Weak–Strong Graph Contrastive Learning Neural Network for Hyperspectral Image Classification

Sirui Wang, Nassim Ait Ali Braham, and Xiao Xiang Zhu, *Fellow, IEEE*

***Abstract*— Deep learning methods have shown promising results in various hyperspectral image (HSI) analysis tasks. Despite these advancements, existing models still struggle to accurately identify fine-classified land cover types on noisy HSIs. Traditional methods have limited performance when extracting features from noisy hyperspectral data. Graph neural networks (GNNs) offer an adaptable and robust structure by effectively extracting both spectral and spatial features. However, supervised models still require large quantities of labeled data for effective training, posing a significant challenge. Contrastive learning (CL), which leverages unlabeled data for pretraining, can mitigate this issue by reducing the dependency on extensive manual annotation. To address the issues, we propose WSGraphCL, a weak–strong graph CL model for HSI classification, and conduct experiments in a few-shot scenario. First, the image is transformed into K-hop subgraphs through a spectral–spatial adjacency matrix construction method. Second, WSGraphCL leverages CL to pretrain a graph-based encoder on the unlabeled HSI. We demonstrate that weak–strong augmentations and false negative pairs filtering stabilize pretraining and get good-quality representations. Finally, we test our model with a lightweight classifier on the features with a handful of labels. Experimental results showcase the superior performance of WSGraphCL compared to several baseline models, thereby emphasizing its efficacy in addressing the identified limitations in HSI classification. The code repository will be published on the GitHub project under the URL: https://github.com/zhu-xlab/WSGraphCL**

***Index Terms*— Contrastive learning (CL), deep learning, graph neural networks (GNNs), hyperspectral image (HSI) classification, self-supervised learning (SSL).**

I. Introduction

HYPERSPECTRAL images (HSIs) exhibit a continuous spectral structure of ground objects [1]. The spectral range can cover visual near-infrared (VNIR) bands and shortwave infrared (SWIR) bands with minimal bandwidth. With these unique characteristics, HSIs can provide valuable information for Earth observation research and have a wide variety of applications in cropland management [2], city planning [3], and minerals exploration [4]. However, extracting knowledge from hyperspectral data poses numerous challenges. Hyperspectral satellite imagery usually has 10 to 30 m spatial resolution, which contains mixed ground objects within each pixel due to their low spatial resolution [5]. This increases the difficulty of classification tasks because of the diverse spectral information inherent in the data. Second, the high dimensionality of HSIs results in increased storage and computational requirements. Training models on such data is more complicated by prolonged processing time than multispectral images. A clear and light network structure is necessary for HSIs. Third, spectral variability and noise in data can influence the scene [6], posing challenges in distinguishing the classes with similar spectral characteristics. For example, "salt-and-pepper" or "belt" noise occurs frequently for hyperspectral SWIR sensors. Managing such variability is essential to enhancing the reliability of classification results. Finally, efficiently capturing complex spatial and spectral relationships is essential to achieving accurate results. Addressing these issues is crucial for effectively unlocking the full potential of hyperspectral imagery in various domains.

In the past few decades, HSI classification has evolved from manual interpretation and band indices exploration. Machine learning (ML) methods started to exhibit their many advantages in the 2000s [7]. However, the methods that work on multispectral sensors are more common due to capacity and computational power limitations. In its early stages, HSI classification predominantly relied on approaches based on dimensionality reduction. For example, certain significant bands or the initial components can be derived from techniques such as principal component analysis (PCA) [8]. People used to apply feature extraction with classical ML algorithms such as support vector machines (SVMs) [9], random forest (RF) [10], classification and regression trees (CART) [11], and minimum distance classifier [12].

In recent years, researchers have successfully adopted deep learning methods for HSI classification. Deep neural networks have powerful representation learning abilities that allow the extraction of nonlinear projections from large datasets. Various network architectures have been designed for HSI classification, such as recurrent neural networks (RNNs) [13], spatial–spectral perception networks (SSP-Nets) [14], convolutional neural networks (CNNs) [15], graph neural

Received 19 May 2024; revised 23 September 2024, 5 December 2024, and 12 March 2025; accepted 10 April 2025. Date of publication 5 May 2025; date of current version 13 May 2025. This work was supported in part by German Federal Ministry of Education and Research (BMBF) in the framework of the International Future AI Laboratory "AI4EO–Artificial Intelligence for Earth Observation: Reasoning, Uncertainties, Ethics and Beyond" under Grant 01DD20001; and in part by Munich Center for Machine Learning. *(Corresponding author: Xiao Xiang Zhu.)*

Sirui Wang is with the Chair of Data Science in Earth Observation (SiPEO), Technical University of Munich (TUM), 80333 Munich, Germany (e-mail: sirui.wang@tum.de).

Nassim Ait Ali Braham is with the Chair of Data Science in Earth Observation (SiPEO), Technical University of Munich (TUM), 80333 Munich, Germany, and also with German Aerospace Center (DLR), Remote Sensing Technology Institute (IMF), 82234 Wessling, Germany (e-mail: Nassim.AitAliBraham@dlr.de).

Xiao Xiang Zhu is with the Chair of Data Science in Earth Observation (SiPEO), Technical University of Munich (TUM), 80333 Munich, Germany, and also with Munich Center for Machine Learning, 80333 Munich, Germany (e-mail: XiaoXiang.Zhu@tum.de).

Digital Object Identifier 10.1109/TGRS.2025.3562261

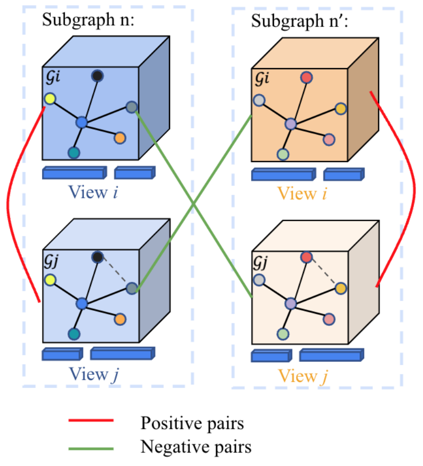


Fig. 1. Illustration of positive and negative pairs.

networks (GNNs) [16], and transformers (SF) [17], [18]. Among them, CNN architectures traditionally employ fixed-size kernels, considering each pixel. This can be challenging for HSIs when the region of interest is large, as it can only take localized spatial information.

In contrast, GNNs are specifically designed to process data structured as graphs rather than patches. It can learn from both local and distant relationships between different pixels in an image. The graph structure can represent the features of nodes and the connectivity between nodes. This mechanism makes GNNs more flexible than CNNs. For example, Qin et al. [19] propose a spectral–spatial GNN (S^2GCN), using spatial distances as the weight of edges and taking spectral features as node attributes.

However, general GNN models are trained in a supervised or semisupervised fashion and require a large number of labeled pixels for training to avoid overfitting. Self-supervised learning (SSL) can mitigate this issue and pretrain an encoder on unlabeled pixels to enable HSI classification with fewer labels [20]. For example, generating pseudo labels can help guide the model more effectively. Ding et al. [21] employ soft labels to supervise the clustering process, then obtain the hidden dimension features. Contrastive learning (CL) is another efficient paradigm among SSL models [22], [23]. It aims to create similar views as the representation of each input by applying different augmentations. An encoder is usually pretrained to maximize the agreement between these views, as shown in Fig. 1. A simple framework for CL, SimCLR [24], is a well-known method that utilizes the information maximizing noise contrastive estimation (InfoNCE) loss. The pretrained stage can obtain discriminative features by training on enormous amounts of unlabeled data. The encoder is frozen and transferred to the downstream task. Only a small number of labels will be used to get the result map with a light and simple classifier.

In this article, we leverage a combination of CL [25] and a weak–strong approach-based GNN (WSGraphCL). The architecture of the weak–strong GraphCL network is shown in Fig. 2. The main contributions of WSGraphCL are as follows.

1) A good graph structure can extract the most essential information from the HSIs. Consequently, a K-hop subgraph data structure is proposed based on superpixels. This structure not only captures localized features within superpixels but also incorporates features and connectivity information from multiscale neighbors. We propose a spectral–spatial graph construction approach. This method employs four distinct edge weighting modes, leveraging spatial locality and spectral similarity within the scene. By incorporating spatial distance, the approach facilitates the extraction of geoinformation from HSI.
2) A graph-based CL framework is proposed for pretraining unlabeled pixels in the image. To ensure a stable pretrained encoder and prevent model collapse, two methods are introduced. First, filtering false negative pairs from all negative pairs in the loss function helps to obtain clean negative pairs. Second, the incorporation of weak–strong augmentation enhances the generation of more effective positive pairs, contributing to the overall efficacy of the pretraining process.
3) The proposed model is evaluated on three popular HSI classification datasets: Indian Pines [26], Pavia University [27], and the MDAS [28] dataset. We investigate the transferability of our model, assessing its adaptability and performance when applied to a distinct dataset.

The structure of the remaining article is organized as follows. Section II reviews relevant prior works. Section III outlines the pipeline of our proposed WSGraphCL method in detail. Section IV delves into the experimental setup, presents results on three datasets, and includes the ablation study for further insights. Section V concludes the article by summarizing key findings and future directions.

II. Related Work

A. GNNs for HSI Classification

A significant body of research has been developed around GNNs for HSI classification. GCNs are convolutional networks based on graph structure data [22]. For example, Ding et al. [29] propose a graph sample and aggregate-attention (GraphSAGE) model that contains only two generations of neighbors. Mou et al. [30] developed a nonlocal graph convolutional network (Nonlocal GCN) that extracts features from the whole image. Wan et al. [31] propose a multiscale dynamic GCN (MDGCN), emerging different scales of subgraphs. Ding et al. [32] propose a diversity-connected GCN (DCGCN) method that enhances HSI classification by refining graph structure through adaptive smoothing and connectivity restrictions, improving interclass separation, and demonstrating low-pass filter properties. Also, Jia et al. [33] propose a graph-in-graph convolutional network (GiGCN) covering information inside and outside superpixels.

Graph attention neural networks (GANs) [34] use spectral, spatial, and temporal attention to give a specific number of graph edges and represent the strength of the connections between nodes. For example, Tang et al. [35] propose a spatial–spectral attention network (3DOC-SSAN) to extract discriminative features and extend their model by a 3-D octave convolution model (3D-OCM) for HSIs classification.

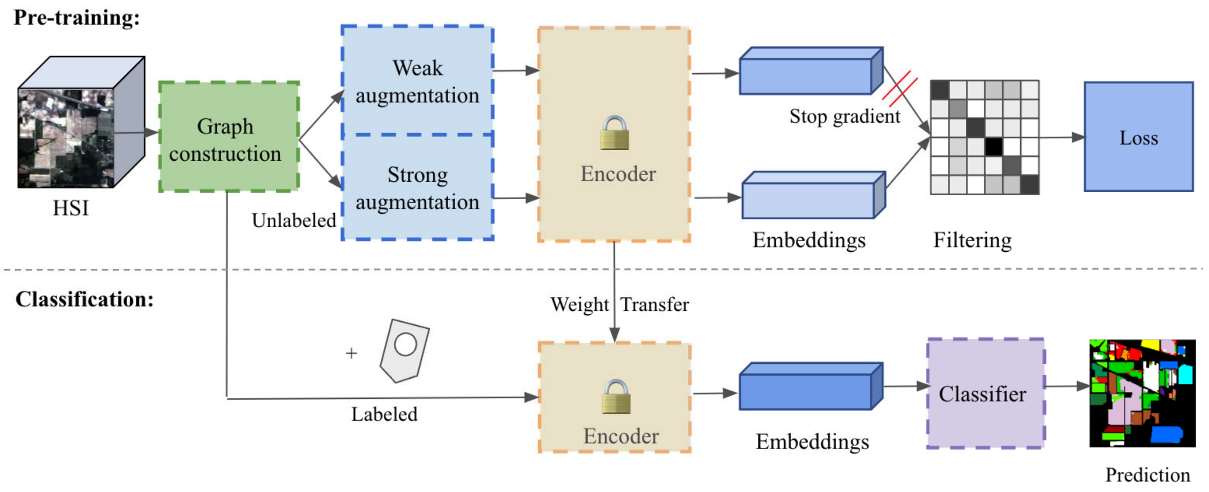


Fig. 2. Architecture of the weak–strong graph CL neural network.

Tong et al. [36] propose an attention-weighted graph convolutional networks (AwGCN) model to find internal relationships in hyperspectral data.

Moreover, some researchers find that graph structure works well when combining other network modules. For example, Hong et al. [37] propose a fusion network FuNet-C, A, M with CNN and consider only mini-batches. Other researchers have found that the performance of graph structure can be improved by creating fusion models with CNN, using such examples as weighted feature fusion of CNN and GAN (WFCG) [38], and CNN enhanced GCN [39]. Chen et al. [40] propose an offset graph U-Net network that takes both spectral–spatial and context-aware information. Zhao et al. [41] propose a fusion model GTN-A by combining GNNs and transformer to get new graph structures. Zhang et al. [42] propose a fusion model TSTnet that mainly combines graph convolutional networks (GCNs) and CNN for cross-scene HSI classification and introduces topological relationship alignment and distribution alignment.

Even though traditional GNNs have various graph structures to extract features, supervised methods are still unable to tackle the challenges within HSI classification tasks effectively, often exhibiting limited performance with few training samples. There is a risk of overfitting during training, where the model becomes too specialized to the small labeled training data and fails to generalize a robust model to unlabeled data. Some researchers, such as Gia-CFSL [43], choose to design a cross-domain feature aggregation structure for few-shot learning. However, CL leverages unlabeled data, making it a powerful approach for addressing the challenges posed by few-sample scenarios.

B. Graph CL

CL involves the generation of two views from the same subgraphs and aims to maximize their agreement [25]. In CL, an encoder is typically pretrained to extract discriminative features by minimizing the associated loss. Positive and negative pairs, derived from augmentations, can be seen as a type of pseudo labels. In this context, positive pairs refer to augmented views originating from the same subgraph, while negative pairs consist of augmented views from different subgraphs, as illustrated in Fig. 1. The primary objective of CL is to maximize the distance between vectors representing negative pairs while minimizing the distance between vectors corresponding to positive pairs. This strategy encourages the encoder to learn representations that can be used effectively for several downstream tasks.

Guan and Lam [44] utilized a CNN CL model for HSI classification. They employ CNN to extract both spectral and spatial signals, which are then optimized into embeddings. Similarly, Yu et al. [45] introduce a contrastive GCN (ConGCN) model, providing initial evidence for the efficacy of contrastive graph networks in HSI classification. Ye et al. [46] propose a cross-domain few-shot learning based on a graph convolution contrast (GCC-FSL) model. GCC-FSL extracts features through a 3D-CNN network and uses a graph contrast loss with a small number of labeled samples. Li et al. [47] propose an end-to-end framework called guided group CL (GGCL) that integrates unsupervised information through a similarity-guided module with limited supervisory signals.

In addition to the previously known benefits of CL, effective improvements in performance rely heavily on the precise definition of positive and negative loss. Generic models may not seamlessly adapt to the unique characteristics of HSIs. Thus, in this article, our goal is to generate a robust model that not only leverages the benefits of CL but also shows the adaptability to both graph structures and hyperspectral data, ensuring performance stability.

III. Weak–Strong Graph CL

As shown in Fig. 2, the proposed WSGraphCL model consists of several components: the graph construction, the GNN architecture, the pretraining algorithm, and the downstream classifier. The details for each component are described in this section.

A. Graph Node Construction

The graph construction involves superpixel map construction and subgraph generation, which we will introduce in detail below.

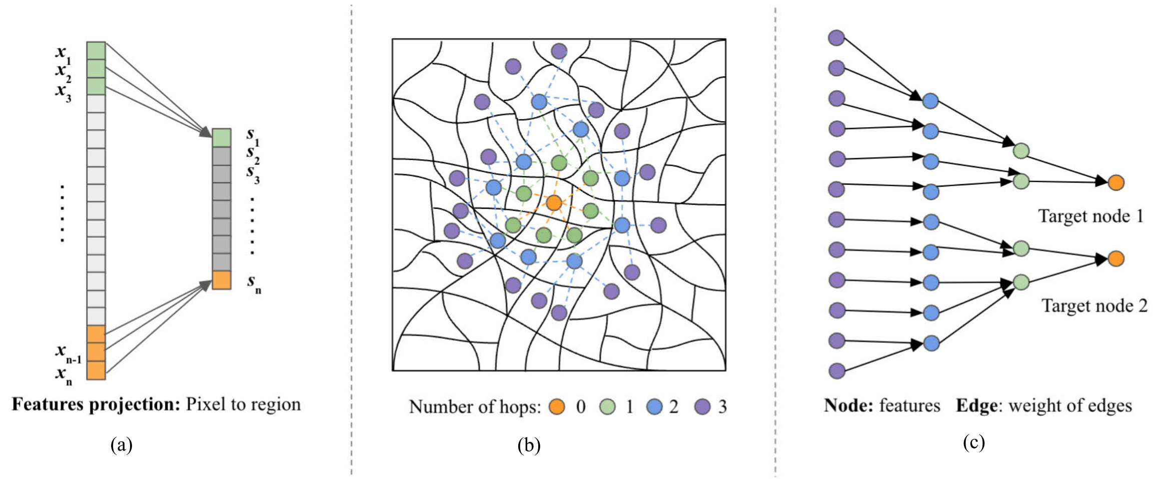

x_1
x_2
x_3
x_{n-1}
x_n
s_1
s_2
s_3
s_n
Features projection: Pixel to region
(a)
Number of hops: ● 0 ● 1 ● 2 ● 3
(b)
Target node 1
Target node 2
Node: features Edge: weight of edges
(c)

Fig. 3. Illustration of superpixel map and graph construction process. (a) Flattened → superpixels. (b) Map of superpixels. (c) Superpixels → nodes.

1) Superpixel Map Generation: Calculating pixel-level distance matrices is memory-intensive for large images. To mitigate this challenge, we use the SLIC [48] algorithm to reduce the dimension to the superpixel level for the graph construction. In this process, the mean value of the pixels within each superpixel is computed, as shown in Fig. 3(a). Each superpixel is treated as an individual node in the constructed graphs, thereby significantly reducing computational complexity. This approach enhances robustness to noise following the projection from pixels to regions. The number of superpixels is a hyperparameter that can be tuned depending on the dataset. Striking a balance is crucial for achieving optimal performance, as too few superpixels may lead to information loss.

2) Subgraph Generation: In this study, we employ the graph classification approach utilizing K-hop subgraphs [49] as input. K-hop subgraphs use the target node as the central point and identify neighbors connected to the target node within k steps. Specifically, the 1-hop subgraph comprises neighbors directly connected to the target node, while the 2-hop subgraph extends to nodes that are two steps away from the central node. This design allows for the consideration of both localized attributes from the target node and attributes from the nodes surrounding its neighbors. The subgraph map, visually represented in Fig. 3(b), utilizes different colors to denote the number of hops from the target node. Ultimately, the signal is transmitted from neighbors to the target node, as depicted in Fig. 3(c). Node attributes are derived from spectral bands in HSIs. This approach offers a comprehensive representation of the spatial and spectral relationships within the hyperspectral data, facilitating effective graph-based classification. We conduct a group of experiments in Section IV with different numbers of hops. In experiments, generating a graph based on the whole image and extracting subgraphs saves training time.

B. Graph Edge Construction

The adjacency matrix serves as an essential component that characterizes edges in graphs, representing both the connectivity and the strength of connectivity between nodes. This matrix plays a crucial role in the convolutional aspect of the graph-based classification. When two nodes are similar, the adjacency matrix assigns a higher edge weight, signifying a stronger connection between these nodes. In the context of HSIs, two superpixels with similar spectral characteristics and close spatial locations can be regarded as connected. The edge attributes are constructed by computing the distance between the connected nodes. Traditional GNN models build both nodes and edges based on spectral information. Our graph structure preserves spatial information by including neighboring pixels. However, it can also capture long-range dependencies and spectral similarity in graph construction. Our adjacency matrix captures the intricacies of spectral–spatial relationships.

1) Spectral–Spatial Adjacency Matrix Approach: In this article, K-nearest neighbor (KNN) [50] is used to define the edge connections and only connect the K most similar nodes. We fix $K = 10$ in all experiments. Let $\mathbf{X} \in \Re^{(H\times W\times C)}$ be the HSI, where H refers to the height, W refers to the width, and C is the number of channels. We consider every superpixel as one node v_i. The adjacency matrix $\mathbf{A}$ represents the graph $G = \{\nu, \varepsilon\}$, where ν is the set of nodes and ε is the set of edges connecting the nodes. Every entry $\mathbf{A}_{i,j}$ of the matrix represents a weighted edge between v_i and v_j, as follows:

$$\mathbf{A}_{i,j} = \begin{cases} f_{\text{Weight Mode}}(d), & \text{if } i \neq j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The edge weight between two nodes is calculated using the function $f_{\text{Weight Mode}}(\cdot)$. Several possible kernels can be used for $f_{\text{Weight Mode}}(\cdot)$. In this article, we consider the heat kernel [51], a Euclidean-like similarity measurement [52], unweighted binary edges, and cosine similarity [53] measurement

$$f_{(\text{Weight Mode=HeatKernel})} = \exp\left(-\frac{d^2_{\text{Spe+Spa}}}{\delta^2}\right) \tag{2}$$

$$f_{(\text{Weight Mode=Euclidean,Cosine})} = 1 - d_{\text{Spe+Spa}} \tag{3}$$

$$f_{(\text{Weight Mode=Binary})} = 1. \tag{4}$$

In the equations above, d represents a spatial–spectral distance between two nodes. We define $\eta \in [0, 1]$ as the

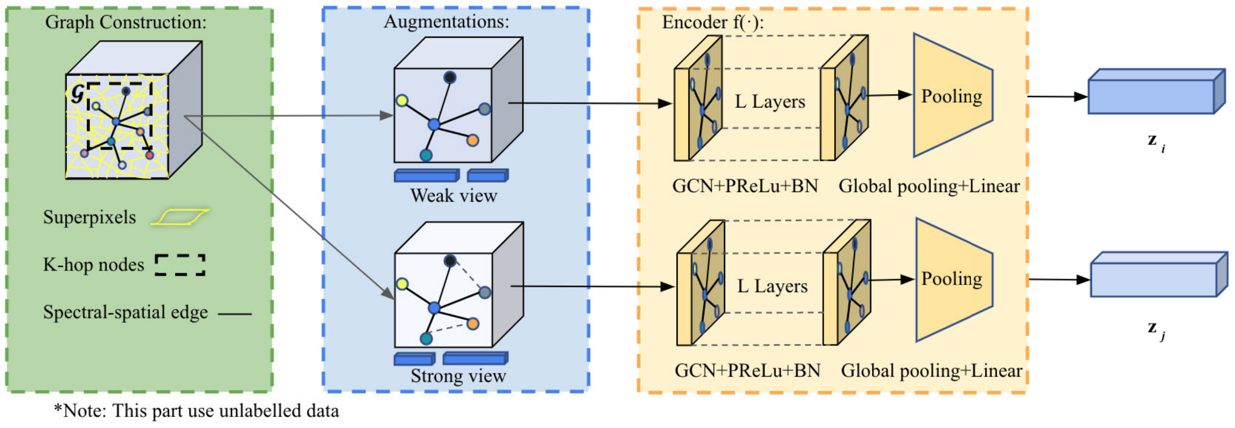


Fig. 4. Illustration of graph construction, augmentations, and graph convolutional encoder in the pretraining stage.

weight scale of spatial distance. The distance d is a convex combination of the spectral distance d_{Spectral} and the spatial distance d_{Spatial}, as detailed in the following equations:

$$d_{\text{Spe+Spa}} = (1-\eta)d_{\text{Spectral}} + \eta d_{\text{Spatial}} \tag{5}$$

$$d_{\text{Spectral}} = \begin{cases} 1-\cos\left(\rho_i', \rho_j'\right), & \text{if }_{\text{Weight Mode=Cosine}} \\ \sqrt{\dfrac{\left\|\rho_i' - \rho_j'\right\|^2}{C}}, & \text{otherwise} \end{cases} \tag{6}$$

$$d_{\text{Spatial}} = \sqrt{\frac{(\mathbf{x}_i' - \mathbf{x}_j')^2 + (\mathbf{y}_i' - \mathbf{y}_j')^2}{2}} \tag{7}$$

where ρ' refers to the spectral features, C is the number of channels, and $\mathbf{x}'$, $\mathbf{y}'$ refer to the pixel coordinates in the image. All values are normalized in the range [0, 1]. The spectral distance is determined by subtracting the cosine similarity from 1 when the weight mode is set to cosine. Otherwise, it will be defined by the rescaled L^2-norm equation. The spatial distance is consistently defined by the L^2-norm (7).

C. Graph Convolutional Encoder

The graph convolutional encoder is composed of graph convolutional layers and pooling layers. The encoder weights are shared for the pretraining and classification stages. Our model takes the subgraph structure as input and outputs an embedding $\mathbf{z}$. We use unlabeled data in the pretraining stage. Our final goal is to extract the discriminative features to be used for classification.

1) Propagation Rule in GCN Layers: With the adjacency matrix $\mathbf{A}_{i,j}$, each GCN layer can be propagated according to the equations as follows:

$$\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j} \tag{8}$$

$$\mathbf{L} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{9}$$

where $\mathbf{D}$ is the diagonal matrix that sums up each column of $\mathbf{A}_{i,j}$. $\mathbf{I}$ is the identity matrix. The output of the $l+1$ layer can be calculated below

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{L}\mathbf{H}^{(l)}\mathbf{W}^{(l)} + \mathbf{H}^{(0)}\right) \tag{10}$$

where $\mathbf{H}^{(l)}$ is the output from the lth layer, $\mathbf{W}^{(l)}$ refers to the trainable weight of the lth layer, and $\mathbf{H}^{(0)}$ is the biases left from layer 0. The PReLU activation function $\sigma(\cdot)$ will be applied following graph convolution.

2) Backbone of Graph Convolutional Encoder: As illustrated in Fig. 4, the subgraphs are input to the encoder $f(\cdot)$. Batch normalization (BN) $\mathbf{BN}(\cdot)$ follows the GCN layer, and a global sum pooling layer aggregates the nodes within subgraphs. Unlike computer vision backbones, GNNs do not benefit from having deep layers. Performance usually peaks when the network has 2 to 3 layers [54]. We adopt parallel branches and concatenate every output from the forward propagation to get deeper networks. Ultimately, a linear projection reduces the hidden dimension of embedding to a size of 1024. The process is formalized as follows:

$$\mathbf{z} = \phi_{\text{Linear}}\left(\begin{bmatrix} \sum_1^s(\mathbf{BN}(\mathbf{H}^{(1)})) \\ \sum_1^s(\mathbf{BN}(\mathbf{H}^{(2)})) \\ \cdots \\ \sum_1^s(\mathbf{BN}(\mathbf{H}^{(L)})) \end{bmatrix}^{\mathrm{T}}\right) \tag{11}$$

where $\mathbf{z}$ is the output embedding, $\phi_{\text{Linear}}(\cdot)$ indicates the linear projection, s is the number of superpixels in one subgraph, L is the total number of layers, and $\mathbf{H}^{(L)}$ is the output of layer L.

D. Contrastive Loss

1) Loss Function: Let View i and View j be the views from one subgraph generated using different augmentations. The encoder $f(\cdot)$ converts the graph structure to compressed embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$. A contrastive loss is calculated based on the cosine similarity between two embeddings $\mathbf{z}_i$, $\mathbf{z}_j$ in (12). The terms $\mathbf{z}_{n,i}$ and $\mathbf{z}_{n,j}$ are from the same subgraph n (positive pairs); $\mathbf{z}_{n,i}$ and $\mathbf{z}_{n',j}$ are from different subgraphs (negative pairs). Thus $\mathbf{z}_+$ and $\mathbf{z}_-$ can be calculated using the equations below

$$\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^{\mathrm{T}} \cdot \mathbf{z}_j}{\|\mathbf{z}_i\|\|\mathbf{z}_j\|} \tag{12}$$

$$\mathbf{z}_+ = \mathrm{sim}(\mathbf{z}_{n,i}, \mathbf{z}_{n,j}) \tag{13}$$
$$\mathbf{z}_- = f_{\text{Filtering}}(\mathrm{sim}(\mathbf{z}_{n,i}, \mathbf{z}_{n',j})). \tag{14}$$

We use the InfoNCE loss [55] defined below

$$\mathcal{L}_n = -\log\left(\frac{\exp(\mathbf{z}_+/\tau)}{\exp(\mathbf{z}_+/\tau) + \sum_{1,n\neq n'}^{N}\exp(\mathbf{z}_-/\tau)}\right) \tag{15}$$

$$\mathcal{L}_n = \underbrace{-\log(\exp(\mathbf{z}_+/\tau))}_{\mathcal{L}_n+} + \underbrace{\log\left(\exp(\mathbf{z}_+/\tau) + \sum_{1,n\neq n'}^{N}\exp(\mathbf{z}_-/\tau)\right)}_{\mathcal{L}_n-} \tag{16}$$

where N is the total number of subgraphs, and τ denotes the temperature parameter that varies from 0 to 1.

Equations (15) and (16) decompose the loss function into two terms. The left part represents the positive loss. Contrastive loss will be smaller when we decrease the distance between features from positive pairs during training. On the other hand, the right part corresponds to the negative loss, wherein a smaller contrastive loss is attained by increasing the distance between features from negative pairs during training. Both terms contribute to minimizing the overall pretraining loss. It indicates the importance of having reliable positive and negative pairs for ensuring stable pretraining.

2) False Negative Pair Filtering: The positive pairs consist entirely of true positives, since they are generated from the same subgraphs. However, in the negative pairs, there are some false negative pairs due to the pretraining stage being based on unlabeled data. True negative pairs represent views from different subgraphs belonging to different classes, while potential positive pairs (false negative pairs) [56] refer to views from different subgraphs that belong to the same class. Maximizing the distance between false negative pairs provides a contradictory training signal that can harm the model. In our experiments, training is based on unlabeled data, which means there is no corresponding label to infer false negative pairs. Therefore, it is difficult to separate true negative pairs precisely. Inspired by prograph CL (ProGCL) [57] and iterative similarity filtering [58], we filter the potential positive pairs that are higher than a threshold λ to ensure they have no influence on the loss function. The filtering function $f_{\text{Filtering}}(\cdot)$ is shown below

$$f_{\text{Filtering}}(\cdot) = \begin{cases} 0, & \text{if } \mathrm{sim}(\mathbf{z}_{n,i}, \mathbf{z}_{n',j}) > \lambda \\ \mathrm{sim}(\mathbf{z}_{n,i}, \mathbf{z}_{n',j}), & \text{otherwise.} \end{cases} \tag{17}$$

The histogram of the similarity distribution for potential positive and true negative pairs in Indian Pines dataset is depicted in Fig. 5, offering insights into the distribution of the two curves. True negative pairs are predominantly found in the lower range of similarity (blue columns), while potential positive pairs exhibit higher values (red columns). By setting the threshold at the intersection of these distribution curves, we can differentiate between the two categories. This approach enables us to eliminate potential positive pairs with high similarity and retain a sufficient number of negative pairs for training at the same time.

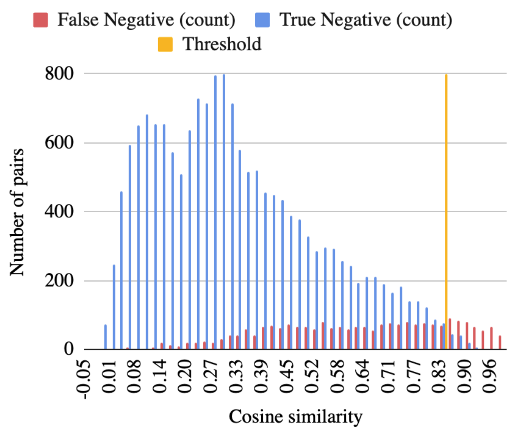


Fig. 5. Distribution histogram of similarity for all negative pairs and the filtering threshold.

In Fig. 5, the threshold is approximately 0.85. However, it is essential to consider the range of similarity when determining the specific numerical value. We conduct a group of experiments with different λ_n regarding each dataset in Section IV, where λ_n means after normalization. We use [Indian Pines: 0.83, Pavia University: 0.90, MDAS: 0.93] as the reference parameter for filtering false negative pairs within each batch. The threshold λ is consequently adopted as follows:

$$\lambda = \lambda_n \cdot (\max(\mathrm{sim}))) - \min(\mathrm{sim})) + \min(\mathrm{sim}) \tag{18}$$

where sim is the similarity matrix of negative pairs before dropout, while $\max(\cdot)$ and $\min(\cdot)$ indicate the maximum and minimum value within the matrix.

E. Weak–Strong Augmentation

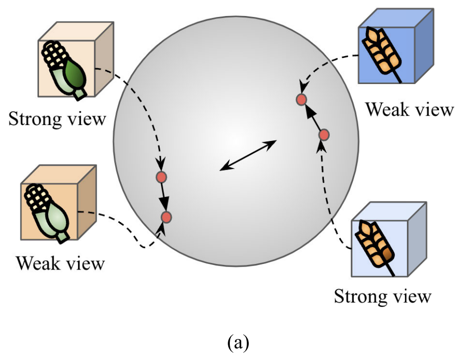
In CL, augmentations are used to generate two views from the same subgraph. In this study, we specifically focus on augmentations at the subgraph level to ensure that the embeddings with and without augmentations differ from each other. We aim to introduce realistic transformations on the node, edge, and attribute levels.

1) Node Level: Uniform sampling randomly drops out a specific ratio of nodes using a uniform distribution, preserving the semantic meaning of the graph. Random walk sampling, or subgraphing, involves a small window randomly traversing the graph and selecting a portion as the remaining graph.

2) Edge Level: Edge perturbations randomly add or remove some edges in the graph. They will mostly keep the semantic meaning of the graph. We use a uniform distribution to add or drop each edge. The number of nodes does not change.

3) Attribute Level: Attribute masking randomly selects certain nodes and masks their attributes to the mean value. Gaussian noise randomly introduces noise to selected attributes, using a standard deviation of 0.05 to simulate pepper noise. This perturbation ensures reasonable attribute variation without overwriting the original signal trends.

These augmentations can be categorized into weak and strong groups. Strong augmentations modify the original graphs more, while weak augmentations make fewer changes. Using only one type of augmentation method for every subgraph can decrease training performance, because two negative



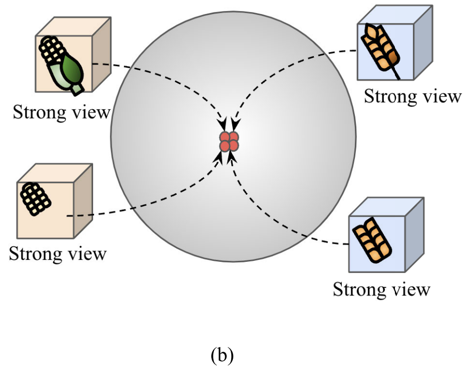


Fig. 6. Illustration of (a) weak–strong augmentation and (b) model collapsing.

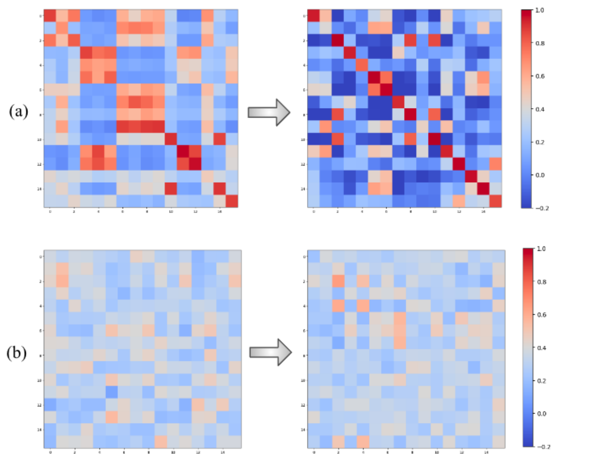


Fig. 7. Similarity matrix for 16 subgraphs before and after training (a) with weak–strong augmentation and (b) with strong–strong augmentation.

TABLE I
OVERVIEW OF WEAK AND STRONG AUGMENTATIONS

Category	Augmentation type	Augmentation ratio
Weak	Edge perturbation	5%
	Gaussian noise	5%
Strong	Edge perturbation	40%
	Uniform sampling	40%
	Subgraph	40%
	Gaussian noise	40%

pairs using the same augmentation are likely to be too similar. To address this, we randomly select the augmentation method for each input subgraph. Furthermore, if strong augmentation is applied to both views, it may randomly remove identifiable features of the subgraph. As shown in Fig. 6, if we imagine the feature space as a hypersphere, the aggressive augmentations can lead to collapse. This could result in the collapse of the pretrained model, identifying all pairs as positive. In this study, we use strong augmentation for one view and weak augmentation for the other. We stop the gradient for embedding $\mathbf{z}_i$ from weak augmentations. This ensures that the encoder makes strong views close to weak views during training. This approach, as depicted in Fig. 6(a), allows the loss function to effectively bring positive pairs closer by pushing negative pairs.

To gain a clearer understanding of model collapse from a vector perspective, we visualize the similarity matrix for 16 subgraph pairs. In the matrix, diagonals represent the similarity of positive pairs, while the triangular areas represent negative pairs. The left column in Fig. 7 represents the matrix before training, and the right column is after training. An effective weak–strong model is expected to increase the similarity of positive pairs and decrease the similarity of negative pairs. Ideally, the diagonal should become more red, while other areas turn more blue. In the case of unsuitable strong–strong augmentation, as depicted in Fig. 7(b), the diagonal of the similarity matrix does not behave as we expected. A model with inappropriate augmentations fails to identify valid negative pairs, resulting in all pairs becoming negative after training. Weak–weak and none–strong augmentations will not lead to model collapse as "strong–strong" augmentations. However, it cannot achieve better classification results than our proposed model, as shown in the ablation study (Section IV-K). The weak–strong augmentations used in our proposed model are presented in Table I.

F. Classification

1) Classifiers: A lightweight classifier is trained on the representations learned by the pretrained model. We experiment with SVM [59], linear classification (Linear) [60], and RF [61]. A lightweight classification protocol helps mitigate overfitting and reduces the computational cost.

2) Evaluation: To evaluate the quality of results, we visualize the prediction map and directly compute the performance. We also evaluate the results with metrics such as the overall accuracy (OA), average accuracy (AA), and kappa coefficient. In matrix representations, columns typically correspond to the ground truth, while rows represent classification results. The OA is the ratio of correctly classified pixels to the total number of pixels. AA is the ratio of pixels correctly classified into class i (diagonal value) to the total number of class i's pixels (sum of the ith row in the confusion matrix). The kappa coefficient measures the ratio of error reduction between classification and completely random classification.

IV. EXPERIMENTS AND ANALYSIS

A. Datasets

1) Indian Pines: The Indian Pines dataset [26] is a well-known HSI classification dataset captured by the AVIRIS sensor. It comprises 145 $\times$ 145 pixels with 200 bands each and a spatial resolution of 20 m. The ground truth for this

TABLE II
CLASSES AND TRAINING SAMPLES OF THE INDIAN PINES DATASET

	Indian Pines		
ID	Class	Total	Train
1	Alfalfa	46	21
2	Corn-notill	1428	753
3	Corn-mintill	830	426
4	Corn	237	138
5	Grass-pasture	483	209
6	Grass-trees	730	376
7	Grass-pasture-mowed	28	26
8	Hay-windrowed	478	228
9	Oats	20	10
10	Soybean-notill	972	469
11	Soybean-mintill	2455	1390
12	Soybean-clean	593	311
13	Wheat	205	125
14	Woods	1265	720
15	Buildings-Grass-Trees-Drives	386	287
16	Stone-Steel-Towers	93	49
	Total	10249	5538

TABLE III
CLASSES AND TRAINING SAMPLES OF THE PAVIA UNIVERSITY DATASET

	Pavia University		
ID	Class	Total	Train
1	Asphalt	6631	20
2	Meadows	18649	20
3	Gravel	2099	20
4	Trees	3064	20
5	Metal Sheets	1345	20
6	Bare Soil	5029	20
7	Bitumen	1330	20
8	Bricks	3682	20
9	Shadows	947	20
	Total	42776	180

dataset includes 16 classes representing the Indian Pines Forest in Indiana, USA, as detailed in Table II. The RGB-colored HSI and corresponding ground truth images are illustrated in Fig. 8(a) and (c).

2) Pavia University: The Pavia University dataset [27] is another widely used HSI classification dataset, acquired using the reflective optics system imaging spectrometer (ROSIS-3) sensor. It encompasses 103 bands and has a spatial resolution of 610 $\times$ 340 pixels with a 1.3 m spatial resolution. The ground truth for this dataset comprises nine classes representing Pavia University in Pavia, Italy, as outlined in Table III. The RGB-colored HSI and corresponding ground truth images are displayed in Fig. 9(a) and (c).

3) MDAS: The new multimodal benchmark dataset for remote sensing-MDAS [28] is a recently introduced benchmark dataset encompassing the city of Augsburg, Germany. It includes HySpex hyperspectral imagery, simulated EnMAP hyperspectral imagery, and OpenStreetMap (OSM) data. The wavelength range of the HSIs captured by the airborne imaging spectrometer system, HySpex, varies from 420 to 2450 nm. The ground truth map, obtained by merging the land use and water layers from the OSM benchmark, features 14 classes detailed in Table IV. This study utilizes a 1 m spatial resolution subarea 3 of HySpex with 368 bands. The RGB-colored HSI and corresponding ground truth images are presented in Fig. 10(a) and (c).

TABLE IV
CLASSES AND TRAINING SAMPLES OF THE MDAS DATASET

	MDAS		
ID	Class	Total	Train
1	Water	57266	20
2	Forest	71841	20
3	Residential	730694	20
4	Industrial	118631	20
5	Farm	667777	20
6	Cemetery	10719	20
7	Allotments	46359	20
8	Meadow	93225	20
9	Commercial	55842	20
10	Recreation ground	6084	20
11	Retail	20811	20
12	Orchard	12	6
13	Scrub	44498	20
14	Grass	12191	20
	Total	1935950	270

These three datasets are designed to address different aspects of ground truth data. Indian Pines primarily focuses on distinguishing various types of crops, including those growing in different seasons. The Pavia University dataset involves the classification of buildings, building shadows, and green vegetation within the university, presenting a moderately challenging scenario. The MDAS dataset centers around land use classification, requiring the distinction among commercial, residential, and industrial areas. Classifying these areas using hyperspectral data poses a relatively challenging task, as the spectral differences between different buildings can be subtle.

B. Experimental Settings

The proposed model is implemented in PyTorch [62]. The entire image is transformed into subgraphs following the strategy outlined in Section III-B. The learning rate is set to 5e^{-7}, and the batch size is fixed as 64. The SGD optimizer is employed, and the maximum number of pretraining epochs is set to 400.

For comparative analysis, we train several baseline models on the three datasets: 1-D CNN [63], 2-D CNN [64], MiniGCN [37], FuNet-A, M, C [37], graph U-Net [40], DSR-GCN [65], SPFormer [18], and a supervised GCN model. The settings for 1-D CNN, 2-D CNN, MiniGCN, and FuNet-A, M, C are adapted from the respective paper [37]. The 1-D CNN employs a 1-D convolutional layer with a filter size of 128, incorporating BN and ReLU activation. The 2-D CNN consists of three 2-D convolutional blocks, each comprising a 2-D convolutional layer, a BN layer, a max-pooling layer, and a ReLU activation layer. The hidden dimensions for the three layers are 3 $\times$ 3 $\times$ 32, 3 $\times$ 3 $\times$ 64, and 1 $\times$ 1 $\times$ 128. MiniGCN utilizes three graph convolutional blocks, each consisting of a graph convolutional layer, a BN layer, a max-pooling layer, and a ReLU activation layer. FuNet combines 2-D CNN and MiniGCN as a fusion model. The graph U-Net model parameters align with the specifications in its original paper [40], using 100 as the segmentation scale for the Indian Pines dataset and 200 for the Pavia University and MDAS datasets. The DSR-GCN and SPFormer model

TABLE V
CLASSIFICATION RESULTS OF ALL METHODS ON THE INDIAN PINES DATASET (RESULTS IN BOLD DENOTE THE BEST PERFORMANCE FOR EACH ROW)

Class	1DCNN	2DCNN	MiniGCN	FuNet-Best	Graph U-net	DSR-GCN	SPFormer	Supervised GCN	WSGraphCL
1 (%)	93.33 ± 4.62	86.67 ±6.11	76.00 ± 6.93	85.33 ± 8.33	41.03 ± 2.22	90.67 ± 8.33	90.67 ± 4.62	5.33 ± 9.24	**94.67** ± 6.11
2 (%)	80.15 ±1.71	74.72 ± 0.62	46.96 ± 0.26	79.75 ± 2.15	59.95 ± 10.51	76.99 ±1.56	78.32 ± 0.82	66.96 ± 4.70	**87,11** ± 7.62
3 (%)	53.80 ± 14.53	64.11 ± 6.22	63.28 ± 5.29	73.60 ± 1.68	73.28 ± 6.45	70.46 ± 0.87	75.17 ± 2.25	64.36 ± 15.24	**81.60** ± 4.74
4 (%)	29.63 ± 1.17	37.37 ± 4.63	37.71 ± 2.92	42.76 ± 4.98	76.73 ± 19.01	80.81 ± 15.25	78.79 ± 2.02	59.93 ± 14.19	**84.85** ±5.25
5 (%)	93.67 ± 0.56	82.85 ± 4.11	96.84 ±0.42	87.96 ±0.73	80.01 ± 4.61	**97.08** ± 2.03	92.46 ± 1.11	77.13 ± 4.62	91.12 ±5.73
6 (%)	98.59 ± 0.75	94.63 ± 2.31	**99.06** ± 0.16	98.12 ± 0.16	69.62 ± 2.83	87.19 ± 0.65	91.53 ± 4.72	87.57 ± 1.41	95.57 ±3.58
7 (%)	**100.00** ± 0.00	**100.00** ± 0.00	0.00 ± 0.00	**100.00** ± 0.00	**100.00** ±0.00	**100.00** ± 0.00	**100.00** ± 0.00	0.00 ± 0.00	**100.00** ±0.00
8 (%)	**100.00** ± 0.00	97.47 ± 1.22	**100.00** ± 0.00	89.73 ± 2.44	83.59 ± 18.71	99.87 ± 0.23	**100.00** ± 0.00	99.73 ± 0.46	**100.00** ± 0.00
9 (%)	96.67 ± 5.77	83.33 ± 5.77	0.00 ± 0.00	**100.00** ± 0.00	3.33 ± 5.77	33.33 ±5.77	83.33 ± 5.77	0.00 ± 0.00	33.33 ±57.74
10 (%)	72.76 ± 7.09	75.08 ± 2.50	**93.37** ± 0.46	74.62 ± 4.41	44.78 ± 14.50	68.26 ± 7.73	80.45 ±6.50	73.69 ± 3.76	70.97 ± 16.55
11 (%)	88.73 ± 1.08	82.47 ± 0.91	78.09 ± 2.06	84.73 ± 1.86	93.70 ±3.76	**95.02** ± 2.97	84.04 ± 1.81	85.92 ± 8.40	85.20 ± 7.12
12 (%)	88.06 ± 1.14	58.63 ± 4.56	71.16 ± 4.09	62.06 ± 3.38	69.88 ± 4.86	87.23 ± 13.38	**93.97** ± 1.55	59.57 ± 20.70	89.13 ± 7.84
13 (%)	98.33 ± 0.72	99.17± 1.44	97.50 ±0.00	**99.58** ± 0.72	56.98 ±9.08	73.33 ± 11.81	67.50 ± 8.66	73.33 ± 24.51	89.17 ± 12.83
14 (%)	91.62 ± 2.62	94.43 ± 0.38	67.89 ± 1.91	93.33 ± 0.91	95.37 ± 4.17	97.92 ± 0.11	**99.63** ± 0.18	95.84 ± 3.44	95.54 ± 2.84
15 (%)	**93.60** ± 0.58	61.95 ± 5.56	53.87 ±1.17	73.06 ± 5.08	18.69 ± 4.41	10.10 ±0.00	10.10 ± 0.00	55.22 ±40.40	21.89 ±20.41
16 (%)	**100.00** ± 0.00	95.39 ± 2.43	99.24 ± 1.31	100.00 ± 0.00	90.58 ± 16.32	73.48 ± 13.89	57.58 ± 9.19	62.12 ± 38.66	85.61 ± 4.73
OA (%)	83.90 ± 2.08	79.74 ± 0.21	75.19 ± 0.22	82.39 ± 0.23	74.50 ± 0.96	84.33 ± 0.56	84.22 ± 0.56	77.69 ± 3.40	**85.66** ± 1.95
AA (%)	**86.18** ± 2.13	80.52 ± 0.68	67.56 ±0.68	84.04 ±0.35	66.09 ± 1.97	77.61 ± 0.93	80.22 ± 1.14	60.42 ± 4.82	81.61 ± 5.95
KPP (%)	81.55 ± 2.46	76.94 ± 0.21	72.06 ± 0.16	79.94 ± 0.25	70.92 ± 1.14	82.13 ± 0.61	82.12 ± 0.62	74.55 ± 3.83	**83.76** ± 2.22

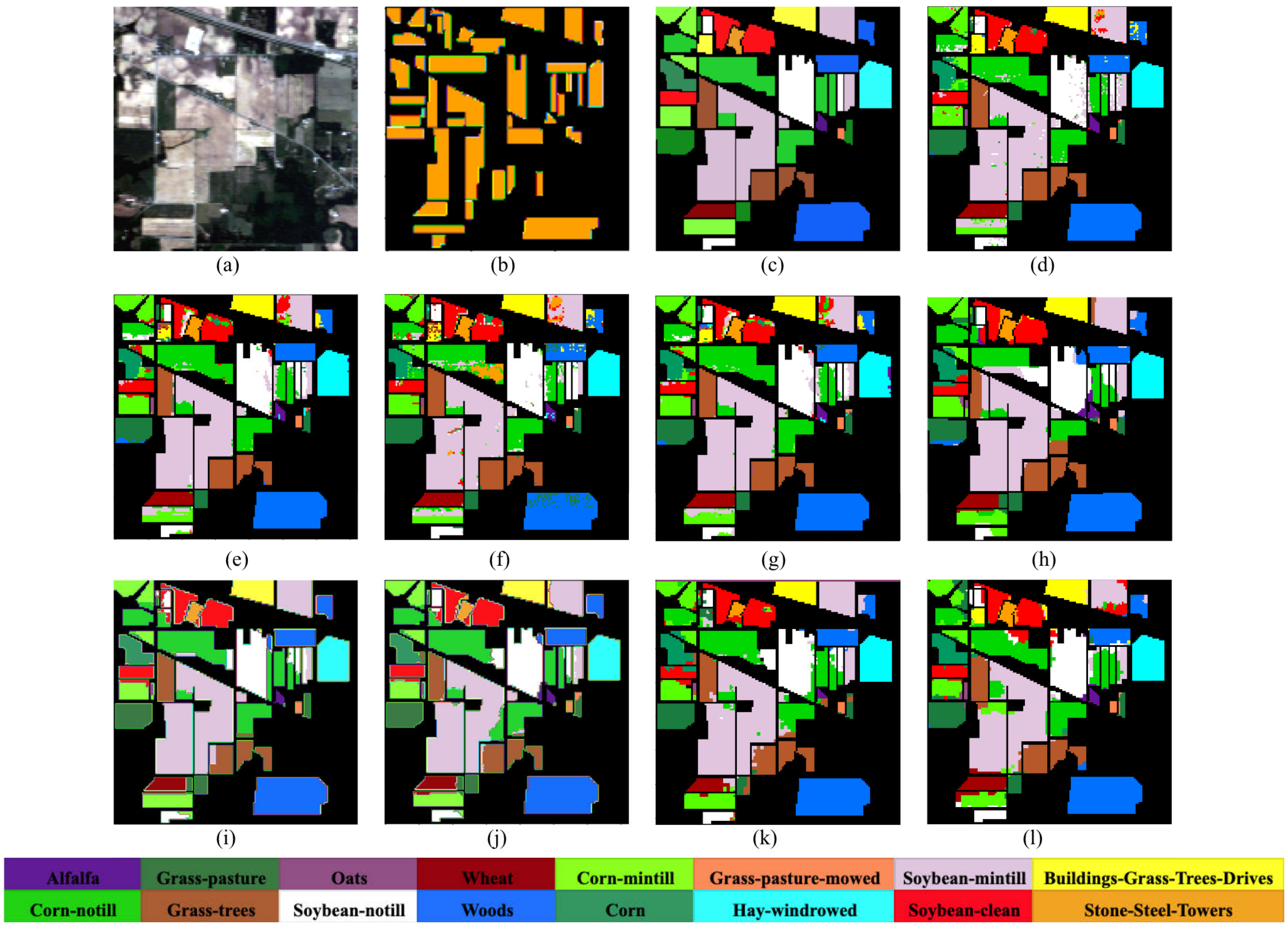


Fig. 8. Classification maps on the Indian Pines dataset of comparable models. (a) RGB representation of HSI (b) Map of training samples. (c) Map of ground truth. (d) Map of 1-D CNN result. (e) Map of 2-D CNN result. (f) Map of MiniGCN result. (g) Map of FuNet-Best result. (h) Map of graph U-Net result. (i) Map of DSR-GCN result. (j) Map of SPFormer result. (k) Map of supervised GCN result. (l) Map of WSGraphCL method result.

parameters follow the best settings in its original paper [18] and [65]. To give enough time for training the DSR-GCN model, we set the maximum epochs as 500–800 regarding loss curves Additionally, a supervised GCN model is implemented with the same graph construction method and two graph convolutional blocks, each containing one GCN layer, a BN layer, and a dropout layer with a ratio of 0.2. All baseline models are trained for 400 epochs with a fixed learning rate.

C. Classification Results

1) Results on the Indian Pines Dataset: The quantitative performance of various methods is reported in terms of OA, AA, kappa (κ), and class-specific accuracy. Table V demonstrates that the WSGraphCL model in this article outperforms the compared methods in OA, AA, and κ for the Indian Pines dataset. Notably, five classes achieve 100% accuracy, and none exhibit poor accuracy.

TABLE VI
CLASSIFICATION RESULTS OF ALL METHODS ON THE PAVIA UNIVERSITY DATASET (RESULTS IN BOLD DENOTE THE BEST PERFORMANCE FOR EACH ROW)

Class	1DCNN	2DCNN	MiniGCN	FuNet-Best	Graph U-net	DSR-GCN	SPFormer	Supervised GCN	WSGraphCL
1 (%)	64.49 ± 4.34	83.06 ± 0.23	60.46 ± 2.66	86.06 ± 0.55	80.65 ± 6.61	92.12 ± 3.97	87.30 ± 0.30	87.75 ± 5.97	**96.99** ± 0.57
2 (%)	66.23 ± 3.33	80.44 ± 0.47	65.86 ± 2.48	77.51 ± 1.55	77.44 ± 4.85	93.34 ± 4.14	78.25 ± 1.54	92.00 ± 0.59	**94.06** ± 1.82
3 (%)	73.09 ± 3.19	73.53 ± 1.78	76.30 ± 0.03	76.27 ± 0.90	95.69 ± 2.81	**96.28** ± 3.56	88.30 ± 0.96	83.88 ± 6.95	92.78 ± 1.67
4 (%)	95.22 ± 0.33	96.23 ± 0.38	95.87 ± 0.12	**96.52** ± 0.16	74.54 ± 10.96	81.07 ± 3.82	96.27 ± 1.72	77.38 ± 15.57	92.30 ± 3.12
5 (%)	96.60 ± 0.72	97.91 ± 0.24	97.45 ± 0.24	99.26 ± 0.00	92.14 ± 2.79	99.98 ± 0.04	**100.00** ± 0.00	95.34 ± 1.20	96.05 ± 0.99
6 (%)	74.09 ± 3.81	60.68 ± 0.88	86.60 ± 1.96	73.60 ± 1.15	98.16 ± 0.87	99.53 ± 0.00	**100.00** ± 0.00	83.99 ± 6.48	98.01 ± 0.73
7 (%)	90.77 ± 2.75	72.75 ± 0.88	92.91 ± 1.09	75.80 ± 1.02	95.85 ± 1.11	98.85 ± 1.16	**99.97** ± 0.04	88.19 ± 3.59	97.65 ± 0.46
8 (%)	72.80 ± 5.66	79.26 ± 2.40	80.02 ± 0.71	73.16 ± 3.30	92.35 ± 4.53	93.69 ± 1.85	**97.29** ± 1.33	79.01 ± 7.93	95.92 ± 3.83
9 (%)	98.47 ± 1.11	99.57 ± 0.23	99.30 ± 0.00	99.54 ± 0.06	69.45 ± 4.69	92.71 ± 1.63	**99.68** ± 0.37	65.58 ± 23.67	86.91 ± 4.62
OA (%)	72.49 ± 1.15	80.08 ± 0.20	74.12 ± 1.05	80.56 ± 0.36	83.24 ± 1.01	93.45 ± 0.94	87.43 ± 0.70	87.13 ± 2.97	**94.95** ± 1.02
AA (%)	81.31 ± 0.53	82.60 ± 0.04	83.86 ± 0.28	84.19 ± 0.27	86.25 ± 2.35	94.17 ± 0.59	94.12 ± 0.21	83.37 ± 4.63	**94.52** ± 0.92
KPP (%)	65.86 ± 1.24	74.40 ± 0.24	68.09 ± 1.14	75.25 ± 0.39	78.83 ± 1.31	91.49 ± 1.17	84.14 ± 0.84	83.06 ± 3.84	**93.43** ± 1.31

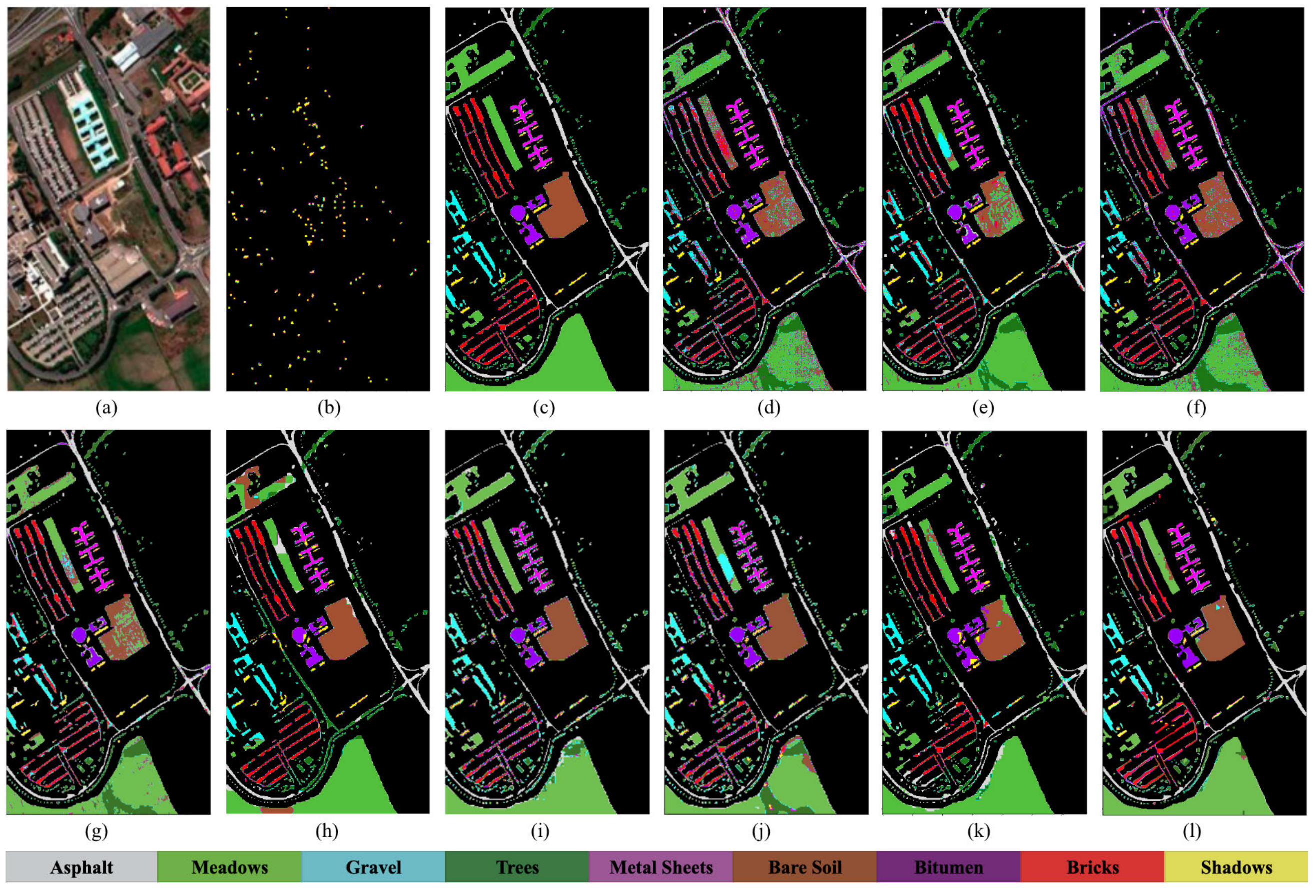


Fig. 9. Classification maps on Pavia University dataset of comparable models. (a) RGB representation of HSI. (b) Map of training samples. (c) Map of ground truth. (d) Map of 1-D CNN result. (e) Map of 2-D CNN result. (f) Map of MiniGCN result. (g) Map of FuNet-Best result. (h) Map of graph U-Net result. (i) Map of DSR-GCN result. (j) Map of SPFormer result. (k) Map of supervised GCN result. (l) Map of WSGraphCL method result.

The classification prediction results are visually depicted in Fig. 8. For the Indian Pines dataset, Fig. 8(d)–(g) presents experimental results using the same models as [37], while Fig. 8(h)–(k) displays results from the graph U-Net [40], DSR-GCN [65], SPFormer [18], and supervised GCN models. Given the small size of Indian Pines, in order to avoid label leakage due to the spatial context with random selection, we evaluated the proposed model by partitioning training and testing into two distinct areas, as suggested by [66]. This approach helps reduce the impact of aggregation, and the proposed method demonstrates favorable results even under such conditions. We also conducted experiments by randomly selecting training samples per class, as detailed in Section IV-J.

2) Results on the Pavia University Dataset: The quantitative results of nine methods on the Pavia University dataset are presented in Table VI. The proposed method exhibits significantly better performance across various metrics and stands out as the only method with no poor accuracy (below 86%) for any class. The best accuracy column in each row is bold, as the proposed method achieves the best accuracy in four classes.

In terms of visual comparison, as depicted in Fig. 9(i), DSR-GCN and the proposed method closely resemble the ground truth map. The proposed method shows fewer errors for the classes in metal sheets and bricks. However, some areas are still misclassified by the model, such as meadows. This aligns

TABLE VII

CLASSIFICATION RESULTS OF ALL METHODS ON THE MDAS DATASET (RESULTS IN BOLD DENOTE THE BEST PERFORMANCE FOR EACH ROW)

Class	1DCNN	2DCNN	MiniGCN	FuNet-Best	Graph U-net	DSR-GCN	SPFormer	Supervised GCN	WSGraphCL
1 (%)	95.68 ± 0.01	91.50 ± 0.21	**96.17** ± 0.03	93.62 ± 1.16	90.29 ± 4.36	94.70 ± 0.06	95.45 ± 0.09	93.67 ± 0.97	94.11 ± 1.04
2 (%)	54.68 ± 0.77	16.46 ± 0.57	55.36 ± 0.72	30.36 ± 10.27	62.48 ± 8.84	**71.33** ± 10.42	63.34 ± 0.44	42.06 ± 35.37	70.46 ± 2.23
3 (%)	24.24 ± 0.75	13.61 ± 0.51	19.76 ± 0.33	21.28 ± 2.65	61.02 ± 8.28	69.31 ± 3.10	64.83 ± 1.48	**78.19** ± 1.37	73.97 ± 0.45
4 (%)	22.28 ± 0.39	15.96 ± 1.40	15.12 ± 0.41	12.38 ± 2.36	67.41 ± 9.70	**76.95** ± 4.37	45.04 ± 2.73	35.97 ± 12.21	63.76 ± 5.83
5 (%)	67.51 ± 0.64	39.95 ± 0.27	67.30 ± 0.55	45.56 ± 2.81	46.79 ± 4.75	61.51 ± 4.45	64.57 ± 0.79	65.96 ± 4.91	**72.09** ± 0.82
6 (%)	43.10 ± 0.09	21.57 ± 0.56	35.68 ± 0.87	25.66 ± 1.47	88.06 ± 11.90	**98.47** ± 2.16	88.64 ± 2.63	61.53 ± 41.70	97.55 ± 3.93
7 (%)	65.44 ± 3.59	22.28 ± 1.03	55.19 ± 0.97	36.43 ± 11.67	74.44 ± 10.83	**94.01** ± 0.99	90.52 ± 0.73	52.67 ± 18.25	89.55 ± 1.90
8 (%)	54.74 ± 1.72	30.89 ± 0.19	**66.85** ± 0.16	37.20 ± 0.55	47.90 ± 6.17	55.15 ± 2.37	60.65 ± 2.04	35.77 ± 2.44	39.90 ± 0.24
9 (%)	52.64 ± 0.09	30.40 ± 0.68	56.51 ± 0.50	40.20 ± 0.33	67.95 ± 19.07	**78.30** ± 12.21	54.78 ± 1.82	66.02 ± 6.87	75.35 ± 4.04
10 (%)	59.48 ± 1.42	51.38 ± 0.34	38.47 ± 1.43	52.05 ± 0.66	89.83 ± 6.90	**92.97** ± 1.73	88.75 ± 1.49	57.83 ± 22.21	85.09 ± 1.81
11 (%)	56.23 ± 0.01	25.45 ± 0.96	56.97 ± 0.74	32.42 ± 0.87	69.22 ± 2.29	**85.07** ± 2.03	77.17 ± 0.96	39.20 ± 28.63	80.59 ± 2.51
12 (%)	**100.00** ± 0.00	**100.00** ± 0.00	**100.00** ± 0.00	**100.00** ± 0.00	41.67 ± 14.43	**100.00** ± 0.00	**100.00** ± 0.00	**100.00** ± 0.00	0.00 ± 0.00
13 (%)	28.36 ± 2.37	21.52 ± 0.06	8.72 ± 0.34	25.57 ± 1.00	35.96 ± 7.56	35.27 ± 0.45	**52.60** ± 2.98	20.87 ± 10.45	31.36 ± 0.56
14 (%)	42.49 ± 1.11	17.69 ± 0.13	29.08 ± 0.78	20.27 ± 2.54	46.71 ± 10.60	**70.46** ± 3.78	67.96 ± 1.16	30.68 ± 13.93	63.81 ± 1.05
OA (%)	46.33 ± 0.56	27.27 ± 0.05	43.98 ± 0.08	33.59 ± 0.25	56.97 ± 2.84	67.71 ± 0.46	64.58 ± 0.18	65.31 ± 4.25	**71.12** ± 0.23
AA (%)	55.21 ± 0.17	36.31 ± 1.06	50.61 ± 0.86	41.65 ± 1.00	63.55 ± 2.79	**77.39** ± 1.17	72.45 ± 0.11	55.74 ± 12.37	66.97 ± 0.78
KPP (%)	42.10 ± 0.45	19.81 ± 0.03	39.93 ± 0.05	27.74 ± 0.57	53.94 ± 3.60	65.33 ± 0.80	61.22 ± 0.11	59.72 ± 6.32	**67.90** ± 0.23

TABLE VIII

CLASSIFICATION RESULTS WITH 20 LABELS PER CLASS OF DIFFERENT NUMBERS OF SUPERPIXELS (1) ON INDIAN PINES, PAVIA UNIVERSITY, AND MDAS DATASETS

Indian Pines		Pavia University		MDAS	
(1) Num of superpixels	OA	Num of superpixels	OA	Num of superpixels	OA
464	60.27%	1002	82.10%	1049	65.76%
1168	78.71%	5248	86.51%	8279	70.46%
2160	88.07%	11917	88.41%	25084	71.00%
4918	90.54%	21850	**95.78%**	42738	**71.34%**
5823	**93.31%**	49906	89.49%	85632	70.59%
21025	91.75%	207400	86.63%	134623	69.17%

*Note: Results in bold denote the best performance for the particular dataset.

with the RGB map of the HSI data shown in Fig. 9(a), where bare soil is also represented as a meadow according to the ground truth map.

3) Results on the MDAS Dataset: As shown in Table VII, the classification results of all investigated methods decreased significantly compared with the other two datasets in terms of the quantitative criteria. The MDAS dataset focuses on land use and contains detailed categories like industrial and commercial areas. Every method has perfect results on water in the first row of our table, as water is spectrally very distinct from buildings. However, it is difficult to distinguish the classes with similar spectral characteristics. Our proposed method effectively addresses this difficulty and achieves the best performance by considering spatial information.

Not surprisingly, the CNN and GCN baselines perform poorly on the MDAS dataset, as depicted in Fig. 10(d)–(g). Graph U-Net's result exhibits noticeable errors in Fig. 10(h). In contrast, the proposed method works well in the industrial and commercial areas. Commercial buildings are usually located in the city center, and industrial buildings are usually located at a distance from the city center. Moreover, GNNs typically demand substantial GPU resources for training. The other precise GCN models, such as graph U-Net and DSR-GCN, require GPUs with larger memory. DSR-GCN must train at least 500–800 epochs to get good performance. In contrast, our proposed model can already achieve accuracy similar to our papers, with only 40 epochs. We included the table of training (encoder and classifier) and testing time required for each model in Fig. 11, measured on NVIDIA GeForce 3090 GPUs. The models' performance and maximum activation GPU memory are visualized together to get an overall view. WSGraphCL and DSR-GCN are the top two models performing better than others. However, WSGraphCL requires less GPU memory and total time. As the largest dataset, MDAS serves as a benchmark for evaluating the efficiency of various models.

The results on the three datasets demonstrate the adaptability of our model to diverse categories and scenes.

D. Impact of Different Number of Hops

As the effectiveness of our graph representation relies on the construction of input subgraphs, it is crucial to explore the potential performance improvements achievable through the adjustment of hyperparameters. As explained in Section III-A2, the number of hops illustrates how many neighbors will be extracted in one subgraph. More neighbors lead to a larger spatial area being treated as a single input data. Additionally, in the context of K-hops, it signifies that a subgraph will encompass 10^k nodes when there are 10 neighbors per node. Too many hops will slow down the model significantly and dilute the local information.

We vary the hops number from 1 to 10 for the three datasets, as shown in Table IX(2). The Indian Pines and MDAS datasets achieve optimal accuracy when utilizing 6–7 hops, benefiting from objects with a higher pixel count, such as cropland and residential areas. A larger subgraph can effectively extract more semantic information in these cases. In contrast, our model performs better on the Pavia University dataset with 4 hops. In this dataset, single objects, like buildings or shadows, consist of fewer pixels. Therefore,

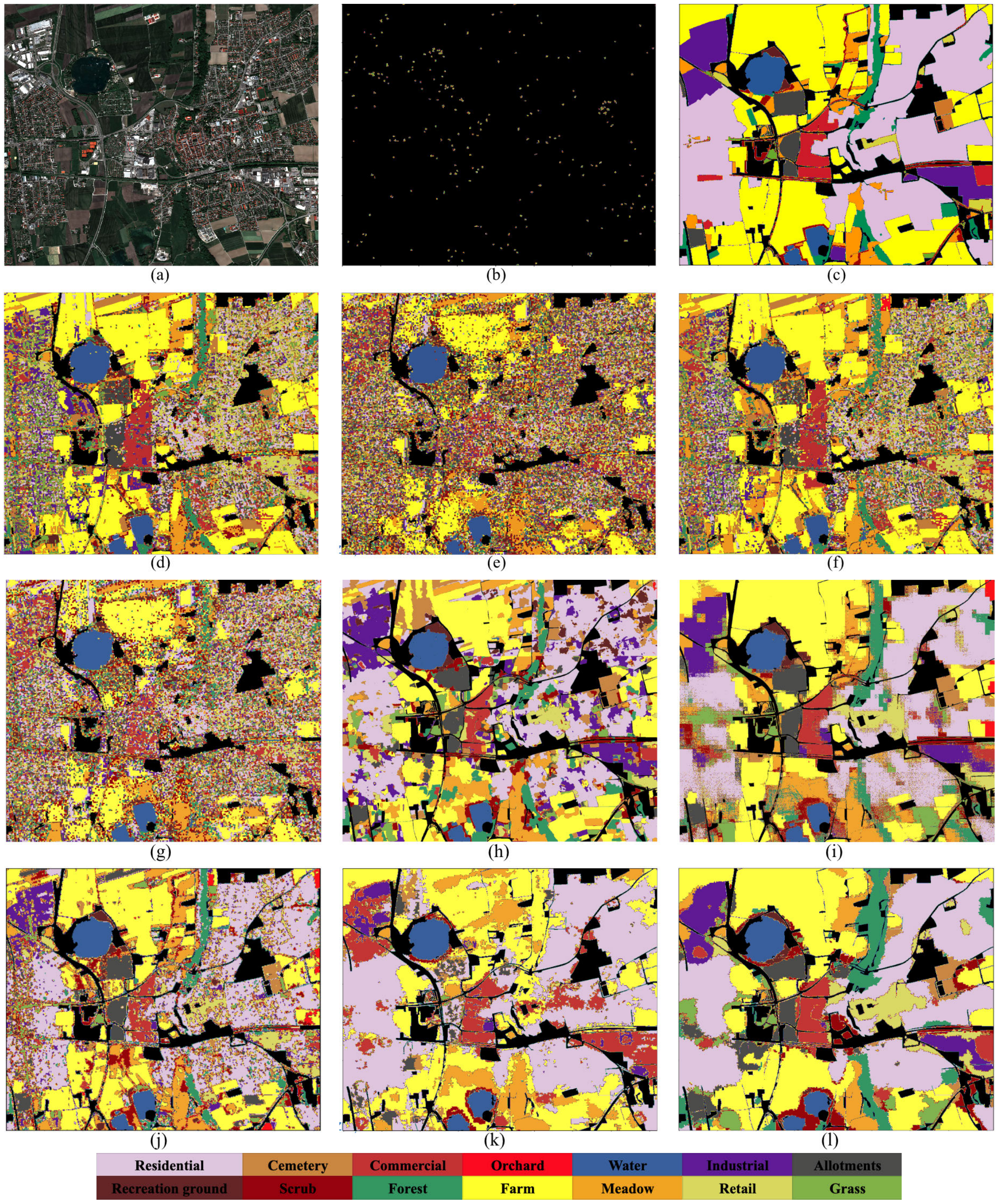
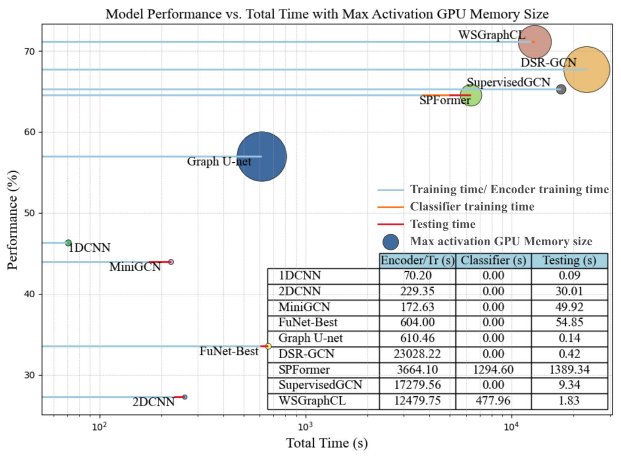


Fig. 10. Classification maps on MDAS dataset of comparable models. (a) RGB representation of HSI. (b) Map of training samples. (c) Map of ground truth. (d) Map of 1-D CNN result. (e) Map of 2-D CNN result. (f) Map of MiniGCN result. (g) Map of FuNet-Best result. (h) Map of graph U-Net result. (i) Map of DSR-GCN result. (j) Map of SPFormer result. (k) Map of supervised GCN result. (l) Map of WSGraphCL method result.

a smaller subgraph suffices, preventing the inclusion of nodes from different classes in a larger subgraph.

E. Impact of the Number of the Superpixels

Utilizing superpixels offers the advantage of reducing input dimensionality and expediting convergence. By computing mean values from pixels within each superpixel for node attributes, the model gains increased robustness to noise. Nevertheless, having too few superpixels results in the merger of pixels from different classes within a single superpixel, causing a loss of details in the map, such as roads and other small structures like metal sheets.



	Encoder/Tr (s)	Classifier (s)	Testing (s)
1DCNN	70.20	0.00	0.09
2DCNN	229.35	0.00	30.01
MiniGCN	172.63	0.00	49.92
FuNet-Best	604.00	0.00	54.85
Graph U-net	610.46	0.00	0.14
DSR-GCN	23028.22	0.00	0.42
SPFormer	3664.10	1294.60	1389.34
SupervisedGCN	17279.56	0.00	9.34
WSGraphCL	12479.75	477.96	1.83

Fig. 11. Efficiency and performance of different models on MDAS dataset.

For this purpose, we illustrate the evolving trend in OA for various numbers of superpixels in Table VIII. The model achieves its highest accuracy when the Indian Pines dataset has 5823 superpixels, the Pavia University dataset has 21 850 superpixels, and the MDAS dataset has 42 738 superpixels. Most importantly, the last row of Indian Pines and Pavia University datasets is the accuracy without superpixels (number of superpixels equals pixels). It indicates that superpixels help to not only accelerate the model but also yield better performance.

F. Impact of the Edge Weight

As mentioned in Section III-B, the edge weight mode and scale of spatial distance can control the connectivity of the nodes in subgraphs. We systematically investigated the model's performance under different implementations, as detailed in Table IX(4). The weight mode, defining the kernel in the adjacency matrix calculation, plays a critical role in representing connections between nodes. Notably, the results reveal that the Indian Pines dataset achieves optimal outcomes with the binary mode, the Pavia University dataset is better suited for the cosine mode, and the MDAS dataset exhibits improved performance with the Eudist mode, as shown in Table IX(4).

Furthermore, the performance of our model is notably responsive to the scale of spatial distance, as demonstrated by varying the scale of spatial information η, in the range [0–1], as shown in Table IX(5). The scale dictates the interplay between spectral and spatial information: at a scale of 0, only spectral information influences the edge attribute, while at a scale of 1, the edge attribute exclusively has spatial information.

G. Impact of Network Depth

The number of network layers is one of the crucial parameters in deep learning methods. We examined the influence of network depth by varying it from 1 to 10. The addition of more layers leads to prolonged computational times and exacerbates overfitting issues. The results in Table IX(6) indicate that performance improves with deeper networks. Notably, the Indian Pines and MDAS datasets achieve the best accuracy with seven–eight layers, while the Pavia University dataset attains its peak accuracy with five layers. Unlike the graphs in other contexts, like social media, our task converts geographic images to graphs. The pixels are highly correlated with respect to spatial information. As a result, we can benefit from an extended spatial context with a deeper network, like other researchers who choose larger patch sizes for CNNs on these benchmarks.

TABLE IX

Classification Results With 20 Labels Per Class of Different (2) Numbers of Hops, (3) λ, (4) Different Weight Modes, (5) Scale of Spatial, (6) Network Depth, and (7) Classifiers on Indian Pines, Pavia University, and MDAS Datasets

	Indian Pines	Pavia University	MDAS
(2) Number of hops		OA	
1	77.23%	85.80%	51.78%
2	82.05%	87.87%	60.11%
3	85.28%	93.33%	61.94%
4	89.17%	**95.78%**	69.38%
5	92.20%	89.40%	69.51%
6	**93.31%**	91.64%	69.37%
7	89.79%	92.04%	**71.34%**
8	86.95%	90.95%	70.54%
9	84.83%	88.68%	69.75%
10	86.56%	87.60%	69.40%
(3) Lamda		OA	
0.8	92.04%	92.32%	69.76%
0.83	**93.31%**	92.41%	69.26%
0.87	92.05%	93.83%	69.97%
0.9	91.57%	**95.78%**	69.53%
0.93	92.00%	93.27%	**71.34%**
0.97	91.72%	93.20%	70.38%
1	90.32%	91.38%	69.04%
(4) Weight Mode		OA	
HeatKernel	92.26%	92.26%	69.32%
Binary	**93.31%**	90.41%	69.83%
Eudist	92.53%	91.73%	**71.34%**
Cosine	88.48%	**95.78%**	69.83%
(5) Scale of Spatial		OA	
0	55.13%	76.84%	38.76%
0.1	82.63%	89.76%	51.94%
0.2	86.84%	89.67%	53.61%
0.3	85.89%	90.35%	59.30%
0.4	92.09%	91.28%	59.99%
0.5	91.67%	90.80%	58.67%
0.6	**93.31%**	92.47%	63.61%
0.7	90.30%	**95.26%**	64.80%
0.8	89.30%	90.35%	66.02%
0.9	88.54%	91.84%	70.88%
1	87.10%	89.99%	**71.34%**
(6) Network Deepth		OA	
1	84.47%	92.15%	63.25%
2	88.21%	91.54%	68.20%
3	89.12%	91.80%	68.49%
4	91.34%	89.48%	67.08%
5	91.14%	**95.78%**	70.62%
6	91.94%	94.56%	69.39%
7	92.58%	94.08%	**71.34%**
8	**93.31%**	94.34%	69.92%
9	92.49%	92.96%	69.85%
10	92.32%	93.66%	69.10%
(7) Classifier		OA	
SVM	90.53%	88.20%	67.32%
Linear	**93.31%**	**95.78%**	66.84%
RF	88.39%	86.03%	**71.34%**

*Note: Results in bold denote the best performance for the particular dataset.

H. Impact of Different Classifiers

To analyze the impact of the downstream classifier, we evaluated the performance of SVM, Linear, and RF, as shown in Table IX(7). Our article focuses on addressing real-world

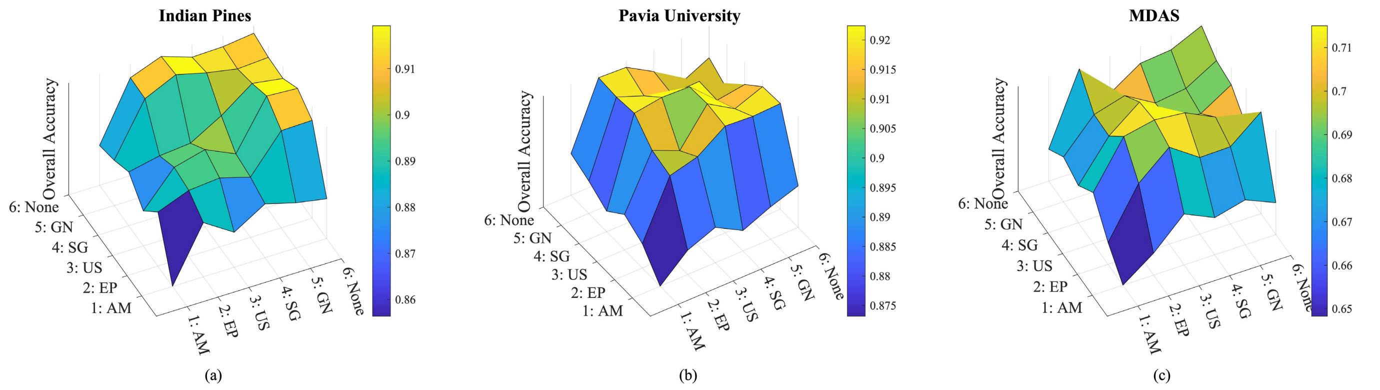


Fig. 12. Quantitative performance comparison of different augmentations with 20 labels per class in terms of OA on (a) Indian Pines, (b) Pavia University, and (c) MDAS datasets. AM—Attribute masking, EP—Edge perturbation, US—Uniform sampling, SG—Subgraph, GN—Gaussian noise, and None—No augmentation.

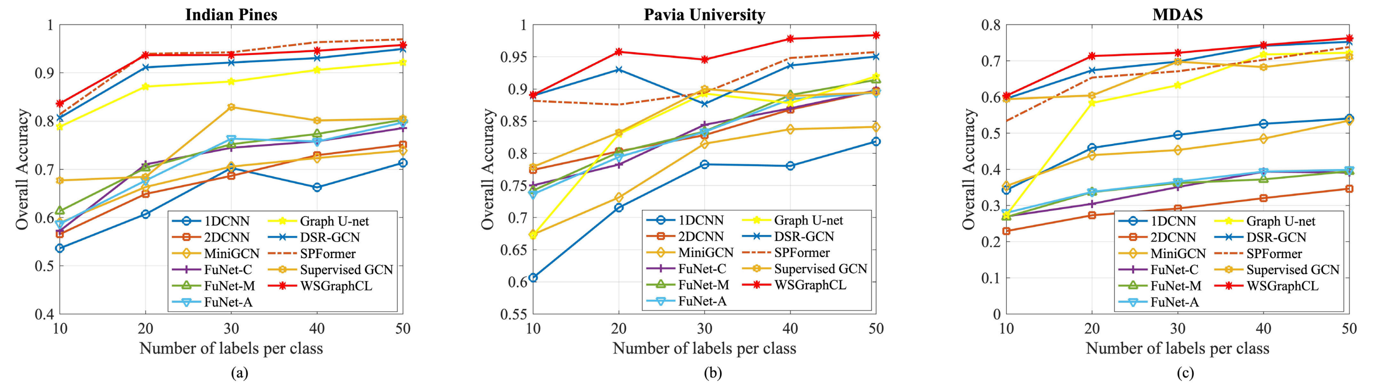


Fig. 13. Comparison of state-of-the-art models with different numbers of training samples per class on (a) Indian Pines, (b) Pavia University, and (c) MDAS datasets.

challenges in HSI classification. We evaluate the entire model rather than just the SSL encoder. The fully connected Linear classifier demonstrated superior performance on the Indian Pines and Pavia University datasets. Conversely, the MDAS dataset exhibited enhanced results with the RF classifier [67], [68]. These findings underscore the dataset-specific efficacy of different classifiers, emphasizing the importance of selecting an appropriate classifier based on the characteristics of the dataset at hand.

I. Impact of Different Augmentations

Utilizing suitable augmentations is a key point to get a stable pretrained encoder and avoid the model collapse. We performed a group of experiments set to test our model on each single augmentation, as shown in Fig. 12. We also tracked the effect on the similarity matrix and pretraining loss. As we expected, attribute masking performs poorly for every dataset, as it removes a portion of the spectral characteristics. Hence, we did not apply this augmentation to our weak–strong approach. All the other augmentations' performance is at the same level. In addition, without any augmentation for our model (none–none), we still obtain reasonable accuracy. This finding can be explained by the effect of the negative pairs in the loss, which contribute to learning useful features even when the positive pairs are not informative.

J. Impact of Number of Training Samples Per Class

To assess the stability of our proposed model in scenarios with limited labeled data, we analyzed the classification accuracy for both our method and competing models across varying numbers of samples per class. The number of training samples was systematically set to 10, 20, 30, 40, and 50. The results depicted in Fig. 13 reveal the superior accuracy achieved by our proposed method across all three datasets. Notably, the proposed method demonstrates remarkable efficiency by achieving a 90% accuracy with only 20 labels for the Indian Pines dataset and ten labels for the Pavia University dataset. This observation highlights the capacity of our method to significantly reduce the demand for labels, presenting a practical advantage for datasets with limited annotated samples. We also plot the supervised baseline as a blue line allowing for a direct visual comparison. The gap between the supervised and proposed plots signifies the improvement gained through our pretraining strategy.

K. Ablation Study

The proposed method encompasses two key components: graph construction and pretraining. An ablation study is conducted to dissect the individual contributions of each component in terms of OA. As detailed in Table X, "w/o K-hop subgraph" denotes the exclusion of the K-hop subgraph construction approach. The "w/o Spatial weight" case involves

TABLE X
EFFECTIVENESS OF THE DIFFERENT MODULES OF THE WSGRAPHCL APPROACH ON INDIAN PINES, PAVIA UNIVERSITY, AND MDAS DATASETS WITH 20 LABELS PER CLASS IN TERMS OF OA

Stage	Methods	Indian Pines	Pavia University	MDAS
Graph construction	w/o K-hop subgraph	77.23%	85.80%	51.78%
	w/o Spatial weight	55.13%	76.84%	38.76%
	w/o Superpixels	91.75%	86.63%	69.17%
Pre-training	w Supervised	68.44%	83.23%	60.40%
	w Strong-strong	82.91%	84.06%	70.36%
	w Weak-weak	91.67%	90.37%	70.94%
	w None-strong	92.72%	94.05%	70.30%
	w/o False negative pair filtering	90.32%	91.38%	69.04%
-	WSGraphCL	**93.31%**	**95.78%**	**71.34%**

*Note: Results in bold denote the best performance for the particular dataset.

setting the scale of spatial information to 0; "w/o Superpixels" entails training at the pixel level by omitting the SLIC algorithm. The "w Supervised" case indicates the removal of the pretraining encoder, training only a general GCN model with the same graph construction approaches. "w Strong-strong," "w Weak-weak," and "w None-strong" represent utilizing unsuitable augmentations. The "w/o False negative pair filtering" case entails removing the filtering module of false negative pairs. As can be seen in Table X, our proposed method obtains better accuracy step by step, emphasizing that our proposed method consistently outperforms alternative configurations. This analysis demonstrated that every part of the model makes a noticeable contribution to the performance, with particular emphasis on the spectral–spatial adjacency matrix and the pretrained encoder. In conclusion, the graph construction approach and the contrastive pretraining phase are the key factors behind our method's enhanced performance.

V. CONCLUSION

In this article, we propose a weak–strong graph CL model called WSGraphCL. WSGraphCL exhibits improved performance over other state-of-the-art approaches, yielding an increase in OA ranging from 0.45% to 2.23% points for the Indian Pines dataset, 0.1% to 4.56% points for the Pavia University dataset, and 0.19% to 3.95% points for the MDAS dataset. Our work includes a systematic analysis of hyperparameters, accompanied by an ablation study to identify the individual contributions of each model component. Unlike traditional methods that often rely on predefined graph structures, our approach dynamically constructs the graph size and graph edges based on the specific dataset. This adaptability allows for a more accurate representation of complex HSI data. K-hop subgraphs can extract features from the region regarding the size of ground objects. Our proposed adjacency matrix approach can define graph edges with spectral and spatial information. Besides, weak–strong augmentations stabilized the pretraining model. In the future, we plan to investigate various augmentation strategies. We also aim to explore diverse SSL architectures, including transformers and U-Nets, to enhance model accuracy in HSI classification. Furthermore, we believe that constructing new HSI classification datasets is an important research direction. This will allow us to evaluate and enhance the adaptability and generalization of deep learning models.

ACKNOWLEDGMENT

The authors would like to thank the Hyperspectral Image (HSI) Analysis group and the NSF Funded Center for Airborne Laser Mapping (NCALM) at the University of Houston for providing the datasets used in this article, and the IEEE GRSS Data Fusion Technical Committee for organizing the 2018 Data Fusion Contest. They would also like to thank the group of Research Repository at Purdue University for providing the dataset Indian Pines, Telecommunications and Remote Sensing Laboratory at the Pavia University for providing the dataset Pavia University, and the group of Data Science of Earth Observation at the Technical University of Munich for providing the dataset MDAS used in this article. They would also like to thank Prof. Danfeng Hong, Rong Chen, and Prof. Zhaohui Xue for providing the script of baseline models.

REFERENCES

[1] X. Ye, K. Sakai, H. Okamoto, and L. O. Garciano, "A ground-based hyperspectral imaging system for characterizing vegetation spectral features," *Comput. Electron. Agricult.*, vol. 63, no. 1, pp. 13–21, Aug. 2008.
[2] Y. Zhang et al., "Transfer-learning-based approach for leaf chlorophyll content estimation of winter wheat from hyperspectral data," *Remote Sens. Environ.*, vol. 267, Dec. 2021, Art. no. 112724.
[3] G. Sun, Z. Jiao, A. Zhang, F. Li, H. Fu, and Z. Li, "Hyperspectral image-based vegetation index (HSVI): A new vegetation index for urban ecological research," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 103, Dec. 2021, Art. no. 102529.
[4] M. Kirsch et al., "Integration of terrestrial and drone-borne hyperspectral and photogrammetric sensing methods for exploration mapping and mining monitoring," *Remote Sens.*, vol. 10, no. 9, p. 1366, Aug. 2018.
[5] Y. Gu, Y. Zhang, and J. Zhang, "Integration of spatial–spectral information for resolution enhancement in hyperspectral images," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 5, pp. 1347–1358, May 2008.
[6] R. A. Borsoi et al., "Spectral variability in hyperspectral data unmixing: A comprehensive review," *IEEE Geosci. Remote Sens. Mag.*, vol. 9, no. 4, pp. 223–270, Dec. 2021.
[7] A. Plaza et al., "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, pp. S110–S122, Apr. 2009.
[8] C. Rodarmel and J. Shan, "Principal component analysis for hyperspectral image classification," *Surv. Land Inf. Syst.*, vol. 62, no. 2, pp. 115–122, Jun. 2002.
[9] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
[10] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.
[11] L. Breiman, *Classification and Regression Trees*. Evanston, IL, USA: Routledge, 2017.
[12] J. A. Richards, *Remote Sensing Digital Image Analysis*, vol. 5. Cham, Switzerland: Springer, 2022.

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

[14] W. Ma, H. Zhang, M. Ma, C. Chen, and B. Hou, "ISSP-net: An interactive spatial–spectral perception network for multimodal classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 4412014.

[15] J. Lin, L. Mou, X. X. Zhu, X. Ji, and Z. J. Wang, "Attention-aware pseudo-3-D convolutional neural network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 9, pp. 7790–7802, Sep. 2021.

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

[17] D. Hong et al., "SpectralFormer: Rethinking hyperspectral image classification with transformers," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5518615.

[18] Z. Li, Z. Xue, Q. Xu, L. Zhang, T. Zhu, and M. Zhang, "SPFormer: Self-pooling transformer for few-shot hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5502019.

[19] 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, Feb. 2019.

[20] N. A. A. Braham, L. Mou, J. Chanussot, J. Mairal, and X. X. Zhu, "Self supervised learning for few shot hyperspectral image classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2022, pp. 267–270.

[21] Y. Ding et al., "Self-supervised locality preserving low-pass graph convolutional embedding for large-scale hyperspectral image clustering," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5536016.

[22] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2412–2429, Feb. 2023.

[23] Y. Wang, C. M. Albrecht, N. A. A. Braham, L. Mou, and X. X. Zhu, "Self-supervised learning in remote sensing: A review," *IEEE Geosci. Remote Sens. Mag.*, vol. 10, no. 4, pp. 213–247, Dec. 2022.

[24] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020, *arXiv:2002.05709*.

[25] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 5812–5823.

[26] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 band aviris hyperspectral image data set: June 12, 1992 Indian pine test site 3," Purdue Univ. Res. Repository, USA, 2015, doi: 10.4231/R7RX991C.

[27] A. Marinoni and P. Gamba, "A novel approach for efficient *p*-linear hyperspectral unmixing," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 6, pp. 1156–1168, Sep. 2015.

[28] J. Hu et al., "MDAS: A new multimodal benchmark dataset for remote sensing," *Earth Syst. Sci. Data Discuss.*, vol. 15, no. 1, pp. 1–26, 2022.

[29] Y. Ding, X. Zhao, Z. Zhang, W. Cai, and N. Yang, "Graph sample and aggregate-attention network for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022.

[30] 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.

[31] 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.

[32] Y. Ding, Y. Chong, S. Pan, and C. Zheng, "Diversity-connected graph convolutional network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 5518118.

[33] S. Jia, S. Jiang, S. Zhang, M. Xu, and X. Jia, "Graph-in-graph convolutional network for hyperspectral image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 1157–1171, Jan. 2024.

[34] 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.

[35] X. Tang et al., "Hyperspectral image classification based on 3-D octave convolution with spatial–spectral attention network," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 3, pp. 2430–2447, Mar. 2021.

[36] X. Tong, J. Yin, B. Han, and H. Qv, "Few-shot learning with attention-weighted graph convolutional networks for hyperspectral image classification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2020, pp. 1686–1690.

[37] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 7, pp. 5966–5978, Jul. 2021.

[38] Y. Dong, Q. Liu, B. Du, and L. Zhang, "Weighted feature fusion of convolutional neural network and graph attention network for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 31, pp. 1559–1572, 2022.

[39] Q. Liu, L. Xiao, J. Yang, and Z. Wei, "CNN-enhanced graph convolutional network with pixel- and superpixel-level feature fusion for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 10, pp. 8657–8671, Oct. 2021.

[40] R. Chen, G. Vivone, G. Li, C. Dai, and J. Chanussot, "An offset graph U-Net for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 5520615.

[41] X. Zhao, J. Niu, C. Liu, Y. Ding, and D. Hong, "Hyperspectral image classification based on graph transformer network and graph attention mechanism," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022.

[42] Y. Zhang, W. Li, M. Zhang, Y. Qu, R. Tao, and H. Qi, "Topological structure and semantic information transfer network for cross-scene hyperspectral image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 6, pp. 2817–2830, Jun. 2023.

[43] Y. Zhang, W. Li, M. Zhang, S. Wang, R. Tao, and Q. Du, "Graph information aggregation cross-domain few-shot learning for hyperspectral image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 2, pp. 1912–1925, Feb. 2024.

[44] P. Guan and E. Y. Lam, "Spatial–spectral contrastive learning for hyperspectral image classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2022, pp. 1372–1375.

[45] W. Yu, S. Wan, G. Li, J. Yang, and C. Gong, "Hyperspectral image classification with contrastive graph convolutional network," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 5530515.

[46] Z. Ye, J. Wang, T. Sun, J. Zhang, and W. Li, "Cross-domain few-shot learning based on graph convolution contrast for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5504614.

[47] B. Li, L. Fang, N. Chen, J. Kang, and J. Yue, "Enhancing hyperspectral image classification: Leveraging unsupervised information with guided group contrastive learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5504317.

[48] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[49] G. Nikolentzos, G. Dasoulas, and M. Vazirgiannis, "K-hop graph neural networks," *Neural Netw.*, vol. 130, pp. 195–205, Oct. 2020.

[50] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[51] B. Xu, H. Shen, Q. Cao, K. Cen, and X. Cheng, "Graph convolutional networks using heat kernel for semi-supervised learning," in *Proc. Twenty-Eighth Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1928–1934.

[52] P.-E. Danielsson, "Euclidean distance mapping," *Comput. Graph. Image Process.*, vol. 14, no. 3, pp. 227–248, 1980.

[53] J. Ye, "Cosine similarity measures for intuitionistic fuzzy sets and their applications," *Math. Comput. Model.*, vol. 53, no. 1, pp. 91–97, 2011.

[54] J. Wang, Y. Wang, Z. Yang, L. Yang, and Y. Guo, "Bi-GCN: Binary graph convolutional network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 1561–1570.

[55] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.

[56] J. Xia, L. Wu, G. Wang, J. Chen, and S. Z. Li, "Progcl: Rethinking hard negative mining in graph contrastive learning," 2022, *arXiv:2110.02027*.

[57] T. Huynh, S. Kornblith, M. R. Walter, M. Maire, and M. Khademi, "Boosting contrastive self-supervised learning with false negative cancellation," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 986–996.

[58] A. Tejankar, S. Abbasi Koohpayegani, V. Pillai, P. Favaro, and H. Pirsiavash, "ISD: Self-supervised learning by iterative similarity distillation," 2020, *arXiv:2012.09259*.

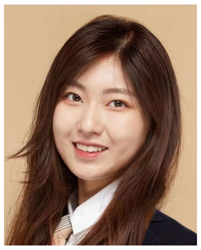
[59] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, and A. Mueller, "API design for machine learning software: Experiences from the scikit-learn project," in *Proc. ECML PKDD Workshop, Languages Data Mining Mach. Learn.*, 2013, pp. 108–122.

[60] T. M. Mitchell, "Logistic regression," *Mach. Learn.*, vol. 10, p. 701, Apr. 2005.

[61] T. K. Ho, "Random decision forests," in *Proc. IEEE 3rd Int. Conf. Anal. Recognit.*, vol. 1, Mar. 1995, pp. 278–282.

[62] M. Liu et al., "DIG: A turnkey library for diving into graph deep learning research," *J. Mach. Learn. Res.*, vol. 22, no. 240, pp. 1–9, Mar. 2021.

[63] T.-H. Hsieh and J.-F. Kiang, "Comparison of CNN algorithms on hyperspectral image classification in agricultural lands," *Sensors*, vol. 20, no. 6, p. 1734, Mar. 2020.
[64] Q. Wang, Z. Yuan, Q. Du, and X. Li, "GETNET: A general end-to-end 2-D CNN framework for hyperspectral image change detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 1, pp. 3–13, Jan. 2018.
[65] Z. Xue, Z. Liu, and M. Zhang, "DSR-GCN: Differentiated-scale restricted graph convolutional network for few-shot hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 5504918.
[66] T. Zhou et al., "ConstrastivePose: A contrastive learning approach for self-supervised feature engineering for pose estimation and behavioral classification of interacting animals," Bioinf. Inst., A*STAR, Singapore, 2022, doi: 10.1101/2022.11.09.515746.
[67] M. G. Kwak et al., "Self-supervised contrastive learning to predict the progression of Alzheimer's disease with 3D amyloid-pet," *Bioengineering*, vol. 10, no. 10, p. 1141, 2023.
[68] T. Zhou et al., "Constrastivepose: A contrastive learning approach for self-supervised feature engineering for pose estimation and behavorial classification of interacting animals," Univ. Houston, TX, USA, Tech. Rep., Nov. 2022.


Sirui Wang received the B.Sc. degree in remote sensing of science and technology from Shandong University of Science and Technology, Qingdao, China, in 2021, and the M.Sc. degree in ESPACE from the Technical University of Munich (TUM), Munich, Germany, in 2023.

She is currently a Research Associate with the Chair of Data Science in Earth Observation (SiPEO), TUM. She worked as a Research Assistant at the Future Laboratory Artificial Intelligence for Earth Observation (AI4EO), Munich, from 2022 to 2023, applying deep learning for flood detection. Her research interests include remote sensing, data science, machine and deep learning, explainable artificial intelligence (AI), and image processing.

Nassim Ait Ali Braham received the M.Sc. degree in computer science from Ecole Nationale Supérieure d'Informatique (ESI), Algiers, Algeria, in 2019, and the M.Sc. degree in artificial intelligence and data science from Université Paris Dauphine-PSL, Paris, France, in 2020. He is currently pursuing the Ph.D. degree in remote sensing with German Aerospace Center, Wessling, Germany, and the Technical University of Munich, Munich, Germany.

In 2019, he spent six months as a Research Intern at the LIRIS-CNRS Laboratory, Lyon, France. In 2020, he spent six months at the LAMSADE-CNRS Laboratory, PSL Research University, Paris. In 2023, he spent six months as a Visiting Researcher at INRIA THOTH, Grenoble, France. His research interests include deep learning, computer vision, self-supervised learning, and remote sensing.

Xiao Xiang Zhu (Fellow, IEEE) received the M.Sc., Dr.-Ing., and Habilitation degrees in signal processing from the Technical University of Munich (TUM), Munich, Germany, in 2008, 2011, and 2013, respectively.

She is the Chair Professor for Data Science in Earth Observation with TUM. She was the Founding Head of the Department "EO Data Science" at the Remote Sensing Technology Institute, German Aerospace Center (DLR), Wessling, Germany. Since May 2020, she has been the PI and Director of the International Future AI Laboratory "AI4EO–Artificial Intelligence for Earth Observation: Reasoning, Uncertainties, Ethics and Beyond," Munich. Since October 2020, she has been the Director of the Munich Data Science Institute (MDSI), TUM. From 2019 to 2022, she has been a Co-Coordinator of the Munich Data Science Research School (www.mu-ds.de) and the Head of the Helmholtz Artificial Intelligence Research Field "Aeronautics, Space and Transport." She was a Guest Scientist or Visiting Professor at the Italian National Research Council (CNR-IREA), Naples, Italy, Fudan University, Shanghai, China, the University of Tokyo, Tokyo, Japan, and the University of California at Los Angeles, Los Angeles, CA, USA in 2009, 2014, 2015, and 2016, respectively. She is currently a Visiting AI Professor with ESA's Phi-Laboratory, Frascati, Italy. Her main research interests include remote sensing and Earth observation, signal processing, machine learning, and data science, with their applications in tackling societal grand challenges, e.g., global urbanization, UN's SDGs, and climate change.

Dr. Zhu is a fellow of the Academia Europaea (the Academy of Europe), AAIA, and ELLIS. She has been a member of the Young Academy (Junge Akademie/Junges Kolleg) at the Berlin-Brandenburg Academy of Sciences and Humanities and the German National Academy of Sciences Leopoldina and the Bavarian Academy of Sciences and Humanities. She serves on the Scientific Advisory Board in several research organizations, among others the German Research Center for Geosciences (GFZ), from 2020 to 2023, and the Potsdam Institute for Climate Impact Research (PIK). She is an Associate Editor of IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING and *Pattern Recognition* and served as an Area Editor responsible for special issues of *IEEE Signal Processing Magazine* from 2021 to 2023.