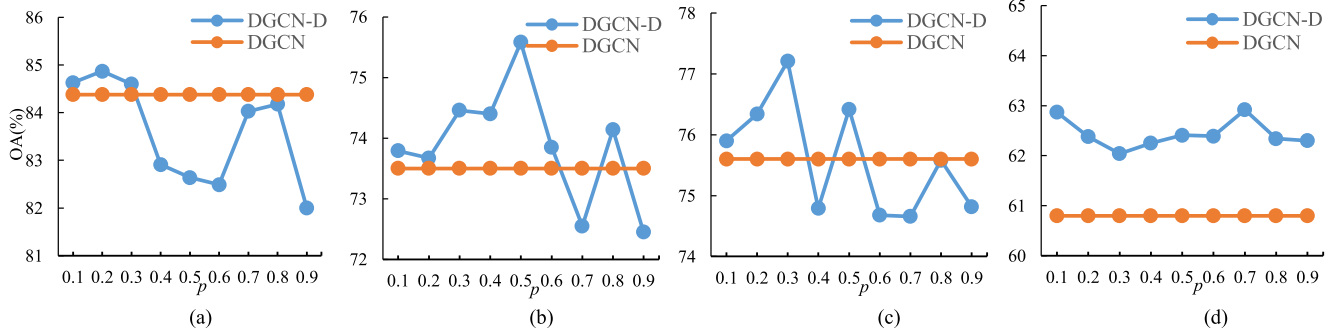


Fig. 6. Impact of dropping rates  $p$  in DGCN-D on the four data sets.

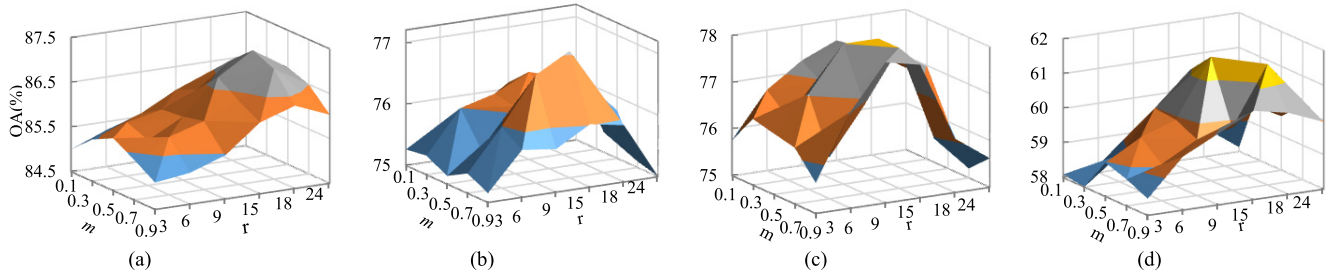


Fig. 7. Impacts of cutout region  $r$  and mask probability  $m$  on the performance of DGCN-M using four data sets.

learning-based methods: the regular CNN, EMP-CNN, SCNN, HySN, and GCN.

Here, the shape of the structuring element is set as a disk with an increasing size from two to eight for the EMP-SVM. The best parameters  $C$  and gamma are obtained using fivefold cross validation. To give fair comparisons, the architectures and parameters are set similar to the parameters for the regular CNN [22]. Besides, EMP with CNN and SCNN are also adopted for spectral-spatial classification. Specifically, the architecture design of EMP-CNN is similar to CNN. SCNN follows the same settings described in [61]. In addition, the latest methods, including HySN and GCN for HSI classification both published in 2020, are also considered for comprehensive comparison [48], [59]. The above methods have been proven to be effective when dealing with a limited number of training samples for HSI classification.

The results of all experiments are reported in Tables III–VI. All the experiments are the average values of ten runs with respect to different random initializations. It can be observed that the proposed DGCN is superior to other existing methods by having five training samples per each class for all three data sets. Especially, compared to EMP-SVM, DGCN improves the classification accuracy by 5.7% on the Indian Pines data set. Besides, CNN works worse than EMP-SVM due to a lack of training samples. In addition, for the Indian Pines data set, compared to EMP-CNN, DGCN achieves about 6% improvements in terms of OA and  $K$ , which means that DGCN is an effective technique in capturing the relationships among samples. For SCNN, when the number of training samples of each class is 5, it cannot improve the classification performance. Compared to SCNN, DGCN increases the OA by 4.21%, 19.27%, 6.59%, and 8.13% on the Salinas, Pavia, Indian Pines, and Houston data sets, respectively. Besides, DGCN offers

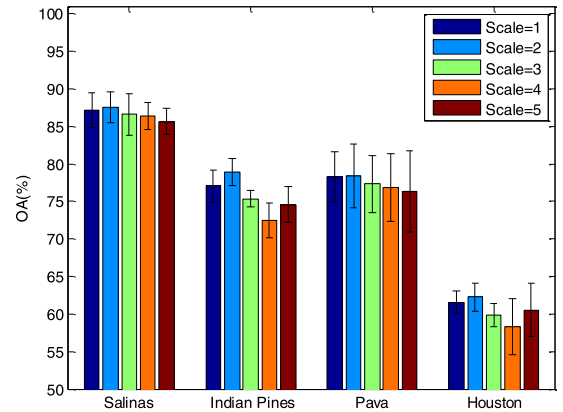


Fig. 8. Analysis of the influence of the number of scales  $s$  of DGCN-M.

significant improvement for the Indian Pines and Salinas data sets with only five training samples per each class compared with HySN. In addition, DGCN outperforms GCN on the four data sets with limited training samples. For example, on average, DGCN is significantly higher. For the Salinas data set, slightly better performance is also achieved for DGCN. Moreover, DGCN also achieves better results with more training samples, as shown in Fig. 9. All the results demonstrate that DGCN is an effective strategy in graph learning for HSI classification.

2) *Visualization of Features Obtained by Different Methods*: To clearly illustrate the effectiveness of DGCN, Figs. 10–13 show the t-SNE visualizations of different comparative methods for the four data sets [62], which can clearly demonstrate the distributions of output features obtained by different methods. Here, different colors mean different kinds of labels. More specifically, it can be seen that all class

TABLE III  
CLASSIFICATION RESULTS (VALUES  $\pm$  STANDARD DEVIATION) ON THE SALINAS DATA SET USING FIVE TRAINING SAMPLES PER EACH CLASS

Method	EMP-SVM	CNN	EMP-CNN	SCNN	HySN	GCN	DGCN	DGCN-D	DGCN-C	DGCN-M	DGCN-DC
OA(%)	83.95 $\pm$ 1.82	78.48 $\pm$ 2.22	83.76 $\pm$ 2.31	80.17 $\pm$ 4.67	86.54 $\pm$ 3.06	83.60 $\pm$ 1.36	84.38 $\pm$ 2.54	84.87 $\pm$ 1.86	87.17 $\pm$ 2.31	<b>87.52<math>\pm</math>2.07</b>	87.33 $\pm$ 2.83
AA(%)	84.04 $\pm$ 1.81	79.61 $\pm$ 2.60	84.63 $\pm$ 2.37	81.31 $\pm$ 4.03	87.47 $\pm$ 3.50	84.17 $\pm$ 0.71	85.23 $\pm$ 1.95	85.56 $\pm$ 1.87	88.13 $\pm$ 2.30	87.33 $\pm$ 2.20	<b>88.23<math>\pm</math>1.69</b>
$K \times 100$	82.88 $\pm$ 1.94	76.97 $\pm$ 2.36	82.63 $\pm$ 2.48	70.37 $\pm$ 4.46	85.57 $\pm$ 3.28	82.45 $\pm$ 1.45	84.24 $\pm$ 2.08	84.55 $\pm$ 1.99	<b>86.31<math>\pm</math>2.46</b>	<b>86.69<math>\pm</math>2.21</b>	86.49 $\pm$ 3.02
Brocoli_green_weeds_1	<b>93.64<math>\pm</math>3.53</b>	49.92 $\pm$ 14.12	80.28 $\pm$ 14.78	87.40 $\pm$ 0.86	<b>98.76<math>\pm</math>1.70</b>	90.73 $\pm$ 5.28	83.34 $\pm$ 2.38	85.00 $\pm$ 1.29	89.30 $\pm$ 2.49	87.65 $\pm$ 2.56	87.73 $\pm$ 1.92
Brocoli_green_weeds_2	89.84 $\pm$ 7.16	85.21 $\pm$ 8.76	91.13 $\pm$ 9.17	79.25 $\pm$ 7.63	<b>98.09<math>\pm</math>1.52</b>	95.24 $\pm$ 14.07	84.58 $\pm$ 2.42	83.90 $\pm$ 3.16	90.30 $\pm$ 2.65	87.10 $\pm$ 1.79	88.90 $\pm$ 2.60
Fallow	64.84 $\pm$ 9.89	85.46 $\pm$ 13.45	81.03 $\pm$ 15.29	81.98 $\pm$ 3.77	87.08 $\pm$ 13.58	55.76 $\pm$ 3.12	84.92 $\pm$ 2.14	88.20 $\pm$ 2.46	87.80 $\pm$ 1.57	87.05 $\pm$ 2.68	<b>88.65<math>\pm</math>1.24</b>
Fallow_rough_plow	<b>99.79<math>\pm</math>0.12</b>	97.41 $\pm$ 2.31	97.12 $\pm$ 1.21	72.17 $\pm$ 19.52	84.19 $\pm$ 21.99	98.06 $\pm$ 0.45	86.10 $\pm$ 3.00	85.70 $\pm$ 1.89	87.20 $\pm$ 2.62	88.38 $\pm$ 2.26	89.52 $\pm$ 1.76
Fallow_smooth	93.43 $\pm$ 7.40	92.71 $\pm$ 4.10	87.98 $\pm$ 8.85	<b>95.26<math>\pm</math>0.88</b>	73.45 $\pm$ 15.94	91.03 $\pm$ 12.13	84.28 $\pm$ 2.89	84.30 $\pm$ 1.61	89.30 $\pm$ 2.34	86.28 $\pm$ 2.09	87.87 $\pm$ 1.92
Stubble	99.17 $\pm$ 0.67	98.79 $\pm$ 1.60	91.14 $\pm$ 7.26	<b>99.54<math>\pm</math>0.52</b>	95.38 $\pm$ 6.03	99.50 $\pm$ 0.53	83.32 $\pm$ 3.84	85.90 $\pm$ 2.32	87.80 $\pm$ 3.38	86.85 $\pm$ 2.05	87.37 $\pm$ 1.59
Celery	<b>98.03<math>\pm</math>1.18</b>	92.94 $\pm$ 2.15	90.44 $\pm$ 8.56	84.46 $\pm$ 0.59	92.24 $\pm$ 9.49	89.77 $\pm$ 2.48	84.50 $\pm$ 2.55	86.20 $\pm$ 2.33	87.60 $\pm$ 2.34	86.80 $\pm$ 2.22	88.38 $\pm$ 1.62
Grapes_untrained	66.86 $\pm$ 28.12	73.84 $\pm$ 9.05	57.96 $\pm$ 9.04	33.99 $\pm$ 0.57	69.41 $\pm$ 10.20	39.15 $\pm$ 3.22	83.96 $\pm$ 3.30	84.00 $\pm$ 2.01	86.70 $\pm$ 2.98	87.23 $\pm$ 2.21	<b>87.32<math>\pm</math>1.59</b>
Soil_vinyard_develop	<b>98.84<math>\pm</math>0.56</b>	95.81 $\pm$ 5.36	97.86 $\pm$ 1.16	87.61 $\pm$ 3.66	92.97 $\pm$ 8.48	98.40 $\pm$ 0.25	85.16 $\pm$ 2.45	85.40 $\pm$ 1.88	87.90 $\pm$ 2.55	87.65 $\pm$ 2.73	89.13 $\pm$ 1.14
Corn_senesced_green_weeds	26.27 $\pm$ 23.51	73.56 $\pm$ 12.84	76.59 $\pm$ 8.08	61.59 $\pm$ 16.98	78.43 $\pm$ 12.25	82.91 $\pm$ 12.06	84.20 $\pm$ 3.30	86.20 $\pm$ 2.11	88.50 $\pm$ 2.86	<b>88.70<math>\pm</math>2.18</b>	88.48 $\pm$ 2.44
Lettuce_roumaine_4wk	93.76 $\pm$ 3.56	81.61 $\pm$ 15.93	88.01 $\pm$ 8.60	<b>97.45<math>\pm</math>2.42</b>	92.76 $\pm$ 5.15	97.39 $\pm$ 7.07	84.24 $\pm$ 3.13	87.20 $\pm$ 1.74	87.00 $\pm$ 1.51	86.88 $\pm$ 2.18	87.51 $\pm$ 1.74
Lettuce_roumaine_5wk	<b>97.69<math>\pm</math>4.94</b>	88.01 $\pm$ 13.30	92.53 $\pm$ 8.41	79.95 $\pm$ 8.84	78.56 $\pm$ 12.07	90.27 $\pm$ 6.43	83.26 $\pm$ 3.01	84.00 $\pm$ 2.47	88.60 $\pm$ 3.01	86.87 $\pm$ 2.69	87.57 $\pm$ 2.18
Lettuce_roumaine_6wk	99.16 $\pm$ 0.27	93.07 $\pm$ 6.88	93.41 $\pm$ 3.42	97.13 $\pm$ 1.34	96.13 $\pm$ 2.95	<b>100.00<math>\pm</math>0.00</b>	83.65 $\pm$ 2.81	84.49 $\pm$ 2.85	85.48 $\pm$ 2.73	87.55 $\pm$ 2.51	87.52 $\pm$ 2.57
Lettuce_roumaine_7wk	<b>98.61<math>\pm</math>0.33</b>	95.61 $\pm$ 2.98	93.03 $\pm$ 2.78	85.46 $\pm$ 3.46	98.09 $\pm$ 2.50	69.20 $\pm$ 17.73	83.66 $\pm$ 3.64	87.50 $\pm$ 2.03	90.00 $\pm$ 3.04	86.78 $\pm$ 3.02	89.35 $\pm$ 1.93
Vinyard_untrained	57.71 $\pm$ 33.10	32.17 $\pm$ 16.03	69.26 $\pm$ 9.62	77.76 $\pm$ 10.44	78.29 $\pm$ 8.84	66.32 $\pm$ 0.36	84.76 $\pm$ 3.42	84.20 $\pm$ 2.35	<b>89.90<math>\pm</math>2.01</b>	88.27 $\pm$ 2.22	88.08 $\pm$ 2.23
Vinyard_vertical_trellis	66.99 $\pm$ 7.28	37.67 $\pm$ 7.97	66.23 $\pm$ 21.20	80.01 $\pm$ 10.93	85.70 $\pm$ 22.80	<b>93.01<math>\pm</math>2.94</b>	84.02 $\pm$ 2.91	86.60 $\pm$ 1.70	86.70 $\pm$ 1.84	87.23 $\pm$ 2.18	88.23 $\pm$ 1.76
Training Time (sec.)	6.95	17.72	3.48	1012.73	14.48	200.09	650.88	602.85	1367.29	844.43	1398.5
Test Time (sec.)	0.15	159.61	11.11	3300.67	146.63	21.94	238.37	218.06	262.81	199.44	195.87

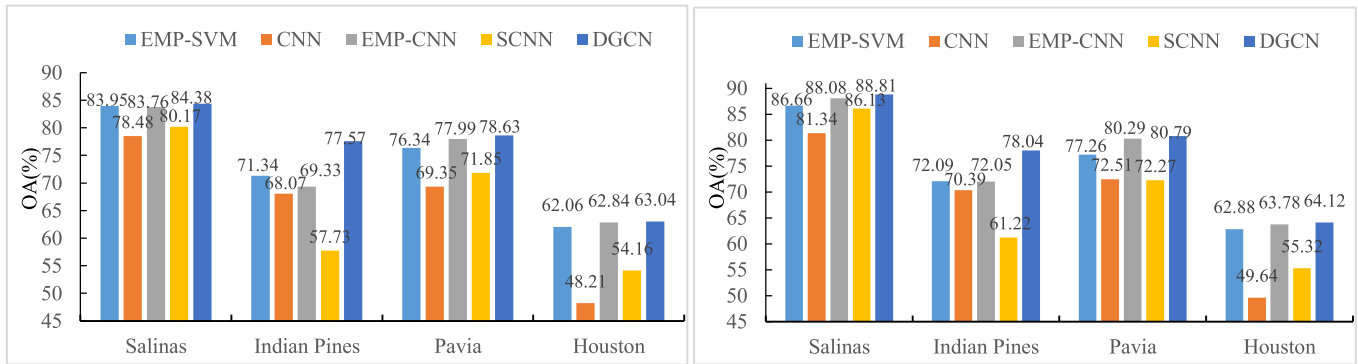


Fig. 9. Accuracy (%) comparisons under different methods on four HSI data sets using six and seven training samples per each class.

distributions of the original HSI are heavily mixed together on the Indian Pines, Pavia, and Houston data sets, while DGCN and DGCN-DC show a good clustering performance, which increases the between-class distances. Taking the Salinas data set as an example, as shown in Fig. 10, intuitively, the original HSI contains distinguishable clusters, but the within-class distance is large. For the CNN, EMP-CNN, and SCNN [see Fig. 10(b)–(d)], the intraclass distances become shorter, but class 15 (blue) and class 8 (light orange) are heavily mixed. Compared to other methods, for the proposed DGCN [see Fig. 10(e)], most samples in each class are clustered with smaller intraclass distances. Besides, class 15 (blue) and class 8 (light orange) are distinguished, which leads to better performance. For the proposed DGCN-DC [see Fig. 10(f)], compared to DGCN, intraclass distances are smaller. In addition, for the Pavia data set, as shown in Fig. 12(b), CNN works

better than the original HSI, but the test samples in different classes are mixed. Moreover, for EMP-CNN, as shown in Fig. 12(c), each class is clustered together, which means that EMP-CNN could increase the between-class distances. For the case of SCNN, the intraclass distance is too large to classify different samples, which causes worse performance. In addition, the results of the proposed DGCN are shown in Fig. 12(e) and (f), respectively. The distributions of output features show good clustering. Compared to other methods, DGCN and DGCN-DC not only increase between-class distances but also minimize intraclass distances among samples. Indian Pines has similar visual results. For the Houston data set (see Fig. 13), all class distributions of the competitive methods are completely mixed, while different kinds of labels gather together and come into several groups for the proposed DGCN and DGCN-DC. Intuitively, the visual results illustrate



TABLE IV  
CLASSIFICATION RESULTS (VALUES  $\pm$  STANDARD DEVIATION) ON THE INDIAN PINES DATA SET USING FIVE TRAINING SAMPLES PER EACH CLASS

Method	EMP-SVM	CNN	EMP-CNN	SCNN	HySN	GCN	DGCN	DGCN-D	DGCN-C	DGCN-M	DGCN-DC
OA(%)	67.80 $\pm$ 4.52	62.57 $\pm$ 7.57	66.80 $\pm$ 7.63	54.23 $\pm$ 2.90	63.60 $\pm$ 6.08	43.36 $\pm$ 0.45	73.50 $\pm$ 5.00	75.58 $\pm$ 2.58	77.06 $\pm$ 2.09	<b>78.92<math>\pm</math>1.83</b>	77.46 $\pm$ 2.89
AA(%)	<b>79.51<math>\pm</math>2.93</b>	78.17 $\pm$ 2.38	81.40 $\pm$ 2.40	59.37 $\pm$ 2.18	76.69 $\pm$ 1.74	51.37 $\pm$ 1.17	70.70 $\pm$ 5.43	74.78 $\pm$ 1.95	75.83 $\pm$ 1.88	79.34 $\pm$ 1.80	77.79 $\pm$ 2.33
$K \times 100$	63.06 $\pm$ 4.93	57.47 $\pm$ 7.75	62.34 $\pm$ 7.84	49.96 $\pm$ 3.09	57.81 $\pm$ 5.60	37.04 $\pm$ 0.11	67.81 $\pm$ 6.39	71.72 $\pm$ 2.80	74.65 $\pm$ 2.11	76.68 $\pm$ 2.02	<b>77.63<math>\pm</math>3.14</b>
Alfalfa	93.73 $\pm$ 4.42	97.15 $\pm$ 3.59	98.70 $\pm$ 2.04	<b>100.00<math>\pm</math>0.00</b>	99.30 $\pm$ 1.84	43.90 $\pm$ 27.59	72.83 $\pm$ 5.76	78.05 $\pm$ 5.60	80.43 $\pm$ 5.86	79.81 $\pm$ 3.62	81.10 $\pm$ 2.59
Corn-notill	39.66 $\pm$ 11.92	37.16 $\pm$ 14.91	46.30 $\pm$ 16.29	14.49 $\pm$ 11.29	38.12 $\pm$ 9.92	38.16 $\pm$ 2.69	72.34 $\pm$ 9.11	71.50 $\pm$ 3.20	78.01 $\pm$ 2.89	<b>80.76<math>\pm</math>1.99</b>	76.20 $\pm$ 2.25
Corn-min	66.67 $\pm$ 14.25	52.95 $\pm$ 13.47	60.97 $\pm$ 14.87	38.90 $\pm$ 12.48	60.13 $\pm$ 11.46	25.09 $\pm$ 0.59	69.88 $\pm$ 8.34	72.85 $\pm$ 2.94	73.61 $\pm$ 2.51	<b>77.90<math>\pm</math>2.96</b>	73.82 $\pm$ 0.50
Corn	81.68 $\pm$ 22.30	94.12 $\pm$ 10.52	99.61 $\pm$ 0.96	43.75 $\pm$ 3.79	<b>82.57<math>\pm</math>12.67</b>	39.22 $\pm$ 10.66	68.14 $\pm$ 15.81	68.53 $\pm$ 4.42	71.31 $\pm$ 3.72	79.69 $\pm$ 0.48	75.50 $\pm$ 1.73
Grass-pasture	65.76 $\pm$ 20.10	53.43 $\pm$ 11.23	64.36 $\pm$ 20.08	46.75 $\pm$ 16.58	67.35 $\pm$ 3.89	18.41 $\pm$ 28.26	66.98 $\pm$ 3.08	71.76 $\pm$ 2.63	71.84 $\pm$ 2.92	<b>79.71<math>\pm</math>2.79</b>	73.83 $\pm$ 0.27
Grass-trees	80.57 $\pm$ 14.94	65.63 $\pm$ 15.48	63.61 $\pm$ 18.23	55.92 $\pm$ 10.40	<b>90.42<math>\pm</math>3.25</b>	74.21 $\pm$ 5.85	64.52 $\pm$ 15.70	72.14 $\pm$ 2.77	74.52 $\pm$ 3.14	78.55 $\pm$ 2.56	78.46 $\pm$ 3.93
Grass-pasture-mowed	<b>100.00<math>\pm</math>0.00</b>	<b>100.00<math>\pm</math>0.00</b>	<b>100.00<math>\pm</math>0.00</b>	78.26 $\pm$ 15.37	97.52 $\pm$ 6.57	86.96 $\pm$ 3.07	67.86 $\pm$ 5.00	73.91 $\pm$ 4.03	71.43 $\pm$ 4.56	84.18 $\pm$ 5.00	80.80 $\pm$ 2.56
Hay-windrowed	86.44 $\pm$ 10.23	97.82 $\pm$ 1.54	96.85 $\pm$ 3.99	<b>99.12<math>\pm</math>8.04</b>	91.08 $\pm$ 12.53	68.08 $\pm$ 16.00	70.61 $\pm$ 10.50	77.38 $\pm$ 3.10	76.57 $\pm$ 2.90	78.36 $\pm$ 3.53	81.75 $\pm$ 7.57
Oats	<b>100.00<math>\pm</math>0.00</b>	92.22 $\pm$ 13.61	95.56 $\pm$ 6.88	<b>100.00<math>\pm</math>0.00</b>	98.10 $\pm$ 5.04	53.33 $\pm$ 14.14	70.00 $\pm$ 14.14	66.67 $\pm$ 7.50	75.00 $\pm$ 5.62	83.57 $\pm$ 5.15	65.56 $\pm$ 7.85
Soybean-notill	70.43 $\pm$ 11.46	68.68 $\pm$ 21.34	66.91 $\pm$ 22.02	31.69 $\pm$ 10.91	77.42 $\pm$ 5.09	37.95 $\pm$ 7.53	72.64 $\pm$ 9.75	73.84 $\pm$ 2.86	77.47 $\pm$ 2.91	<b>78.01<math>\pm</math>2.25</b>	77.16 $\pm$ 3.34
Soybean-mintill	63.19 $\pm$ 13.17	53.82 $\pm$ 23.64	58.88 $\pm$ 27.02	51.25 $\pm$ 11.07	45.90 $\pm$ 18.58	23.21 $\pm$ 0.42	72.90 $\pm$ 4.85	76.60 $\pm$ 2.65	75.67 $\pm$ 2.06	78.32 $\pm$ 1.77	<b>78.44<math>\pm</math>5.14</b>
Soybean-clean	63.31 $\pm$ 10.95	54.85 $\pm$ 12.10	68.85 $\pm$ 6.18	31.96 $\pm$ 6.60	46.88 $\pm$ 3.68	15.48 $\pm$ 3.49	73.53 $\pm$ 4.53	78.06 $\pm$ 3.24	78.08 $\pm$ 1.90	79.45 $\pm$ 2.31	<b>81.07<math>\pm</math>5.37</b>
Wheat	95.57 $\pm$ 0.98	97.50 $\pm$ 3.63	94.92 $\pm$ 5.23	65.29 $\pm$ 2.20	97.71 $\pm$ 4.25	<b>96.50<math>\pm</math>1.41</b>	76.10 $\pm$ 2.07	77.00 $\pm$ 2.72	72.20 $\pm$ 3.64	76.03 $\pm$ 1.94	79.34 $\pm$ 2.35
Woods	88.85 $\pm$ 5.74	90.88 $\pm$ 3.37	<b>91.36<math>\pm</math>3.00</b>	66.97 $\pm$ 6.28	76.05 $\pm$ 9.80	86.27 $\pm$ 17.34	68.78 $\pm$ 5.95	73.90 $\pm$ 3.28	75.10 $\pm$ 1.72	79.56 $\pm$ 2.34	76.81 $\pm$ 0.24
Buildings-Grass-Trees	<b>96.88<math>\pm</math>2.95</b>	96.36 $\pm$ 1.84	96.74 $\pm$ 2.76	44.57 $\pm$ 3.46	79.79 $\pm$ 9.52	25.46 $\pm$ 5.38	71.50 $\pm$ 8.80	74.54 $\pm$ 1.82	71.76 $\pm$ 3.68	76.79 $\pm$ 2.40	79.00 $\pm$ 2.23
Stone-Steel-Towers	97.14 $\pm$ 2.43	98.24 $\pm$ 2.07	<b>98.63<math>\pm</math>1.89</b>	78.83 $\pm$ 1.66	78.73 $\pm$ 18.99	89.77 $\pm$ 4.02	72.58 $\pm$ 9.89	89.77 $\pm$ 5.39	90.32 $\pm$ 3.65	78.65 $\pm$ 3.01	85.80 $\pm$ 5.62
Training Time (sec.)	4.87	6.54	4.13	1462.84	19.06	80.75	699.76	632.87	1336.95	1092.4	1084.31
Test Time (sec.)	0.05	0.07	4.06	1454.67	43.02	18.08	250.70	230.59	263.79	269.71	262.20

TABLE V  
CLASSIFICATION RESULTS (VALUES  $\pm$  STANDARD DEVIATION) ON THE PAVIA DATA SET USING FIVE TRAINING SAMPLES PER EACH CLASS

Method	EMP-SVM	CNN	EMP-CNN	SCNN	HySN	GCN	DGCN	DGCN-D	DGCN-C	DGCN-M	DGCN-DC
OA(%)	74.71 $\pm$ 3.39	69.42 $\pm$ 2.41	74.01 $\pm$ 4.51	69.01 $\pm$ 6.11	65.76 $\pm$ 4.32	65.12 $\pm$ 2.73	75.60 $\pm$ 6.00	77.21 $\pm$ 4.49	78.28 $\pm$ 3.36	78.40 $\pm$ 4.24	<b>79.87<math>\pm</math>5.74</b>
AA(%)	77.22 $\pm$ 2.91	71.91 $\pm$ 2.29	75.75 $\pm$ 4.08	68.89 $\pm$ 5.32	65.99 $\pm$ 3.91	67.51 $\pm$ 1.85	77.05 $\pm$ 5.98	79.80 $\pm$ 4.47	78.46 $\pm$ 3.35	78.43 $\pm$ 4.73	<b>79.84<math>\pm</math>5.81</b>
$K \times 100$	71.45 $\pm$ 3.81	65.47 $\pm$ 2.72	70.67 $\pm$ 5.07	65.21 $\pm$ 8.08	61.26 $\pm$ 4.86	60.56 $\pm$ 3.03	72.75 $\pm$ 6.77	74.26 $\pm$ 5.07	75.46 $\pm$ 3.77	75.61 $\pm$ 4.78	<b>77.09<math>\pm</math>6.52</b>
Asphalt	71.87 $\pm$ 10.58	56.50 $\pm$ 5.91	54.38 $\pm$ 9.78	65.94 $\pm$ 5.05	37.43 $\pm$ 20.27	60.15 $\pm$ 13.79	76.10 $\pm$ 5.50	<b>79.65<math>\pm</math>4.22</b>	77.15 $\pm$ 2.87	78.16 $\pm$ 4.04	78.99 $\pm$ 3.62
Meadows	65.36 $\pm$ 21.95	69.71 $\pm$ 12.50	77.65 $\pm$ 12.66	<b>81.62<math>\pm</math>3.23</b>	74.10 $\pm$ 13.18	27.22 $\pm$ 9.14	76.70 $\pm$ 7.11	78.90 $\pm$ 5.05	76.85 $\pm$ 3.89	77.66 $\pm$ 5.18	80.91 $\pm$ 4.72
Gravel	49.25 $\pm$ 11.06	39.51 $\pm$ 10.88	39.88 $\pm$ 23.29	28.98 $\pm$ 12.78	<b>82.86<math>\pm</math>18.29</b>	50.48 $\pm$ 21.65	77.40 $\pm$ 5.91	80.90 $\pm$ 4.45	79.25 $\pm$ 2.96	79.12 $\pm$ 4.63	79.43 $\pm$ 4.02
Trees	<b>89.87<math>\pm</math>10.27</b>	89.61 $\pm$ 5.61	77.76 $\pm$ 16.43	73.26 $\pm$ 5.18	60.18 $\pm$ 18.73	80.35 $\pm$ 10.18	76.50 $\pm$ 5.79	79.60 $\pm$ 4.47	77.85 $\pm$ 2.70	78.15 $\pm$ 4.82	79.23 $\pm$ 4.16
Metal sheets	<b>98.18<math>\pm</math>1.04</b>	90.93 $\pm$ 5.41	92.77 $\pm$ 4.22	75.20 $\pm$ 2.41	94.81 $\pm$ 7.56	96.87 $\pm$ 1.39	78.74 $\pm$ 6.60	80.33 $\pm$ 4.85	77.58 $\pm$ 4.03	78.42 $\pm$ 5.17	80.79 $\pm$ 4.07
Bare soil	77.37 $\pm$ 21.03	58.81 $\pm$ 6.28	79.35 $\pm$ 11.31	73.99 $\pm$ 4.07	71.17 $\pm$ 26.24	<b>96.69<math>\pm</math>29.03</b>	77.35 $\pm$ 5.88	80.15 $\pm$ 4.20	79.65 $\pm$ 3.56	79.11 $\pm$ 4.88	81.03 $\pm$ 3.55
Bitumen	88.82 $\pm$ 6.23	76.61 $\pm$ 9.90	87.16 $\pm$ 4.73	77.86 $\pm$ 8.61	<b>97.57<math>\pm</math>2.85</b>	49.37 $\pm$ 13.71	76.25 $\pm$ 5.94	78.54 $\pm$ 4.67	80.16 $\pm$ 3.59	78.70 $\pm$ 5.12	80.60 $\pm$ 4.41
Bricks	75.26 $\pm$ 12.57	68.78 $\pm$ 8.60	<b>87.46<math>\pm</math>4.81</b>	61.76 $\pm$ 2.45	49.73 $\pm$ 22.57	47.22 $\pm$ 29.92	75.70 $\pm$ 6.20	80.75 $\pm$ 4.72	79.55 $\pm$ 4.32	78.28 $\pm$ 5.18	79.30 $\pm$ 4.16
Shadow	97.73 $\pm$ 1.47	96.72 $\pm$ 1.40	85.33 $\pm$ 9.71	81.37 $\pm$ 7.27	26.10 $\pm$ 14.59	<b>99.22<math>\pm</math>1.01</b>	78.75 $\pm$ 5.46	79.34 $\pm$ 3.86	78.36 $\pm$ 3.12	78.28 $\pm$ 4.35	78.33 $\pm$ 4.83
Training Time (sec.)	4.67	7.13	3.28	357.42	11.33	85.92	260.95	195.20	536.63	388.12	449.25
Test Time (sec.)	0.07	47.74	6.56	883.79	111.53	6.68	164.50	140.91	167.92	125.55	110.42

that the proposed dual graph learning, including DGCN and DCGN-DC, plays an important role in capturing relationships in and among samples for HSI classification by increasing between-class distances and minimizing intraclass distances.

3) *Overall Results of DGCN-D*: The results of the proposed DGCN-D are reported in the last second column in Tables III–VI. It can be seen that DGCN-D achieves higher accuracy on the four data sets compared with the proposed

DGCN. More specifically, DGCN-D increases accuracies by 0.49%, 0.33%, and 0.31% in terms of OA, AA, and  $K$  on the Salinas data set. Besides, DGCN-D outperforms DGCN by 2.08% in terms of OA, 4.08% in terms of AA, and 3.91% in terms of  $K$  on the Indian Pines data set; by 1.61% in terms of OA, 2.75% in terms of AA, and 1.51% in terms of  $K$  on the Pavia data set; and by 2.12% in terms of OA, 2.36% in terms of AA and 2.23% in terms of  $K$  on the Houston

TABLE VI

CLASSIFICATION RESULTS (VALUES  $\pm$  STANDARD DEVIATION) ON THE 2018 HOUSTON DATA SET USING FIVE TRAINING SAMPLES PER EACH CLASS

Method	EMP-SVM	CNN	EMP-CNN	SCNN	HySN	GCN	DGCN	DGCN-D	DGCN-C	DGCN-M	DGCN-DC
OA(%)	57.80 $\pm$ 1.30	47.68 $\pm$ 2.30	60.17 $\pm$ 2.55	52.67 $\pm$ 1.73	56.08 $\pm$ 2.85	54.55 $\pm$ 0.47	60.80 $\pm$ 3.14	62.92 $\pm$ 2.64	61.58 $\pm$ 1.44	62.24 $\pm$ 1.85	<b>63.52<math>\pm</math>1.15</b>
AA(%)	61.84 $\pm$ 0.92	49.76 $\pm$ 2.30	63.26 $\pm$ 2.42	54.67 $\pm$ 1.64	61.08 $\pm$ 2.57	59.64 $\pm$ 0.42	60.69 $\pm$ 3.31	63.05 $\pm$ 2.58	61.62 $\pm$ 1.32	62.30 $\pm$ 1.78	<b>63.60<math>\pm</math>1.29</b>
$K \times 100$	55.37 $\pm$ 1.39	44.68 $\pm$ 2.44	57.81 $\pm$ 2.70	49.96 $\pm$ 1.82	53.52 $\pm$ 2.98	51.88 $\pm$ 0.04	58.65 $\pm$ 3.32	60.88 $\pm$ 2.78	59.46 $\pm$ 1.52	60.16 $\pm$ 1.95	<b>61.51<math>\pm</math>1.22</b>
Healthy grass	<b>87.30<math>\pm</math>7.59</b>	76.94 $\pm$ 15.07	81.75 $\pm$ 10.26	77.17 $\pm$ 4.59	62.26 $\pm$ 17.40	69.52 $\pm$ 6.17	61.30 $\pm$ 4.15	61.59 $\pm$ 1.99	60.27 $\pm$ 2.06	60.80 $\pm$ 2.53	61.65 $\pm$ 0.35
Stressed grass	68.91 $\pm$ 17.55	50.01 $\pm$ 16.24	57.81 $\pm$ 16.17	38.92 $\pm$ 3.62	49.16 $\pm$ 11.96	<b>77.29<math>\pm</math>13.86</b>	61.53 $\pm$ 3.21	63.84 $\pm$ 2.43	63.20 $\pm$ 0.91	61.30 $\pm$ 1.10	64.97 $\pm$ 0.76
Synthetic grass	84.43 $\pm$ 17.03	43.93 $\pm$ 10.77	89.12 $\pm$ 20.30	65.60 $\pm$ 2.19	90.77 $\pm$ 16.94	<b>100.00<math>\pm</math>0.00</b>	63.71 $\pm$ 3.70	<b>61.97<math>\pm</math>2.61</b>	59.69 $\pm$ 1.72	60.82 $\pm$ 3.41	61.53 $\pm$ 1.41
Evergreen Trees	83.19 $\pm$ 7.70	73.48 $\pm$ 11.85	84.48 $\pm$ 4.80	70.07 $\pm$ 5.86	73.90 $\pm$ 24.16	<b>91.63<math>\pm</math>1.06</b>	61.38 $\pm$ 2.89	63.60 $\pm$ 2.65	62.20 $\pm$ 0.59	63.10 $\pm$ 1.75	63.74 $\pm$ 0.76
Deciduous Trees	64.87 $\pm$ 8.02	42.89 $\pm$ 13.01	56.84 $\pm$ 11.70	64.91 $\pm$ 5.73	46.99 $\pm$ 10.47	<b>65.31<math>\pm</math>6.35</b>	61.38 $\pm$ 3.00	63.89 $\pm$ 3.34	62.50 $\pm$ 3.45	64.60 $\pm$ 5.84	63.29 $\pm$ 1.29
Soil	71.03 $\pm$ 15.15	55.96 $\pm$ 10.49	80.29 $\pm$ 16.99	86.13 $\pm$ 1.72	<b>93.94<math>\pm</math>3.92</b>	87.72 $\pm$ 0.28	62.48 $\pm$ 2.41	61.71 $\pm$ 2.84	59.82 $\pm$ 1.41	61.30 $\pm$ 1.39	62.47 $\pm$ 4.15
Water	86.54 $\pm$ 21.28	58.46 $\pm$ 17.79	69.46 $\pm$ 17.79	61.38 $\pm$ 9.01	<b>99.07<math>\pm</math>0.96</b>	98.47 $\pm$ 0.01	59.02 $\pm$ 5.35	66.00 $\pm$ 2.91	66.35 $\pm$ 1.94	63.53 $\pm$ 1.28	65.42 $\pm$ 1.07
Residential	<b>71.13<math>\pm</math>5.81</b>	55.61 $\pm$ 14.76	72.05 $\pm$ 12.39	48.68 $\pm$ 5.82	35.31 $\pm$ 13.44	22.56 $\pm$ 4.96	60.95 $\pm$ 2.14	62.96 $\pm$ 3.81	62.00 $\pm$ 1.65	60.50 $\pm$ 1.77	63.14 $\pm$ 1.18
Commercial	20.91 $\pm$ 9.94	20.96 $\pm$ 8.07	23.54 $\pm$ 6.05	22.44 $\pm$ 6.26	29.03 $\pm$ 10.42	33.08 $\pm$ 0.99	62.75 $\pm$ 4.34	62.01 $\pm$ 3.56	60.48 $\pm$ 2.44	62.90 $\pm$ 2.37	<b>63.09<math>\pm</math>0.26</b>
Road	28.02 $\pm$ 5.56	18.54 $\pm$ 6.55	21.10 $\pm$ 6.65	22.62 $\pm$ 4.42	9.77 $\pm$ 3.20	14.39 $\pm$ 6.00	60.78 $\pm$ 3.83	<b>62.34<math>\pm</math>3.80</b>	60.80 $\pm$ 3.05	62.00 $\pm$ 2.01	61.74 $\pm$ 0.05
Sidewalk	15.01 $\pm$ 7.20	13.46 $\pm$ 3.63	12.37 $\pm$ 3.99	13.26 $\pm$ 2.74	12.96 $\pm$ 11.97	9.32 $\pm$ 0.42	60.53 $\pm$ 3.64	64.03 $\pm$ 3.08	63.15 $\pm$ 1.30	63.90 $\pm$ 3.20	<b>64.59<math>\pm</math>0.12</b>
Crosswalk	38.49 $\pm$ 10.83	25.42 $\pm$ 6.40	29.07 $\pm$ 6.99	29.77 $\pm$ 8.01	27.12 $\pm$ 9.80	19.99 $\pm$ 1.50	60.13 $\pm$ 3.00	61.53 $\pm$ 2.59	59.32 $\pm$ 1.22	58.40 $\pm$ 2.21	<b>62.15<math>\pm</math>2.19</b>
Major Thoroughfares	24.00 $\pm$ 5.75	21.59 $\pm$ 6.81	36.50 $\pm$ 8.80	26.07 $\pm$ 1.93	24.40 $\pm$ 6.19	11.73 $\pm$ 4.22	60.60 $\pm$ 5.23	60.44 $\pm$ 2.44	60.00 $\pm$ 1.46	60.30 $\pm$ 0.66	<b>63.65<math>\pm</math>2.19</b>
Highway	59.60 $\pm$ 15.50	64.47 $\pm$ 13.14	80.54 $\pm$ 8.30	79.73 $\pm$ 6.76	<b>82.51<math>\pm</math>12.01</b>	61.45 $\pm$ 9.08	59.60 $\pm$ 4.00	62.27 $\pm$ 3.70	60.30 $\pm$ 2.76	64.10 $\pm$ 2.62	64.22 $\pm$ 3.09
Railway	82.34 $\pm$ 12.04	80.15 $\pm$ 10.46	<b>87.68<math>\pm</math>10.50</b>	80.20 $\pm$ 7.27	83.28 $\pm$ 16.95	83.71 $\pm$ 4.54	60.30 $\pm$ 4.57	63.99 $\pm$ 2.19	62.00 $\pm$ 1.19	60.00 $\pm$ 1.02	65.00 $\pm$ 1.98
Paved Parking Lot	66.53 $\pm$ 8.16	42.70 $\pm$ 17.43	<b>68.21<math>\pm</math>15.27</b>	66.69 $\pm$ 8.97	66.40 $\pm$ 19.65	46.92 $\pm$ 10.32	58.90 $\pm$ 2.83	63.41 $\pm$ 3.77	61.75 $\pm$ 3.52	62.70 $\pm$ 2.07	65.19 $\pm$ 2.85
Gravel Parking Lot	97.92 $\pm$ 3.33	90.97 $\pm$ 14.93	99.11 $\pm$ 2.36	87.64 $\pm$ 7.34	98.91 $\pm$ 2.89	<b>100.00<math>\pm</math>0.00</b>	58.56 $\pm$ 5.46	63.57 $\pm$ 3.79	59.23 $\pm$ 1.60	63.09 $\pm$ 2.51	64.66 $\pm$ 7.28
Cars	58.46 $\pm$ 6.77	50.53 $\pm$ 13.50	<b>78.26<math>\pm</math>2.22</b>	28.13 $\pm$ 7.06	65.26 $\pm$ 16.43	57.04 $\pm$ 1.32	58.60 $\pm$ 3.01	63.97 $\pm$ 3.92	62.78 $\pm$ 2.61	62.60 $\pm$ 2.55	63.09 $\pm$ 1.15
Trains	56.19 $\pm$ 4.98	52.01 $\pm$ 17.04	66.06 $\pm$ 12.82	76.06 $\pm$ 5.48	<b>78.89<math>\pm</math>11.01</b>	68.77 $\pm$ 9.85	60.38 $\pm$ 3.02	63.99 $\pm$ 2.22	63.98 $\pm$ 0.13	64.50 $\pm$ 1.47	64.49 $\pm$ 1.01
Seats	71.89 $\pm$ 10.37	57.09 $\pm$ 12.84	71.07 $\pm$ 21.20	47.95 $\pm$ 8.18	<b>91.65<math>\pm</math>3.98</b>	73.98 $\pm$ 2.31	60.88 $\pm$ 2.15	63.87 $\pm$ 3.95	62.35 $\pm$ 2.41	65.60 $\pm$ 1.32	64.05 $\pm$ 2.33
Training Time (sec.)	19.36	12.08	6.28	649.60	28.87	279.31	1060.29	1220.52	2113.68	1272.77	2033.39
Test Time (sec.)	0.17	25.76	6.63	2138.19	43.70	82.81	135.31	140.11	130.87	140.33	142.84

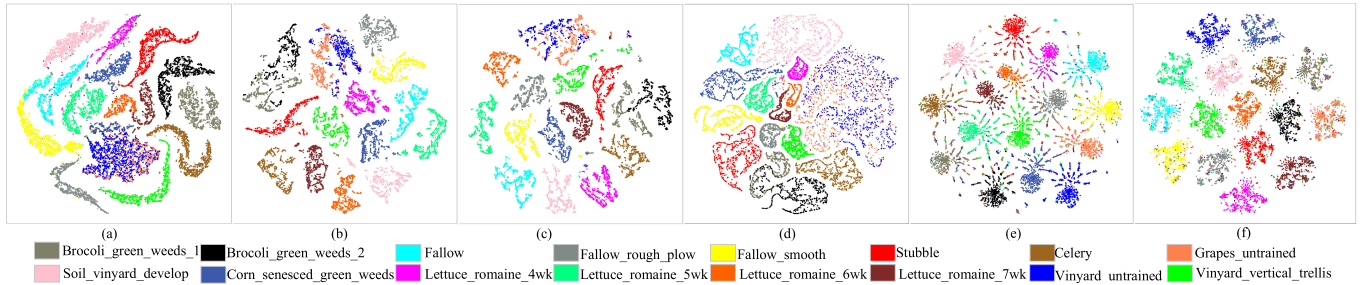


Fig. 10. T-SNE visualization on the Salinas data set.

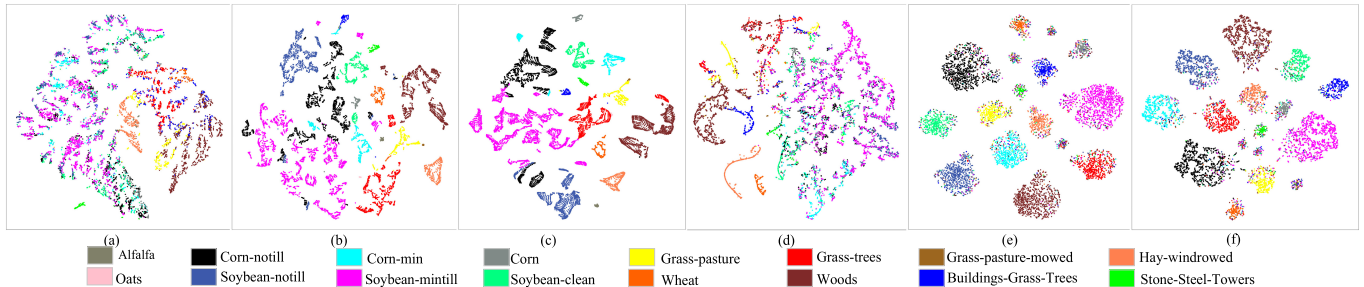


Fig. 11. T-SNE visualization on the Indian Pines data set.

data set. From the above results, DGCN-D further delivers promising enhancement on different data sets. It demonstrates that DGCN-D is an effective method to prevent the overfitting problem in graph learning for HSI classification.

4) *Overall Results of DGCN-DC*: The results of the integration of DGCN-C and DGCN-D are presented for comparison on the four data sets, which are shown in Tables III–VI. Compared to other methods (i.e., EMP-SVM, CNN, DGCN,

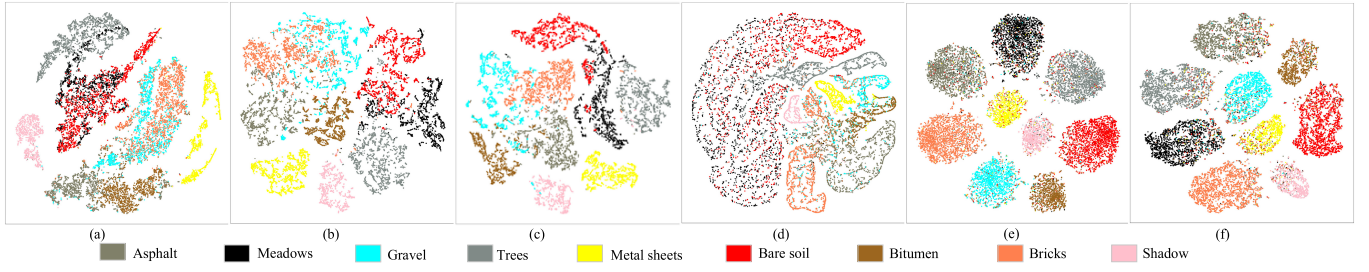


Fig. 12. T-SNE visualization on the Pavia data set.

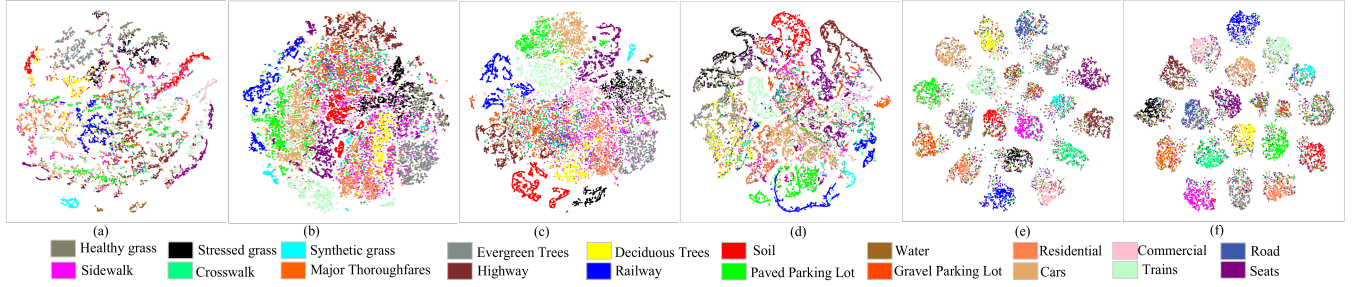


Fig. 13. T-SNE visualization on the Houston data set.

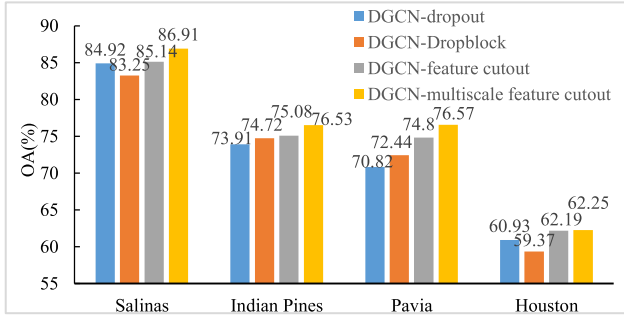


Fig. 14. Classification results of DGCN with dropout, dropblock, feature cutout, or multiscale feature cutout.

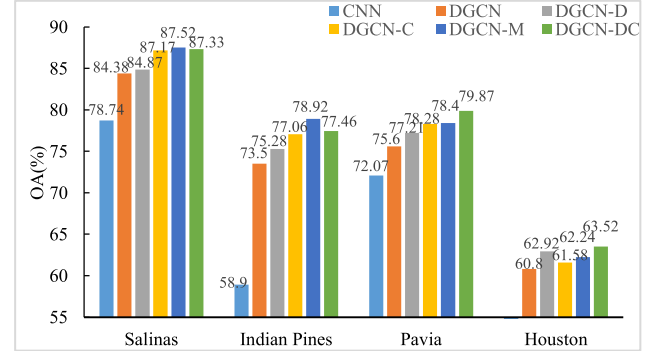


Fig. 15. Effects of each part of different methods (i.e., CNN, DGCN, DGCN-D, DGCN-C, DGCN-M, and DGCN-DC) on improving classification accuracy.

DGCN-D, and DGCN-C), it can be seen that DGCN-DC, which combines DGCN-D with DGCN-C, achieves the highest classification accuracy on the four data sets. Especially, for the Pavia data set, DGCN-DC increases the OA compared with the DGCN, DGCN-D, and DGCN-C by 4.27%, 2.66%, and 1.59%, respectively. The results demonstrate that DGCN-DC, which integrates two different strategies for HSI classification improvement, is also an effective method to prevent the overfitting problem in graph learning for HSI classification.

5) *Classification Results of DGCN-M*: We compare DGCN, DGCN with feature cutout, and DGCN with multiscale feature cutout to validate the effectiveness of the proposed technique for HSI classification. The test accuracies are shown in Tables III–VI. From the results, it can be easily observed that DGCN-C always has a superior performance compared to DGCN. For example, the OA of DGCN-C reaches 87.17%, 77.06%, 78.28%, and 61.58% with only five training samples per each class on the four data sets. Especially, for the Pavia data set, DGCN-C has an improvement of nearly 2% compared to DGCN. Furthermore, the

OAs of DGCN-M are higher than DGCN-C on the four data sets.

6) *Comparing DGCN With Dropout, Feature Cutout, or DropBlock*: In order to evaluate the effectiveness of the proposed feature cutout, dropout and DropBlock are utilized for comparison. Here, we randomly selected five samples per each class, the feature cutout is added after the first convolutional layer, and dropout is added after the first FC layer. Besides, to verify the effectiveness of the proposed DGCN with DropBlock and feature cutout, the settings of both methods are the same for a fair comparison. The obtained results by the three methods are shown in Fig. 14. It can be observed that DGCN with feature cutout exhibits better OA than DGCN with dropout on all four data sets. Especially, for the more complex Pavia scene, the classification accuracy of DGCN with feature cutout is higher than DGCN with dropout by 3.98%, which illustrates that the proposed feature cutout is more effective than dropout in alleviating the overfitting problem. In addition, compared to DGCN with DropBlock,



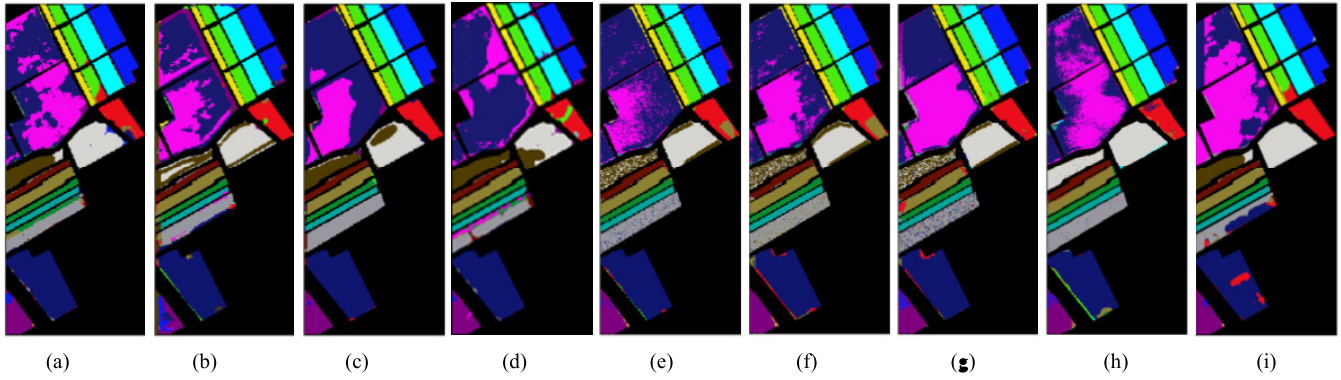


Fig. 16. Salinas: classification maps of (a) EMP-SVM, (b) CNN, (c) EMP-CNN, (d) SCNN, (e) DGCN, (f) DGCN-D, (g) DGCN-C, (h) DGCN-M, and (i) DGCN-DC.

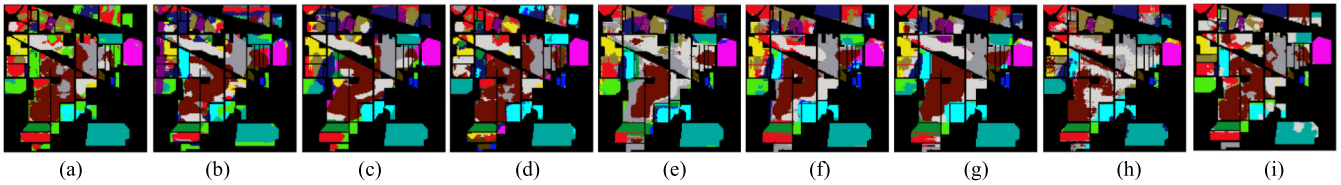


Fig. 17. Indian Pines: classification maps of (a) EMP-SVM, (b) CNN, (c) EMP-CNN, (d) SCNN, (e) DGCN, (f) DGCN-D, (g) DGCN-C, (h) DGCN-M, and (i) DGCN-DC.

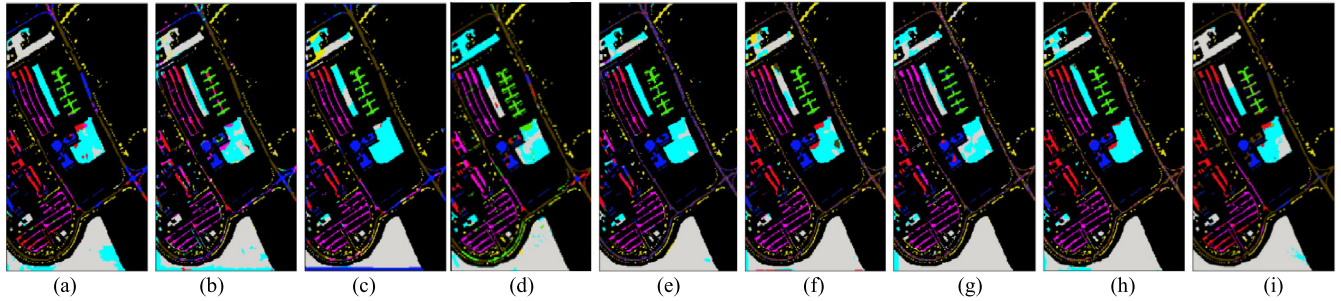


Fig. 18. Pavia: classification maps of (a) EMP-SVM, (b) CNN, (c) EMP-CNN, (d) SCNN, (e) DGCN, (f) DGCN-D, (g) DGCN-C, (h) DGCN-M, and (i) DGCN-DC.

DGCN with feature cutout achieves the highest classification accuracy on the four data sets and obtains higher robustness on the obtained results, which achieves 1.89%, 0.36%, 2.36%, and 2.82% on the Salinas, Indian Pines, Pavia, and Houston data sets, respectively. The classification results fully prove the feature cutout is a simple and powerful regularization method by dropping out the continuous sections of feature maps, and feature cutout improves the performance of DGCN by influencing the subsequent feature extraction steps.

7) *Analysis of the Effect of Each Part of the Proposed Methods on Improving Classification Accuracy:* The proposed methods include DGCN, DGCN-D, DGCN-C, DGCN-M, and DGCN-DC. Each method consists of several parts. The effect of each part of the aforementioned methods in terms of classification accuracy is analyzed, which is shown in Fig. 15. Especially, DGCN contains three main parts (i.e., data pre-processing, CNN feature extraction, and dual graph learning). Since the output features of CNN are used to initialize the nodes in the graph, the nodes in DGCN cannot be constructed

without CNN. Thus, in this method, we only analyze the effect of the single CNN part in the DGCN. A comparison of the single CNN part in the DGCN and DGCN is shown in Fig. 15. As we can observe, compared to the single CNN part, the proposed DGCN improves the classification results of the four data sets by 5.64%, 14.60%, 3.53%, and 9.35% in terms of OA. Besides, DGCN-D consists of two main parts (i.e., DGCN and drop edge), where the architecture of DGCN is the same as above. Compared to DGCN, DGCN-D delivers promising improvement for different data sets. In addition, DGCN-C includes two main parts, including DGCN and feature cutout. Similarly, there are two main parts in DGCN-M, including DGCN and multiscale feature cutout. In the experiments, the architectures of DGCN in the DGCN-C and DGCN-M are the same. The results illustrate the effectiveness of the two proposed techniques for HSI classification. For DGCN-DC, there are three main parts in the DGCN-DC, which includes the DGCN, drop edge, and feature cutout. To analyze the effect of these three parts, the results of DGCN, DGCN-D,



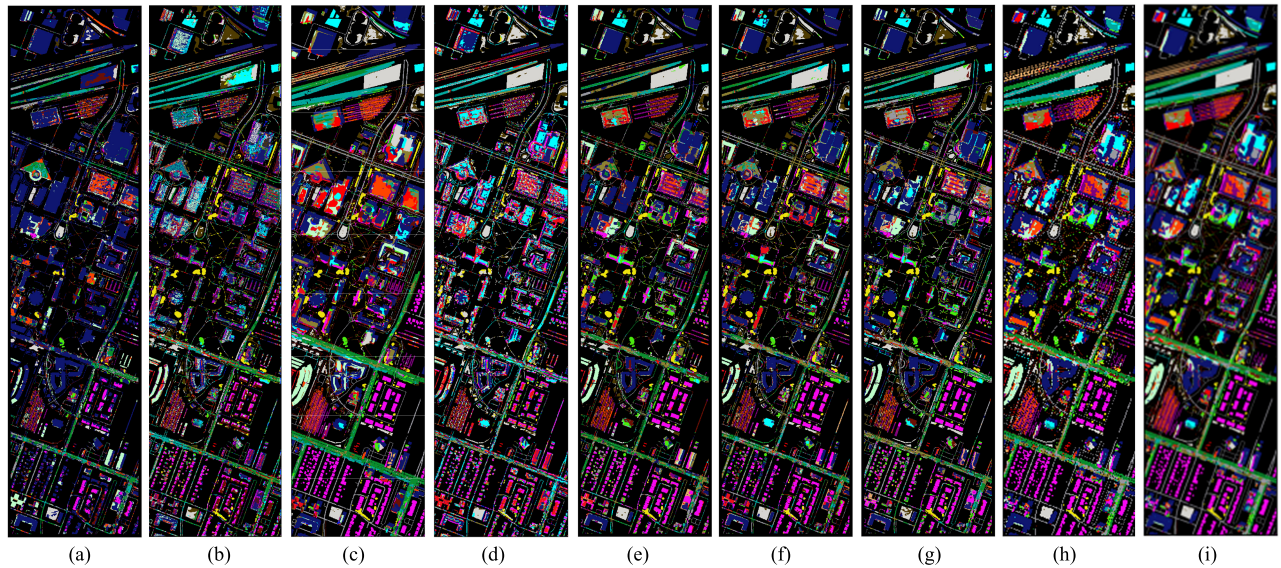


Fig. 19. Houston: classification maps of (a) EMP-SVM, (b) CNN, (c) EMP-CNN, (d) SCNN, (e) DGCN, (f) DGCN-D, (g) DGCN-C, (h) DGCN-M, and (i) DGCN-DC.

DGCN-C, and DGCN-DC are also shown in Fig. 15. The results demonstrate that DGCN-DC is also an effective method to prevent the overfitting problem in graph learning for HSI classification.

#### E. Discussion of Running Time

The training and test time of different methods are summarized in the last row of Tables III–VI when only five training samples per each class are utilized. All experiments are performed on a computer with an Intel Core i5-8500 processor with 3 GHz, 12 GB of DDR4 RAM, and an NVIDIA GeForce GTX TITAN X graphical processing unit (GPU).

As can be observed from Tables III–VI, the training and test time of EMP-SVM is short, and EMP-SVM achieves competitive results. For the competitive deep-learning-based methods (i.e., CNN and EMP-CNN), the processing time of these methods is shorter than DGCN because fewer parameters are needed to train. However, CNN shows poor performance with limited training samples. In addition, SCNN takes the longest time since it needs thousands of iterations to train the model. Besides, DGCN is time-consuming on all data sets compared to other methods except SCNN. Because there are two parts in the proposed DGCN: CNN and dual graph network. CNN is used to extract the features from the individual sample, and the dual graph network is used to extract the features among training samples. The proposed DGCN has more parameters than CNN, but it achieves better results than CNN. In addition, DGCN-D is able to facilitate the training of DGCN, which also reaches higher accuracy. Specifically, compared to DGCN, the training time of DGCN-D decreases by 48.03, 66.89, and 65.75 s on the Salinas, Indian Pines, and Pavia data sets, respectively. Besides, while DGCN-M has a higher processing time than DGCN, it has a good performance in terms of accuracies, which proves that DGCN-M is an effective regularization method to classify HSI.

#### F. Classification Map

Figs. 16–19 visualize the classification maps of different methods, which includes EMP-SVM, CNN, EMP-CNN, SCNN, DGCN, DGCN-D, DGCN-C, DGCN-M, and DGCN-DC. As shown in Figs. 16–19, it can be observed that EMP-SVM has more correctly classified pixels than those deep-learning-based methods. In addition, classification maps of CNN and SCNN produce more errors for the four data sets with the small training sample size. For the Salinas data set, many pixels are misclassified on the boundary between different classes [see Fig. 16(b) and (d)]. For the visual result of EMP-CNN [see Fig. 16(c)], there are more correctly classified pixels compared to other comparative deep-learning approaches, while the proposed methods (i.e., DGCN, DGCN-D, DGCN-C, DGCN-M, and DGCN-DC) preserve edge information well compared to other approaches, especially for the class of the grapes untrained with mulberry color. Besides, taking the Pavia data set as an example (see Fig. 18), for the proposed DGCN-DC, many pixels on the bottom are correctly classified; however, the competitive methods provide wrong classification results. The Indian Pines and Houston data sets follow similar results as the Pavia data set. All the visual results illustrate that the proposed methods based on graph learning for HSI classification have better performance than other competitive approaches.

## IV. CONCLUSION

In this study, GCN was explored for HSI classification with limited training samples. Several GCN-based methods, including DGCN, DGCN-D, DGCN-C, DGCN-DC, and DGCN-M, were proposed.

For DGCN, we used two GCNs to fully extract features in and among HSI training samples, which led to better classification performance compared with traditional CNN-based methods.

In order to mitigate the overfitting problem caused by limited training samples, DGCN-D, DGCN-C, DGCN-DC, and DGCN-M used drop edge, feature cutout, drop edge and feature cutout, and multiscale feature cutout to enhance the test accuracy of HSI, respectively.

The superiority of the proposed methods has been illustrated on four hyperspectral data sets by achieving better results in terms of classification accuracies. The proposed DGCN demonstrated that the proper usage of extracting features among training samples has great potential for accurate classification of HSI.

## REFERENCES

- [1] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, no. 1, pp. 110–122, Sep. 2009.
- [2] C.-I. Chang, *Hyperspectral Data Exploitation: Theory and Applications*. Hoboken, NJ, USA: Wiley, 2007.
- [3] P. Ghamisi *et al.*, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 37–78, Dec. 2017.
- [4] G. Camps-Valls, D. Tuia, L. Bruzzone, and J. A. Benediktsson, "Advances in hyperspectral image classification: Earth monitoring with statistical learning methods," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 45–54, Jan. 2014.
- [5] P. Ghamisi *et al.*, "New frontiers in spectral-spatial hyperspectral image classification: The latest advances based on mathematical morphology, Markov random fields, segmentation, sparse representation, and deep learning," *IEEE Geosci. Remote Sens. Mag.*, vol. 6, no. 3, pp. 10–43, Sep. 2018.
- [6] P. Soille, "Morphological image analysis: Principles and applications," *Sensor Rev.*, vol. 20, no. 3, pp. 800–801, Mar. 1999.
- [7] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3804–3814, Nov. 2008.
- [8] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, "Classification of hyperspectral data from urban areas based on extended morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 480–491, Mar. 2005.
- [9] L. Fang, N. He, S. Li, P. Ghamisi, and J. A. Benediktsson, "Extinction profiles fusion for hyperspectral images classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 3, pp. 1803–1815, Mar. 2018.
- [10] W. Kim and M. M. Crawford, "Adaptive classification for hyperspectral image data using manifold regularization kernel machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 11, pp. 4110–4121, Nov. 2010.
- [11] J. Li, P. R. Marpu, A. Plaza, J. M. Bioucas-Dias, and J. A. Benediktsson, "Generalized composite kernel framework for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 9, pp. 4816–4829, Sep. 2013.
- [12] Q. Wang, X. He, and X. Li, "Locality and structure regularized low rank representation for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 911–923, Feb. 2019.
- [13] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proc. 27th Int. Conf. Mach. Learn.*, Jun. 2010, pp. 663–670.
- [14] X. Sun, Q. Qu, N. M. Nasrabadi, and T. D. Tran, "Structured priors for sparse-representation-based hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 7, pp. 1235–1239, Jul. 2014.
- [15] L. He, J. Li, C. Liu, and S. Li, "Recent advances on spectral-spatial hyperspectral image classification: An overview and new guidelines," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 3, pp. 1579–1597, Mar. 2018.
- [16] S. Jia, L. Shen, and Q. Li, "Gabor feature-based collaborative representation for hyperspectral imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 2, pp. 1118–1129, Feb. 2015.
- [17] B. Hou, T. Huang, and L. Jiao, "Spectral-spatial classification of hyperspectral data using 3-D morphological profile," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2364–2368, Dec. 2015.
- [18] S. Jia, J. Hu, J. Zhu, X. Jia, and Q. Li, "Three-dimensional local binary patterns for hyperspectral imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 4, pp. 2399–2413, Apr. 2017.
- [19] J. Zhu, J. Hu, S. Jia, X. Jia, and Q. Li, "Multiple 3-D feature fusion framework for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 1873–1886, Apr. 2018.
- [20] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Deep learning for hyperspectral image classification: An overview," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 6690–6709, Sep. 2019.
- [21] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Deep learning classifiers for hyperspectral imaging: A review," *ISPRS J. Photogramm. Remote Sens.*, vol. 158, pp. 279–317, Dec. 2019.
- [22] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.
- [23] W. Zhao and S. Du, "Spectral-spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4544–4554, Aug. 2016.
- [24] L. He *et al.*, "Feature extraction with multiscale covariance maps for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 202–216, Feb. 2019.
- [25] E. Aptoula, M. C. Ozdemir, and B. Yanikoglu, "Deep learning with attribute profiles for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1970–1974, Dec. 2016.
- [26] L. He, C. Liu, J. Li, Y. Li, S. Li, and Z. Yu, "Hyperspectral image spectral-spatial-range Gabor filtering," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 7, pp. 4818–4836, Jul. 2020.
- [27] 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.
- [28] H. Zhang, Y. Li, Y. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery using a dual-channel convolutional neural network," *Remote Sens. Lett.*, vol. 8, no. 5, pp. 438–447, May 2017.
- [29] Y. Xu, L. Zhang, B. Du, and F. Zhang, "Spectral-spatial unified networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 10, pp. 5809–5893, Oct. 2018.
- [30] Y. Li, H. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery with 3D convolutional neural network," *Remote Sens.*, vol. 9, no. 1, pp. 67–80, 2017.
- [31] M. He, B. Li, and H. Chen, "Multi-scale 3D deep convolutional neural network for hyperspectral image classification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3904–3908.
- [32] Z. Zhong, J. Li, Z. Luo, and M. Chapman, "Spectral-spatial residual network for hyperspectral image classification: A 3D deep learning framework," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 847–858, Feb. 2018.
- [33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
- [34] J. Song, L. Xiao, M. Molaei, and Z. Lian, "Multi-layer boosting sparse convolutional model for generalized nuclear segmentation from histopathology images," *Knowl.-Based Syst.*, vol. 176, pp. 40–53, Jul. 2019.
- [35] W. Wang *et al.*, "Alternately updated spectral-spatial convolution network for the classification of hyperspectral images," *Remote Sens.*, vol. 11, no. 5, pp. 1794–1805, Jul. 2019.
- [36] 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.
- [37] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. J. Plaza, and F. Pla, "Deep pyramidal residual networks for spectral-spatial hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 740–754, Feb. 2019.
- [38] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
- [39] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Deep & dense convolutional neural network for hyperspectral image classification," *Remote Sens.*, vol. 10, no. 9, pp. 1454–1474, Sep. 2018.
- [40] R. Hang, Q. Liu, D. Hong, and P. Ghamisi, "Cascaded recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5384–5394, Aug. 2019.
- [41] K. Zhu, Y. Chen, P. Ghamisi, X. Jia, and J. A. Benediktsson, "Deep convolutional capsule network for hyperspectral image spectral and spectral-spatial classification," *Remote Sens.*, vol. 11, no. 3, pp. 223–243, Jan. 2019.



- [42] M. E. Paoletti *et al.*, "Capsule networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2145–2160, Apr. 2019.
- [43] 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.
- [44] A. Sha, B. Wang, X. Wu, and L. Zhang, "Semisupervised classification for hyperspectral images using graph attention networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 1, pp. 157–161, Jan. 2021.
- [45] S. Wan, C. Gong, P. Zhong, S. Pan, G. Li, and J. Yang, "Hyperspectral image classification with context-aware dynamic graph convolutional network," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 1, pp. 597–612, May 2020.
- [46] 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.
- [47] 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.
- [48] 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.*, early access, Aug. 18, 2020, doi: 10.1109/TGRS.2020.3015157.
- [49] B. Liu, X. Yu, A. Yu, P. Zhang, G. Wan, and R. Wang, "Deep few-shot learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2290–2304, Apr. 2019.
- [50] S. Jia, Z. Zhu, L. Shen, and Q. Li, "A two-stage feature selection framework for hyperspectral image classification using few labeled samples," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 4, pp. 1023–1035, Apr. 2014.
- [51] Q. Sami ul Haq, L. Tao, F. Sun, and S. Yang, "A fast and robust sparse approach for hyperspectral data classification using a few labeled samples," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 6, pp. 2287–2302, Jun. 2012.
- [52] R. Huang *et al.*, "DropEdge: Towards deep graph convolutional networks on node classification," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Mar. 2020, pp. 10–28.
- [53] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
- [54] Y. Li and A. Gupta, "Beyond grids: Learning graph representations for visual recognition," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 9225–9235.
- [55] Y. S. Aurelio, G. M. de Almeida, C. L. de Castro, and A. P. Braga, "Learning from imbalanced data sets with weighted cross-entropy function," *Neural Process. Lett.*, vol. 50, no. 2, pp. 1937–1949, Oct. 2019.
- [56] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Nov. 2017, pp. 710–723.
- [57] G. Ghiasi, T. Y. Lin, and Q. V. Le, "DropBlock: A regularization method for convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Cambridge, MA, USA, 2018, pp. 10727–10737.
- [58] Y. Chen, L. Zhu, P. Ghamisi, X. Jia, G. Li, and L. Tang, "Hyperspectral images classification with Gabor filtering and convolutional neural network," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 12, pp. 2355–2359, Dec. 2017.
- [59] 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 Geosci. Remote Sens. Lett.*, vol. 17, no. 2, pp. 277–281, Feb. 2020.
- [60] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, vol. 2, Jul. 2015, pp. 1–8.
- [61] L. Huang and Y. Chen, "Dual-path siamese CNN for hyperspectral image classification with limited training samples," *IEEE Geosci. Remote Sens. Lett.*, early access, Mar. 19, 2020, doi: 10.1109/LGRS.2020.2979604.
- [62] L. van der Maaten, "Accelerating t-SNE using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3221–3245, Oct. 2014.



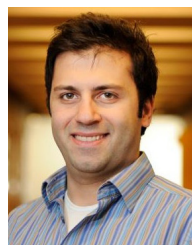
**Xin He** received the M.S. degree from the Harbin University of Science and Technology, Harbin, China, in 2019. She is pursuing the Ph.D. degree with the School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin.

Her research interests include remote sensing image processing based on deep learning methods.



**Yushi Chen** (Member, IEEE) received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2008.

He is an Associate Professor with the School of Electronics and Information Engineering, Harbin Institute of Technology. His research interests include remote sensing data processing and machine learning.



**Pedram Ghamisi** (Senior Member, IEEE) is the Head of the Machine Learning Group, Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany. His research interests include interdisciplinary research on remote sensing and machine (deep) learning, image, and signal processing, and multi-sensory data fusion.