Semi-Supervised Mini-Graph Convolutional Networks for Hyperspectral Image Classification

Zahra Dadashi Asiabar[1], Nasehe Jamshidpour[2], Mahdi Hasanlou[3*]
[1] School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran - zahra.dadashi40@ut.ac.ir
[2] School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran - njamshidpour@ut.ac.ir
[3] School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran - hasanlou@ut.ac.ir

Keywords: Mini-GCN, Graph Convolution Network (GCN), Semi-Supervised Learning, Hyperspectral classification.

Abstract

Hyperspectral image (HSI) classification requires models that leverage long-range spectral and spatial dependencies while handling scarce labels and the high dimensionality of the data. This paper introduces a semi-supervised Graph Convolutional Network (GCN) that builds a graph over both labeled and unlabeled pixels to better capture the data manifold. It also proposes the implementation of Mini-GCN, an inductive mini-batch variant that preserves graph reasoning with far lower memory and computational cost. On top of these, a hybrid end-to-end FuNet architecture fuses a CNN with Mini-GCN (for mid- and long-range topology) via three fusion schemes: additive, multiplicative, and concatenation, to learn complementary representations. Experiments on two benchmark datasets, Indian Pines and Pavia University, using limited labeled training samples, are compared against various conventional and Deep Learning (DL) CNN-based algorithms, and also supervised GCN. Semi-supervised GCN already outperforms CNNs and supervised GCN; Mini-GCN further enhances efficiency without compromising accuracy, and the proposed fusion networks yield the best performance. Notably, FuNet-C attains an OA of 95.79% and κ of 0.95 on Indian Pines, and OA 92.36% and κ 0.90 on Pavia University, with marked gains on minority classes, confirming that combining mini-batch graph reasoning with CNN features is an effective, label-efficient paradigm for HSI classification.

1. Introduction

Deep learning (DL) techniques, such as Convolutional Neural Networks (CNNs), have driven significant progress in fields such as image classification, natural language processing (NLP), and hyperspectral (HS) image classification due to their powerful modeling capabilities and ability to automatically extract high-level features (Hong et al., 2020). However, traditional CNNs can only effectively process data that reside in Euclidean space or grid-like structures. GCNs are introduced as an innovative technology within the field of artificial intelligence (AI) that specifically addresses the challenge of applying deep learning techniques to non-Euclidean data (Gao et al., 2018).GCNs are a prominent model type within the broader family of Graph Neural Networks (GNNs). GCNs introduce the convolution operation into the graph structure, generalizing the idea of CNNs from regular grids to irregular domains (Bhatti et al., 2023).

In GCNs, the graph structure is leveraged to aggregate node information from neighborhoods in a convolutional fashion. This process learns representations of nodes that encode both the local graph structure and the features of nodes (Zhang et al., 2019).

The creation of graph convolution operators is central to extending CNNs to graph data analysis. GCNs are generally categorized based on how the convolution operation is defined. (1) Spectral methods define graph convolutions from a spectral perspective. They are based on spectral graph theory and use the spectral decomposition of the graph Laplacian matrix to form a Fourier basis, which is analogous to the classical Fourier basis in Euclidean space (Zhang et al., 2019).

(2) Spatial methods build graph convolution in the vertex domain (or node domain). They operate directly on the graph structure, typically focusing on aggregations of node representations from the node neighborhoods (Bhatti et al., 2023). The convolution layer aggregates the representations of all adjacent nodes to generate a new feature representation for a given node (Gao et al., 2018).

In the domain of remote sensing, GCNs are specifically introduced to address the shortcomings of traditional CNNs in HSI classification (Hong et al., 2020). HSI classification involves classifying land use and land cover (LULC) using data characterized by rich and detailed spectral information, enabling fine and accurate detection of materials (Shahraki and Prasad, 2020). While CNNs are effective for capturing short-range spatial and spectral features (often using patch-wise input), GCNs excel because they can naturally model long-range spatial relations between samples (or vertices) by leveraging the graph structure, features that are often not considered by CNNs. This capability allows GCNs to effectively characterize the underlying data structure of HS images in high-dimensional space (Hong et al., 2020).

In CNN-based approaches, the input is usually selected as spatial patches, which are extracted from the hyperspectral image and then processed by the network. The output of this process is the class labels, which are generated in a one-shot manner. In contrast, GCNs operate at the pixel level and require the construction of an adjacency matrix to represent the relationships among pixels. This matrix must be computed before the training phase.

* Corresponding author

CNNs demonstrate strong capability in extracting spectral–spatial patterns within local and short-range regions, whereas GCNs, due to their graph-based structure, can model mid- and long-range dependencies among pixels. Accordingly, CNNs mainly focus on the immediate neighborhood, while GCNs represent a network of more complex relationships among samples across broader spatial scales. The proposed architecture consists of three main components: (1) graph construction from hyperspectral data, (2) multi-scale feature extraction and fusion using CNN and GCN layers to capture short-, mid-, and long-range dependencies, and (3) a semi-supervised learning module that leverages a limited number of labeled samples to propagate information throughout the graph and to generate the final classification map. This strategy enhances the model's ability to exploit long-range spatial correlations while simultaneously reducing its reliance on labeled training data.

In CNNs, training is typically performed using mini-batches, which allows for faster processing and more efficient memory utilization. However, GCNs, by default, require the use of the entire dataset in a full-batch training scheme, since the dependencies across the whole graph must be simultaneously incorporated into the network.

The computational complexity of each layer in CNNs and GCNs is mainly determined by matrix multiplication operations. For CNNs, the cost is on the order of $O(NDP)$. In contrast, GCNs introduce an additional term, resulting in complexity $O(NDP+N^2D)$, which makes the computations more expensive, especially for large graphs. Here, N, D, and P denote the number of samples, the dimensions of the input and output features, respectively.

When applying a mini-batch strategy in GCNs, the computational cost can be reduced to $O(NDP + NMD)$, where M is the mini-batch size and $M \ll N$. As a result, the computational order with respect to N becomes nearly comparable to that of CNNs.

2. Methodology

The objective of graph-based methods is to represent data in the form of a graph, and to learn the relationships among samples within a non-Euclidean space. A graph provides a powerful tool for modeling complex data structures beyond the Euclidean domain. In our problem, each pixel of the hyperspectral image is considered as a node, while the connections between pixels are defined as edges based on their spectral similarity. Accordingly, we obtain an undirected graph $\mathcal{G} = (V, E)$, where V denotes the set of nodes (pixels) and E represents the set of edges that represent the similarities between the nodes. This formulation incorporates the relational structure of the data more explicitly into the network.

To incorporate the relationships among nodes into the network computations, an adjacency matrix A is defined. Each element of this matrix represents the degree of similarity between two nodes, which is typically calculated using a similarity function, such as the radial basis function (RBF):

$$Ai, j = exp(-\frac{|| xi - xj ||^2}{\sigma^2}) \quad (1)$$

where xi and xj denote the spectral signatures associated with nodes vi and vj, and σ is the parameter controlling the width of the RBF kernel.

Subsequently, the graph's Laplacian matrix is constructed from A and the degree of each node. The Laplacian serves as the foundation for spectral computations on graphs and plays a crucial role in enhancing the generalization capability of the model:

$$L = D - A \quad (2)$$

where D is a diagonal matrix, whose entries correspond to the degrees of A (Hong et al., 2019a; Hong et al., 2019b).

To improve stability and reduce sensitivity to data variations, the normalized version of the Laplacian is often employed, which is known as the symmetric normalized Laplacian (Chung, 1997):

$$\begin{aligned} L_{sym} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \\ &= I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \end{aligned} \quad (3)$$

where I denotes the identity matrix.

2.1 Graph Convolutions in the Spectral Domain

The main idea of GCNs is analogous to classical convolutional networks, where filtering operations are applied to images. In graphs, filters can be applied to signals defined over the nodes. In the spectral domain, this operation is equivalent to multiplying the data by the normalized Laplacian matrix. For practical implementation, a simplified propagation rule is employed, in which each new layer of the network is formed as a linear combination of the features of the nodes and their neighbors. A nonlinear activation function (e.g., ReLU) is then applied to introduce nonlinearity. Through this process, the network progressively extracts and generalizes graph features in deeper layers (Hong et al., 2020).

2.2 Proposed Method

Based on the conducted analysis, we introduce an optimized version of GCNs, referred to as Mini-GCN. The Mini-GCN is designed to efficiently capture mid- and long-range spectral–spatial dependencies in hyperspectral image classification, while reducing computational cost and employing more suitable mechanisms. In the following, three different strategies for feature fusion are described, and finally, the overall end-to-end network architecture is presented. This architecture combines the Mini-GCN with a convolutional branch to enhance accuracy and robustness in semi-supervised settings.

2.3 Proposed Mini-GCNs

As the size of the graphs increases, the computational cost of GCNs grows rapidly. To mitigate this burden in large graphs, a practical and effective solution is to employ mini-batch processing, analogous to the approach commonly used in CNNs. Inspired by inductive learning (Michalski, 1983), we introduce the Mini-GCN model, which allows GCNs to be trained using mini-batches. It is noteworthy that in this inductive setting, no information from the features or graph structure of the test nodes is utilized during the training process.
Before introducing the convolutional update rule in Mini-GCN, we present a key theorem proven in (Zeng et al., 2019) to ensure the feasibility of the mini-batch strategy. Consider a complete

graph $\mathcal{G}$ with N nodes within the labeled set. We define a random sampler with a size of M, where $M \ll N$. Before each epoch, the sampler is applied repeatedly over the graph so that all nodes are selected at least once, resulting in a collection of subgraphs:

$$G = \left\{ g_S = (V_S, E_S) \middle| s = 1, \cdots, \left\lceil \frac{N}{M} \right\rceil \right\} \tag{4}$$

If a node v is selected from the subgraph Vs, an unbiased estimate of the feature of node v in the GCN layer of the full-batch $(\ell+1)^{th}$ GCN can be obtained. It is computed by aggregating the features of node v and all nodes $u \in V$ in the ℓ^{th} layer. To ensure correctness, the normalization constant is adjusted based on the number of times each node or edge appears across all sampled subgraphs.

Relying on this theorem, Mini-GCN can perform graph convolution operations on mini-batches like CNNs. The update rule within a mini-batch is defined as follows:

$$\tilde{H}_s^{(l+1)} = h(\tilde{D}_s^{-\frac{1}{2}} \tilde{A}_s \tilde{D}_s^{-\frac{1}{2}} \tilde{H}_s^{(l)} W^{(l)} + b_s^{(l)}) \tag{5}$$

Here, s denotes both the subgraph and the mini-batch during network training. In the simple case of sampling without replacement, the normalization values are set to one.

By aggregating the outputs from all mini-batches, the final output of the $(\ell + 1)^{th}$ layer is formed as:

$$H^{(l+1)} = \left[\tilde{H}_1^{(l+1)}, \ldots, \tilde{H}_s^{(l+1)}, \ldots, \tilde{H}_{\left\lceil \frac{N}{M} \right\rceil}^{(l+1)} \right] \tag{6}$$

2.4 Fusion Networks

Different network architectures are capable of extracting complementary features from hyperspectral images. For instance, CNNs excel at capturing spectral–spatial features, whereas GCNs effectively model topological relationships among samples. However, relying on a single model may not provide sufficient feature diversity and could result in suboptimal performance. To overcome this limitation, we propose a model and feature fusion approach: CNNs and Mini-GCNs are trained simultaneously in an end-to-end manner to enhance feature discriminability. Unlike traditional GCNs, our Mini-GCNs support mini-batch learning, which allows them to be seamlessly integrated with CNNs. This fused network is hereafter referred to as FuNet. Three main strategies are considered for feature fusion in FuNet:

1-Additive Fusion (A): This method performs element-wise addition of features extracted from CNN and Mini-GCN. It is simple and parameter-efficient.

2- Multiplicative Fusion (M): This approach applies an element-wise multiplication, which preserves interactions between features.

3- Concatenation Fusion (C): In this strategy, feature vectors are concatenated and passed through one or more fully connected layers. It provides the highest learning capacity but also incurs the greatest parameter complexity.

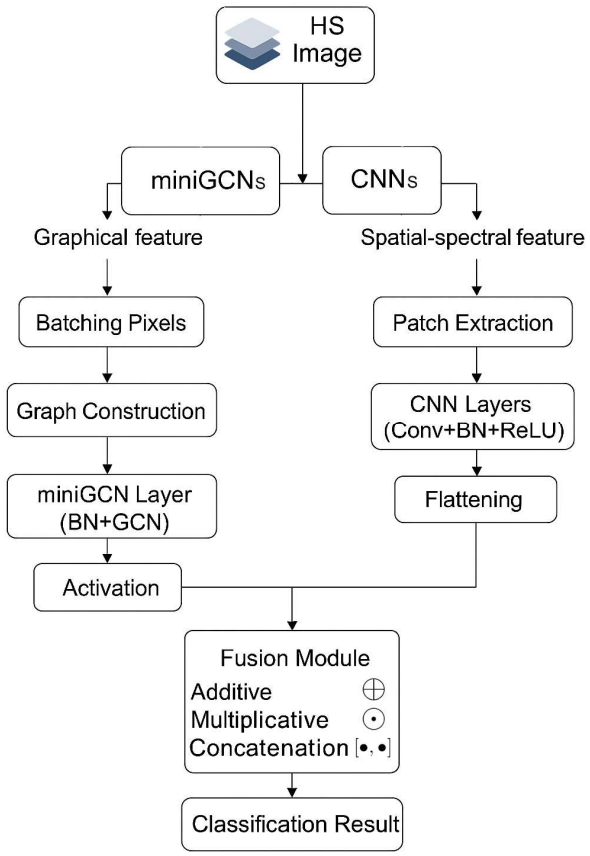

Figure 1. Overview of the final end-to-end network (FuNet), illustrating one mini-batch training iteration.

Symbolically, these three fusion strategies can be formulated as follows:

$$H_{FuNet-A}^{(l+1)} = H_{CNNs}^{(l)} \oplus H_{miniGCNs}^{(l)} \tag{7}$$

$$H_{FuNet-M}^{(l+1)} = H_{CNNs}^{(l)} \otimes H_{miniGCNs}^{(l)} \tag{8}$$

$$H_{FuNet-C}^{(l+1)} = \left[H_{CNNs}^{(l)}, H_{miniGCNs}^{(l)} \right] \tag{9}$$

where $\oplus$ = element-wise addition
$\otimes$ = element concatenation -wise multiplication
$[.;.]$
$H_{CNNs}^{(l)}, H_{miniGCNs}^{(l)}$ = represent the features extracted at the l^{th} layer from each network

Figure 1 illustrates an example of one mini-batch training iteration in FuNet, which includes a feature extraction module (with CNN and Mini-GCN) and a fusion module. The features extracted by the first module are combined using one of the three aforementioned fusion strategies and then passed to the final classifier.

3. Experiments

3.1 Data Description

To evaluate the performance of the proposed models, two well-known hyperspectral datasets were employed. The evaluations were conducted both quantitatively, using classification accuracy and other overall performance metrics, and qualitatively, through visual inspection of the classification maps.

3.1.1 Indian Pines Dataset

This dataset was acquired by the AVIRIS sensor over the northwest region of Indiana, USA. The scene consists of 145×145 pixels with a ground sampling distance of 20 meters and 220 spectral bands covering the wavelength range of 400–2500 nm, with a spectral resolution of 10 nm. To reduce noise, 35 bands—including noisy bands and water absorption bands (0–3, 102–112, 147–165, 216–220)—were removed, resulting in 185 remaining bands. The dataset contains 16 land-cover classes.

3.1.2 Pavia University Dataset

The second scene was acquired by the Reflective Optics System Imaging Spectrometer (ROSIS) sensor and consists of 340×610 pixels with a ground sampling distance of 1.3 meters. This sensor recorded 103 spectral bands within the wavelength range of 430–860 nm. The dataset contains 9 land-cover classes.

3.2 Experimental Settings

3.2.1 Implementation Details

All models used in this study were implemented in the TensorFlow environment. The Adam (Kingma, 2014) optimizer was selected for network optimization due to its stability and fast convergence in hyperspectral classification tasks. The initial learning rate was set to 0.001 and was gradually, dynamically decreased during training. Training was performed for up to 200 epochs, and the mini-batch size was set to 32 in the experiments; however, in the Mini-GCN algorithm, this choice was determined optimally.

To improve stability and prevent overfitting, Batch Normalization (BN) (Ioffe and Szegedy, 2015) with a momentum of 0.9 and L2-regularization with a coefficient of 0.001 were employed. For fine-tuning hyperparameters (e.g., learning rate and regularization coefficients), 5-fold cross-validation was used.

To demonstrate the efficiency of the proposed classification method, we used a limited number of ground truth data. For the Indian Pines dataset, which has relatively lower labeled samples, the data were randomly split into 30% for training and 10% for validation. For the Pavia University dataset, only 10% of labeled samples are used for training, and 10% for validation. In both datasets, the rest of the labeled data remains completely unseen in the training phase and is used for performance evaluation.

3.2.2 Comparison with state-of-the-art baseline methods

To demonstrate the effectiveness of the proposed models, they were compared with a set of conventional and advanced algorithms. These methods include: K-Nearest Neighbor (KNN), Random Forest (RF), Support Vector Machine (SVM), 1-D CNN, 2-D CNN, Supervised-GCN, Mini-GCN, as well as the three proposed fusion network variants (FuNet-A, FuNet-M, FuNet-C). For all traditional classification methods, including KNN, RF, and SVM, the data were first normalized. Then, the optimal hyperparameter values were selected using 5-fold cross-validation on the training and validation sets.

In the 1-D CNN approach, the standardized data were fed into a one-dimensional convolutional network with 128 filters and a kernel size of 3. To enhance generalization, Dropout with a rate of 0.3 and Batch Normalization were applied. The final layer consisted of a Dense layer with a softmax activation for class discrimination. The model was trained using the Adam optimizer with an initial learning rate of 0.001 and an adaptive learning rate reduction mechanism. To prevent overfitting, Early Stopping (with a patience of 30 epochs) was employed, and the best weights were saved. The training process was conducted for a maximum of 200 epochs with a batch size of 32. Class weights were computed in a balanced manner.

In the 2-D CNN approach, for each pixel in the hyperspectral image, a patch of size 185×7×7 was extracted. The network consists of three convolutional blocks (with 32, 64, and 64 filters), along with Batch Normalization, MaxPooling, and ReLU activation. To prevent overfitting, Dropout with a rate of 0.3 and L2 regularization were applied. The fully connected layer included 64 neurons, followed by a softmax layer for class prediction. The model was trained using the Adam optimizer with an initial learning rate of 0.001 and an adaptive learning rate reduction mechanism. The number of epochs was set to 200, and the batch size was 32.

In the Supervised GCN architecture, the graph was constructed based on spectral similarity only using the training samples (Kipf and Welling, 2016), using (K=10) (k-nearest neighbors) and a Gaussian weighting function with parameter (t=2.0). The adjacency matrix was symmetrically normalized to ensure training stability. The network consists of two graph convolutional layers; the hidden layer has 64 neurons with ReLU activation and Dropout with a rate of 0.5, and the output layer is configured according to the number of classes in the dataset.

While in the Semi-supervised GCN, the graph is constructed using all the samples, both labeled and unlabeled to take advantage of the manifold and semi-supervised learning to estimate the underlying structure of the data more accurately. The rest of the implementation details remain the same.

The Mini-GCN algorithm retains the same structure as the semi-supervised GCN but is designed with mini-batch processing, resulting in improved computational efficiency. In this approach, for each batch, the local adjacency matrix is constructed using K = 10 (k-nearest neighbors) and a Gaussian kernel, and then normalized. To select the optimal batch size, different values (16, 32, 64, 128) were evaluated. Other settings remained the same as the original GCN.

3.3 Evaluation Metrics

To quantitatively evaluate the performance of the models, three commonly used metrics were employed: Overall Accuracy (OA), Average Accuracy (AA), and the κ coefficient. Additionally, classification maps were generated for the entire image area.

4. Experimental Results

4.1 Quantitative Evaluation

The classification performance of different models on the Indian Pines and Pavia University datasets is quantitatively analyzed. As shown in the tables, traditional classifiers such as KNN, RF, and

SVM provide relatively reasonable accuracy; however, differences among them can be observed.

For Indian Pines, KNN achieves lower overall accuracy compared to RF and SVM (OA: 81.41% vs. 82.98% and 89.51%), which is likely due to the presence of noisy training samples. In contrast, for Pavia University, these models produce similar results, with SVM achieving the best performance (OA: 93.98%).

The use of deep learning models (1-D CNN, 2-D CNN, and GCN) has led to a significant improvement in classification accuracy. In particular, 2-D CNN, by extracting spatial–spectral information, performs considerably better than pixel-based models (1-D CNN).

As shown in the tables, for Indian Pines, the semi-supervised GCN provides significant improvements of approximately 7.94% in OA, 11.56% in AA, and 0.09 in κ compared to the 1-D CNN, which is attributed to better modeling of spatial relationships among samples. The supervised GCN also shows moderate gains of about 2.38% in OA and 2.32% in AA over the 1-D baseline.

On the other hand, Mini-GCN demonstrates improved performance compared to 1-D CNN and basic GCN approaches. For Indian Pines, it achieves an OA of 70.19% and κ of 0.66, while the proposed FuNet-C framework yields superior results with OA of 95.79% and κ of 0.95. Similarly, for Pavia University, semi-supervised GCN reaches OA of 86.61% and κ of 0.82, while FuNet-C achieves outstanding performance with OA of 92.36% and κ of 0.90.

The combination of models in the FuNet framework substantially enhances the accuracy, with FuNet-C achieving the highest accuracy on both datasets (Indian Pines: OA 95.79%, κ 0.95; Pavia University: OA 92.36%, κ 0.90). This represents significant improvements over the 1-D CNN baseline (Indian Pines: +22.94% OA, +0.26 κ; Pavia University: +13.90% OA, +0.19 κ).

Furthermore, for classes with limited samples, such as Alfalfa, Grass-pasture-mowed, and Oats in Indian Pines, the fusion of spatial-spectral features and graph structure in FuNet-C provides remarkable performance gains, achieving up to 100% accuracy for several challenging classes, including Corn-notill, Soybean-notill, and Stone-steel-towers.

Table 1. Quantitative comparison of different algorithms in terms of OA, AA, and κ on the Indian Pines dataset. (best results are shown in bold)

Class No.	KNN	RF	SVM	1-D CNN	2-D CNN	Supervised GCN	Semi-GCN	Mini-GCN	FuNet-A	FuNet-M	FuNet-C
1	76.89	77.29	86.06	72.61	73.21	69.92	70.03	55.75	85.81	78.60	**95.35**
2	68.32	66.44	85.45	38.01	86.99	44.51	55.09	41.92	84.80	81.20	**92.80**
3	51.83	46.95	74.39	39.63	83.54	32.62	66.67	0.71	**97.14**	87.14	94.29
4	94.83	85.92	92.24	80.46	87.93	83.56	88.26	90.60	94.63	92.62	**97.99**
5	94.46	96.94	95.98	90.63	95.22	92.41	95.31	96.88	96.88	92.86	**98.66**
6	97.67	98.83	98.83	98.54	99.71	98.63	98.63	99.32	99.32	97.28	**100.00**
7	79.65	83.19	89.23	69.91	88.05	73.15	76.25	75.73	86.55	81.38	**93.10**
8	85.65	90.10	91.72	67.88	78.41	86.09	89.87	75.22	94.60	94.06	**97.03**
9	60.70	67.44	84.65	73.95	70.93	27.37	59.62	46.07	85.87	63.59	**90.22**
10	99.33	92.62	96.64	99.33	100.00	96.06	98.43	99.21	96.88	96.88	**100.00**
11	95.81	95.47	95.81	94.37	91.94	97.55	97.30	98.97	96.39	**96.65**	98.45
12	35.34	53.76	68.05	52.63	65.79	47.37	50.88	7.02	**90.35**	82.46	89.47
13	88.06	82.09	70.15	83.58	97.01	87.72	87.72	80.70	82.76	96.55	**96.55**
14	55.26	52.63	86.84	23.68	44.74	3.03	45.45	0.00	37.50	68.75	**81.25**
15	84.21	21.05	78.95	42.11	47.37	73.33	73.33	0.00	**100.00**	62.50	87.50
16	64.29	21.43	71.43	7.14	14.29	58.33	66.67	0.00	50.00	66.67	**100.00**
OA	81.41	82.98	89.51	72.85	83.09	75.23	80.79	70.19	91.41	87.46	**95.79**
AA	77.02	70.76	85.40	64.66	76.57	66.98	76.22	54.26	86.22	83.70	**94.54**
Kappa	0.79	0.80	0.88	0.69	0.81	0.71	0.78	0.66	0.90	0.86	**0.95**

Table 2. Quantitative comparison of different algorithms in terms of OA, AA, and κ on the Pavia University dataset. (best results are shown in bold)

Class No.	KNN	RF	SVM	1-D CNN	2-D CNN	Supervised GCN	Semi-GCN	Mini-GCN	FuNet-A	FuNet-M	FuNet-C
1	86.28	92.23	93.28	95.96	86.14	89.39	89.07	82.07	87.33	85.52	**89.59**
2	97.28	97.75	98.31	94.09	93.20	98.77	95.88	98.52	**97.75**	97.59	97.27
3	54.87	69.89	79.68	42.28	61.59	15.90	65.63	13.16	56.67	62.38	**79.05**
4	80.20	90.54	92.10	81.87	91.52	75.29	83.44	82.63	83.01	85.29	85.29
5	98.68	98.68	99.59	99.59	99.34	99.35	99.35	98.79	92.59	**100.00**	97.78
6	50.12	66.34	87.94	46.83	84.47	23.02	58.99	28.51	57.85	54.67	**87.67**
7	79.70	73.18	86.72	24.23	73.60	0.09	72.65	68.61	77.44	76.69	**88.72**
8	84.64	86.51	90.43	40.89	84.07	89.78	84.56	91.38	87.50	84.51	**89.67**
9	99.88	99.88	99.77	99.41	99.18	99.74	99.74	99.34	98.95	97.89	**100.00**
OA	85.19	89.66	**93.98**	78.94	88.34	78.86	86.61	80.90	86.72	86.37	**92.36**
AA	81.29	86.11	91.98	69.46	85.90	65.70	83.26	73.67	82.12	82.73	**90.56**
Kappa	0.80	0.86	0.92	0.71	0.85	0.71	0.82	0.74	0.82	0.82	**0.90**

4.2 Visual Comparison

Visual analysis of the classification results using the generated maps also shows that pixel-based models (KNN, RF, SVM, 1-D CNN) suffer from noise (salt-and-pepper effect) and the class boundaries are not accurately delineated. Although GCN can model spatial relationships, due to full-batch training and the large graph size, it produces relatively noisy maps.

On the other hand, Mini-GCN, by preserving the local graph structure in each batch, provides more stable representations, and its maps are comparable to those of a 2-D CNN. This strategy reduces errors caused by manually computing the adjacency matrix and limits error propagation between layers.

The use of FuNet results in smoother and more detailed maps, as the fusion of diverse features enhances the model's ability to represent HS data. However, due to batch-wise processing in CNNs, some edge details may be lost, which explains why FuNet maps are slightly less sharp compared to Mini-GCN.

4.3 Parameter Sensitivity Analysis

The performance of GCN and Mini-GCN networks largely depends on the quality of the adjacency matrix (A), see Equation (1). Therefore, examining the effects of two key parameters, namely the number of neighbors (K) and the RBF kernel width σ, is crucial for optimizing model performance.

For this purpose, the Indian Pines dataset was selected for the sensitivity analysis.

The results indicate that the K parameter has the most significant impact on improving the overall accuracy (OA), and increasing K leads the models to a stable performance level. On the other hand, variations in σ only cause minor fluctuations in accuracy, indicating that the models are less sensitive to this parameter and require less precise tuning.

Specifically, Mini-GCN exhibits gradual and smooth increases or decreases in performance when these two parameters are varied, whereas the standard GCN, due to full-batch training, experiences more significant performance variations and may reach convergence bottlenecks.

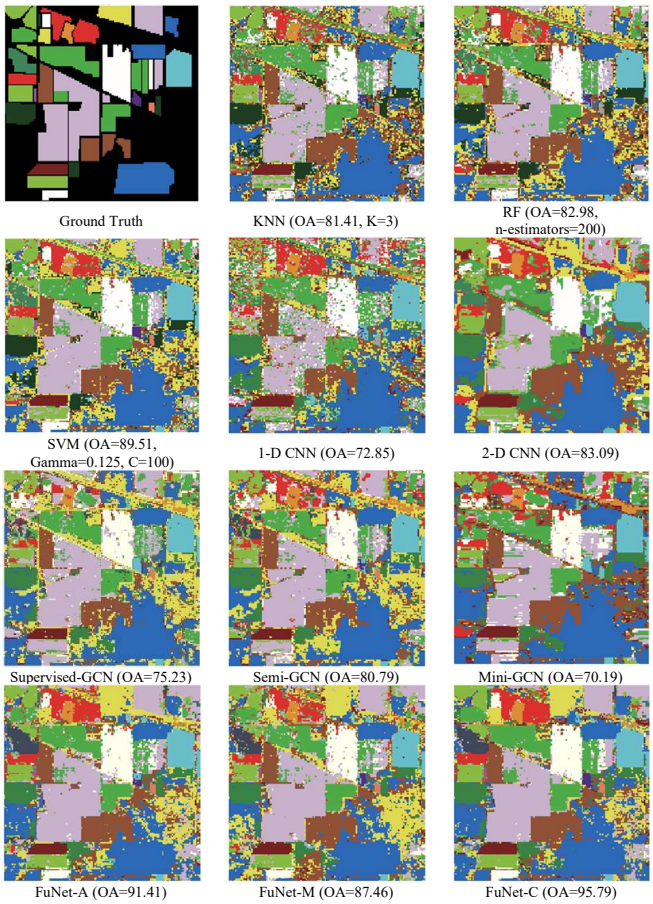

Figure 2. Classification maps obtained by different methods on the Indian Pines dataset.

Based on these results, the combination (K, σ) was selected for all experiments. This choice is not only stable but also allows straightforward application to other datasets (e.g., Pavia). The parameter sensitivity plots for both models clearly illustrate the trends in overall accuracy and provide a solid basis for optimal parameter selection in subsequent experiments.

Figure 3. Classification maps obtained by different methods on the Pavia University dataset.

5. Conclusion

GCNs are capable of modeling the structure of hyperspectral (HS) data in high-dimensional spaces; however, they face limitations such as high computational cost, large memory requirements, and gradient vanishing/exploding issues during full-batch training. To address these challenges, we introduced Mini-GCN, which allows large-scale graph networks to be trained efficiently and flexibly using mini-batch learning. In addition to reducing computational cost, this model can predict new samples without the need for retraining.

The combination of Mini-GCN with CNN (FuNet) enables the extraction of more diverse and discriminative features, providing optimal classification performance on the Indian Pines and Pavia University datasets. Experimental results demonstrate that Mini-GCN and FuNet significantly outperform traditional GCNs and single CNN models, achieving high accuracy even for challenging classes and imbalanced samples.

References

Bhatti, U.A., Tang, H., Wu, G., Marjan, S., Hussain, A., 2023: Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. *International Journal of Intelligent Systems*, 2023(1), 8342104.

Chung, F.R., 1997: *Spectral graph theory*. American Mathematical Soc., 92.

Gao, H., Wang, Z., Ji, S., 2018: Large-scale learnable graph convolutional networks. *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 1416-1424.

Hong, D., Gao, L., Yao, J., Zhang, B., Plaza, A., Chanussot, J., 2020: Graph convolutional networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 59(7), 5966-5978.

Hong, D., Yokoya, N., Chanussot, J., Zhu, X.X., 2019a: CoSpace: Common subspace learning from hyperspectral-multispectral correspondences. *IEEE Transactions on Geoscience and Remote Sensing*, 57(7), 4349-4359.

Hong, D., Yokoya, N., Ge, N., Chanussot, J., Zhu, X.X., 2019b: Learnable manifold alignment (LeMA): A semi-supervised cross-modality learning framework for land cover and land use classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 147, 193-205.

Ioffe, S., Szegedy, C., 2015: Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proc. Int. Conf. Machine Learning*, 448-456.

Kingma, D.P., 2014: *Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980*.

Kipf, T.N., Welling, M., 2016: *Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907*.

Michalski, R.S., 1983: A theory and methodology of inductive learning. *Machine Learning*, 83-134.

Shahraki, F.F., Prasad, S., 2020: Joint Spatial and Graph Convolutional Neural Networks-A Hybrid Model for Spatial-Spectral Geospatial Image Analysis. *Proc. IEEE Int. Geoscience and Remote Sensing Symposium*, 4003-4006.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V., 2019: *Graphsaint: Graph sampling based inductive learning method. arXiv preprint arXiv:1907.04931*.

Zhang, S., Tong, H., Xu, J., Maciejewski, R., 2019: Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1), 1-23.