International Journal of Remote Sensing

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tres20

HyperGCN – a multi-layer multi-exit graph neural network to enhance hyperspectral image classification

Haseena Rahmath P, Kuldeep Chaurasia, Anika Gupta & Vishal Srivastava

To cite this article: Haseena Rahmath P, Kuldeep Chaurasia, Anika Gupta & Vishal Srivastava (2024) HyperGCN – a multi-layer multi-exit graph neural network to enhance hyperspectral image classification, International Journal of Remote Sensing, 45:14, 4848-4882, DOI: 10.1080/01431161.2024.2370501

To link to this article: https://doi.org/10.1080/01431161.2024.2370501

INTERNATIONAL JOURNAL OF REMOTE SENSING
2024, VOL. 45, NO. 14, 4848–4882
https://doi.org/10.1080/01431161.2024.2370501

Taylor & Francis
Taylor & Francis Group

HyperGCN – a multi-layer multi-exit graph neural network to enhance hyperspectral image classification

Haseena Rahmath P[a], Kuldeep Chaurasia[a], Anika Gupta[a] and Vishal Srivastava[b]

[a]School of Computer Science Engineering and Technology, Bennett University, Greater Noida, UP, India; [b]Department of Computer Science & Engineering, Motilal Nehru National Institute of Technology Allahabad, Prayagraj, UP, India

ABSTRACT

Graph neural networks (GNNs) have recently garnered significant attention due to their exceptional performance across various applications, including hyperspectral (HS) image classification. However, most existing GNN-based models for HS image classification are limited depth models and often suffer from performance degradation as model depth increases. This study introduces HyperGCN, an exclusive GNN-based model designed with multiple graph convolutional layers to exploit the rich spectral information inherent in HS images, thereby enhancing classification performance. To address performance degradation, HyperGCN incorporates techniques resistant to oversmoothing into its architecture. Additionally, multiple-side exit branches are integrated into the intermediate layers of HyperGCN, enabling dynamic management of the complexity of HS images. Less complex HS images are processed by fewer layers, exiting early via attached branches, while more complex images traverse multiple layers until reaching the final output layer. Extensive experiments on four benchmark HS datasets (Indian Pines, Pavia University, Salinas, and Botswana) demonstrate HyperGCN's superior performance over basic GNN-based models. Notably, HyperGCN outperforms or performs comparably to the CNN-GNN combined model in classifying HS images. Furthermore, the superior performance of multi-exit HyperGCN over its single-exit counterpart emphasizes the effectiveness of incorporating side exit branches in GNN-based HS image classification. Compared to state-of-the-art models, multi-exit HyperGCN demonstrates competitive performance, highlighting its effectiveness in handling complex spectral information in HS images while maintaining an acceptable balance between accuracy and computational efficiency.

ARTICLE HISTORY
Received 28 January 2024
Accepted 7 June 2024

KEYWORDS
Graph convolution network; multi-layer graph neural network; multi-exit; early-exit; hyperspectral image classification; oversmoothing

1. Introduction

Hyperspectral (HS) imaging captures images at multiple wavelengths using high-resolution remote sensors (Landgrebe 2002). Unlike standard imaging, which uses red, green, and blue colours, HS imaging employs hundreds of high-resolution channels. This technique aids in object identification, material classification, and

CONTACT Haseena Rahmath P haseenarahmath@gmail.com Bennett University, Plot Nos 8-11, TechZone II, Greater Noida, UP 201310, India

process detection by precisely characterizing hues and providing detailed information (Yan et al. 2021). Over the past few decades, HS imaging has become popular in military, industrial, scientific, and agricultural fields (Harsanyi and Chang 1994; Patel and Upla 2022). Traditional classification techniques, such as the nearest neighbour classifier (Huang et al. 2016), support vector machines (Melgani and Bruzzone 2004), and logistic regression (Khodadadzadeh et al. 2014), combine spectral signatures with pattern recognition. However, these methods often neglect spatial associations, leading to errors. In the past decade, advanced deep learning techniques have been applied to HS image classification. These include Convolutional Neural Networks (CNNs) (Y. Chen et al. 2016), Recurrent Neural Networks (RNNs) (Liu et al. 2017), Generative Adversarial Networks (GANs) (Hang et al. 2020), and stacked autoencoders (Y. Chen et al. 2014). These methods effectively extract spatial-spectral features, achieving high accuracy (Y. Chen et al. 2016; Patel and Upla 2022). While CNNs excel at extracting information from short-range regions in HS images, they struggle with modelling complex relationships between samples or vertices in graphs. This limitation makes them less effective at capturing long-range spatial relations, which are crucial for comprehensive HS image analysis and classification (Hong et al. 2020).

Graph neural networks (GNNs) are an emerging family of neural networks that excel with non-Euclidean graph-structured data (Gori, Monfardini, and Scarselli 2005). Kipf and Welling (2017) introduced graph convolutional networks (GCNs), extending CNN capabilities into non-Euclidean domain. GCNs capture structural relationships between nodes, making them effective at handling long-range spatial associations within HS images and revealing their underlying geometric properties (Kipf and Welling 2017). The potential of GNNs in HS image classification has sparked increased research activity. Studies by Shahraki and Prasad (2018) and Wang and Wang (2022) integrated CNNs and GCNs for HS image classification. Qin et al. (2018) used specialized GCNs to extract spatial and spectral features, while Wan et al. (2020) employed a segmentation followed by a GCN-based approach. Hong et al. (2020) introduced MiniGCN, processing images in small batches and proposing fusion models (FuNet-A, FuNet-M, and FuNet-C) that integrate CNNs and GCNs into an end-to-end network, achieving impressive results. Liu et al. (2021) proposed a CNN-enhanced GCN model (CEGCN), where CNNs perform feature learning on small-scale regions and GCNs on large-scale regions, generating complementary spectral-spatial features. To further enhance combined GNN and CNN performance in HS image classification, Yu et al. (2023a) introduced the GPF-Net model, fusing GCN and GAT models to extract features from large, irregular regions. Additionally, Yu et al. (2023b) improved GCN feature representation using contrastive learning, developing the Contrastive Graph Convolutional Network (ConGCN) to explore supervision signals from both spectral information and spatial relations. These studies typically used a limited number of graph convolution layers (usually 2 or 3) due to observed performance degradation with more layers (Zhao and Akoglu 2020). Deep-layered models can extract valuable information from high-order neighbours (G. Li et al. 2019), but balancing this advantage with potential drawbacks is challenging. The addition of more layers, coupled with the introduction of non-linearities, may lead to performance degradation. Factors contributing to this decline include over-fitting due to an increased number of parameters, training difficulties stemming from vanishing gradients, and oversmoothing resulting from numerous graph convolution operations (Zhao and Akoglu 2020).

Oversmoothing is a phenomenon where multiple graph convolutions cause node embeddings to become indistinguishable. Li, Han, and Wu (2018) investigated this depth limitation in GCNs, revealing that deep GCNs can induce oversmoothing. As layers increase, feature representation at nodes converges to the same value, making them indistinguishable. Other studies (Kipf and Welling 2017; R. Li, Han, and Wu 2018; Pham et al. 2017; Rahimi, Cohn, and Baldwin 2018) have confirmed this. Kipf and Welling (2017) observed performance reduction with more than three layers, and Pham et al. (2017) and Rahimi, Cohn, and Baldwin (2018) demonstrated accuracy reduction with deeper layers. To address oversmoothing, recent works have proposed various strategies. Li et al. (2019) introduced residual and dense connections, as well as dilated convolutions, while Xu et al. (2018) employed dense skip connections. Rong et al. (2020) presented DropEdge, a technique that randomly eliminates a specified number of edges from the input graph during each training epoch. Chen et al. (2020) extended GCN to greater depth using initial residual and identity mapping techniques. Zhao and Akoglu (2020) introduced PairNorm, a normalization technique for GNNs. By inserting a normalization layer after the graph convolution layer, PairNorm prevents node embeddings from becoming overly similar. This approach enhances the robustness of deep GNNs against oversmoothing and enables the training of deeper models without sacrificing performance. PairNorm achieves this by preserving the total pairwise feature distances across layers, ensuring that distant pairs of nodes have less similar features. This prevents feature mixing across clusters and helps retain the unique characteristics of nodes, even in deep GNNs.

This study introduces HyperGCN, an exclusive deep GNN model designed for effective HS image classification. HyperGCN incorporates multiple graph convolution layers and is enhanced with oversmoothing-resistant techniques to ensure robust performance. Additionally, side exit branches are integrated into the intermediate layers of HyperGCN, transforming it into a multi-exit model that dynamically manages the depth and complexity in alignment with the unique characteristics of HS images. This enables early exits for less complex HS images through the added side exit branches, accelerating the inference process. Consequently, most HS images can terminate executions with accurate predictions before reaching the final output layer. This approach also reduces the risk of oversmoothing when HS images propagate through multiple graph convolution layers.

The main contributions to this article are as follows:

(1) Introduction of HyperGCN, an exclusive GNN-based model that leverages multiple graph convolutional layers to create a deep architecture, thereby significantly enhancing HS image classification performance.
(2) Implementation of two oversmoothing-resistant techniques within HyperGCN architecture, along with a comparison of their performance.
(3) Augmentation of multiple side-exit branches to the intermediate layers of HyperGCN model, transforming it into a multi-exit model that provides dynamic inferences with respect to variations in HS image complexity.

The remaining sections of the paper are organized as follows: Section 2 introduces GNNs, specifically focusing GCN. It discusses the oversmoothing problem inherent in GNNs and explores various approaches employed to mitigate it. Additionally, it provides an

overview of multi-exit neural networks. Section 3 explains the methodology employed in this study and details the proposed HyperGCN architecture. Section 4 and 5 detail the experiments and discuss the results. Finally, Section 6 concludes this article.

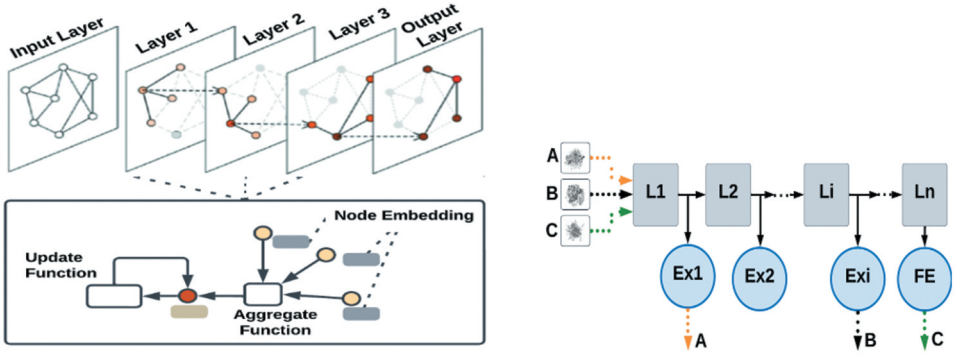
2. Graph neural networks

Geometric deep learning, an extension of deep neural networks (DNNs) to non-Euclidean domains, has emerged as a significant research area, with GNNs being a part of this field (Bronstein et al. 2017). GNNs are designed to transform nodes, edges, and the global context of a graph in a permutation-invariant manner, allowing them to extract crucial information for classification or prediction tasks. At the core of GNNs is the concept of node embedding, where each node is represented by a low-dimensional vector. In HS image analysis, nodes typically represent individual pixels, while edges encode spatial relationships between them. Each node is associated with a node embedding or feature vector representation, $F_i \in \mathbb{R}^d$, where d represents the dimensionality of the feature vector. This representation effectively captures the spectral characteristics of each pixel. GNNs learn node embeddings through message passing between node neighbours, a fundamental mechanism that enables nodes to update their embeddings based on information from neighbouring nodes. In HS image analysis, this process involves each pixel aggregating information from its neighbours to update its spectral embedding. Through this iterative process, each pixel accumulates information from its neighbours and gains a comprehensive understanding of the entire image structure, leading to insightful conclusions (Gilmer et al. 2017).

This graph learning process can be conceptualized as a combination of aggregation and update operations, as described in Equation (1) (G. Li et al. 2019). Aggregation functions collect feature representations from neighbouring nodes, while update functions apply non-linear transformations to compute new node embedding.

$$\begin{aligned} \mathbf{H}^{l} &= g(\mathbf{H}^{l-1}, \mathbf{W}^{l}) \\ &= Update(Aggregate(\mathbf{H}^{l-1}, \mathbf{W}_a^{l}), \mathbf{W}_u^{l}). \end{aligned} \tag{1}$$

where $\mathbf{H}^{(l-1)}$ represents the output from the $(l-1)^{th}$ layer or the input to the l^{th} layer, $\mathbf{W}_a^l$ and $\mathbf{W}_u^l$ are the learnable weights of aggregation function and update functions, respectively. There are several variations on these two functions. For instance, mean aggregation (Kipf and Welling 2017) and max-pooling (Hamilton, Ying, and Leskovec 2017) can be used in the aggregation function. Similarly, gated network (Y. Li et al. 2016) and multi-layer perceptron (Hamilton, Ying, and Leskovec 2017) can be utilized in the update function. In the context of HS image classification, aggregation functions can gather spectral information from neighbouring pixels, such as by computing the mean or max-pooling the spectral values. For instance, mean aggregation calculates the average spectral signature of neighbouring pixels. Update functions then update the node feature representations based on the aggregated information. A gated network, for example, could selectively update node features based on certain conditions, while a multi-layer perceptron could apply a series of non-linear transformations to capture complex relationships between node features. Overall, the combination of aggregation and update functions in graph convolution operations enables GNNs to effectively capture spatial-spectral features in HS image data, leading to accurate classification.



(a) Graph neural network (b) Multi-exit mechanism

Figure 1. General working of a graph neural network and visual representation of the multi-exit mechanism.

Similar to other neural networks, GNNs can be constructed using fully connected layers, graph convolution layers, and pooling layers. The choice of neural layers depends on the complexity and intended outcome of the input samples. Figure 1(a) illustrates the typical architecture of a GNN model. The graph-structured input data (e.g. HS imagery) is first fed into the input layer of the GNN model. This layer converts information about the nodes and their relationships within the graph into a vector representation known as node embedding. These embeddings then undergo processing through multiple layers of the GNN, where each layer conducts message passing between nodes to refine the embeddings. The output layer of the GNN generates the final embeddings, which can be utilized for various graph-related applications such as node classification (Kipf and Welling 2017), link prediction (Hwang et al. 2022), and graph classification (Zhang et al. 2018). To address the diverse requirements of these applications, researchers have developed numerous GNN variants with different information-passing mechanisms, including GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), and Graph Attention Network (GAT) (Veličković et al. 2018). The subsequent section explore the specifics of GCN, a fundamental element of the proposed model.

2.1. Graph convolution neural network

GCN, or graph convolution neural network, is an extension of CNN that can perform convolution operations on graph-structured data efficiently. GCN has demonstrated impressive performance in many areas, including social network mining (Tang and Liu 2009), recommendation systems (Ying et al. 2018), and natural language processing (Zhuang et al. 2017). Because of its effectiveness in handling non-Euclidean data, this paper employs a multi-layer graph convolution network to extract the contextual relationships between pixels in an HS image. The HS pixels and the spectral relationships among them can be represented using a graph, denoted as $G = (V, E)$, where V signifies nodes (in this case, the HS image pixels) and E signifies edges or the similarity between any two nodes. if $e_{u,v} \in E$ then the nodes u and v are connected by the edge $e_{u,v}$. The adjacency

matrix, denoted as $\mathcal{A}$, specifies the edges or relationships between the nodes and is generated by using Equation (2) (Hong et al. 2020)

$$\mathcal{A}_{u,v} = exp\left(-\frac{||\mathbf{X}_u - \mathbf{X}_v||^2}{\sigma^2}\right), \tag{2}$$

where $\mathbf{X}_u$ and $\mathbf{X}_v$ indicate the spectral signature associated with the nodes u and v and the parameter σ determines the width of the function. Here, the spectral relationship between nodes u and v is determined by comparing their spectral signatures $\mathbf{X}_u$ and $\mathbf{X}_v$. The Euclidean distance $||\mathbf{X}_u - \mathbf{X}_v||^2$ between these signatures is used to quantify the similarity or dissimilarity between the spectral characteristics of the two pixels. This distance is then used in the exponential function to calculate the weight of the edge connecting nodes u and v in the graph. The parameter σ controls the width of the exponential function, which in turn affects the influence of spectral similarity on the edge weights. From $\mathcal{A}$, Laplacian matrix, $\boldsymbol{L}$ can be generated as $\boldsymbol{L} = \mathcal{D} - \mathcal{A}$, where $\mathcal{D}$ is the diagonal matrix corresponding to the degree of adjacency matrix, i.e. $\mathcal{D}_{i,i} = \sum \mathcal{A}_{i,j}$ (Hong et al. 2019). The normalized symmetric Laplacian matrix can be generated (Hong et al. 2020) using Equation (3)

$$\begin{aligned} \mathcal{L} &= \left(\mathcal{D}^{-\frac{1}{2}}\right)\boldsymbol{L}\left(\mathcal{D}^{-\frac{1}{2}}\right) \\ &= I - \left(\mathcal{D}^{-\frac{1}{2}}\right)\mathcal{A}\left(\mathcal{D}^{-\frac{1}{2}}\right). \end{aligned} \tag{3}$$

where $\mathbf{I}$ is the identity matrix. The dimension of the input feature matrix is $n \times b$, and the dimension of the adjacency matrix is $n \times n$, where n and b denote the total number of nodes and bands, respectively.

The symmetric Laplacian matrix, $\mathcal{L}$ obtained in Equation (3) is a symmetric positive semidefinite matrix with eigen decomposition $\mathbf{U}\Lambda\mathbf{U}^T$, where $\mathbf{U}$ indicates the eigenvectors of $\mathcal{D}$ and $\boldsymbol{\Lambda}$ indicates the diagonal matrix of eigenvalues of $\mathcal{L}$ (M. Chen et al. 2020). The use of eigenvectors and eigenvalues of the Laplacian matrix enables the analysis and processing of graph signals in the spectral domain, providing insights into the graph's structure and facilitating operations such as filtering and transformation. In the graph Fourier domain, the graph G's spectral filtering operation can be expressed as a multiplication of a signal, x, and a filter $g_\theta = \mathcal{D}(\theta)$ (Wan et al. 2020), i.e.

$$g_\theta * x = \mathbf{U}g_\theta\mathbf{U}^T x. \tag{4}$$

The graph convolution propagation rule is based on Equation (4). By varying the filter function, g_θ, several propagation rules are proposed to simplify graph convolution. g_θ can be thought of as a function of eigenvalues of Laplacian matrix, L, i.e. $(g_\theta(\boldsymbol{\Lambda}))$. Hammond, Vandergheynst, and Gribonval (2011) attempted to reduce parameter complexity by approximating $g_\theta(\boldsymbol{\Lambda})$ with a Chebyshev polynomials' k^{th} order truncated expansion, $T_k(x)$ as shown in Equation (5).

$$g_{\theta'}(\boldsymbol{\Lambda}) \approx \sum_{k=0}^{k} \theta'_{\boldsymbol{k}} T_k(\tilde{\boldsymbol{\Lambda}}), \tag{5}$$

where $\theta^{'}$ is Chebyshev coefficient vector and $(\tilde{\mathbf{\Lambda}})$ is $\frac{2}{\lambda_{max}}\mathbf{\Lambda} - \mathbf{I}$ where λ_{max} is the largest eigenvalue and Equation (4) can be rewritten as

$$\begin{aligned} g_\theta * x &\approx \mathbf{U}\left(\sum_{k=0}^{k} \theta_k^{'} T_k(\tilde{\Lambda})\right)\mathbf{U}^T x \\ &\approx \left(\sum_{k=0}^{k} \theta_k^{'} \mathbf{U} T_k(\tilde{\mathbf{\Lambda}})\mathbf{U}^T\right)x \\ &\approx \left(\sum_{k=0}^{k} \theta_k^{'} T_k(\tilde{\mathcal{L}})\right)x, \end{aligned} \tag{6}$$

where $\theta_{\boldsymbol{k}}^{'}$ is Chebyshev coefficients, T_k is Chebyshev polynomial, and $(\tilde{\mathcal{L}})$ is Laplacian matrix after scaling, i.e. $\frac{2}{\lambda_{max}}\mathcal{L} - \mathbf{I}$. To simplify the graph convolution even further, Kipf and Welling (2017) used a first order (k = 1) Chebyshev polynomial and set the largest eigenvalue of the Laplacian matrix to two, i.e. $\lambda_{max} = 2$, resulting in Equation (7).

$$\begin{aligned} g_\theta * x &\approx \theta_0^{'} x + \theta_1^{'}(\mathcal{L} - \mathbf{I})x \\ &\approx \theta_0^{'} x + \theta_1^{'}(\mathcal{D}^{-\frac{1}{2}})\,\mathcal{A}(\mathcal{D}^{-\frac{1}{2}})x, \end{aligned} \tag{7}$$

by setting $\theta_0^{'}$ and $-\theta_1^{'}$ to θ

$$g_\theta * x \approx \theta\left(I + \left(\mathcal{D}^{-\frac{1}{2}}\right)\mathcal{A}\left(\mathcal{D}^{-\frac{1}{2}}\right)\right)x. \tag{8}$$

Multiple applications of this operation may result in numerical instability and vanishing or exploding gradient problems since the Eigenvalues of $(\boldsymbol{I} + (\mathcal{D}^{-\frac{1}{2}})\,\mathcal{A}(\mathcal{D}^{-\frac{1}{2}}))$ falls between 0 and 2 (Jia et al. 2024). Kipf and Welling (2017) solved this problem through re-normalization technique.

So that $(\boldsymbol{I} + (\mathcal{D}^{-\frac{1}{2}})\,\mathcal{A}(\mathcal{D}^{-\frac{1}{2}}))$ become $((\tilde{\mathcal{D}}^{-\frac{1}{2}})\tilde{\mathcal{A}}(\tilde{\mathcal{D}}^{-\frac{1}{2}}))$ where $\tilde{\mathcal{A}} = \mathcal{A} + \mathbf{I}$ and $\tilde{\mathcal{D}}_{i,j} = \sum_j \tilde{\mathcal{A}}_{i,j}$

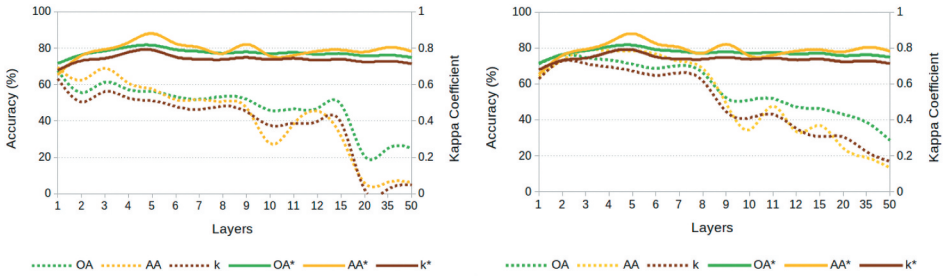
From Equation (8), the GCN propagation rule can be formulated as follows:

$$\begin{aligned} \mathbf{H}^{l} &= (\tilde{\mathcal{D}}^{-\frac{1}{2}}\tilde{\mathcal{A}}(\tilde{\mathcal{D}}^{-\frac{1}{2}}))\mathbf{H}^{l-1}\mathbf{W}^{l} + \mathbf{W}_0^{l} \\ &= \tilde{\boldsymbol{F}}\,\mathbf{H}^{l-1}\mathbf{W}^{l} + \mathbf{W}_0^{l}. \end{aligned} \tag{9}$$

where $\tilde{\boldsymbol{F}}$ is $\left(\tilde{\mathcal{D}}^{-\frac{1}{2}}\tilde{\mathcal{A}}\left(\tilde{\mathcal{D}}^{-\frac{1}{2}}\right)\right)$, $\mathbf{H}^{l}$ and $\mathbf{H}^{l-1}$ are the outputs of l^{th} layer and $(\mathrm{l}-1)^{th}$ layer respectively, and $\mathbf{W}^{l}$ is the l^{th} layer's trainable weight matrix. Each hidden layer, $\mathbf{H}^{l}$ is associated with a $n \times b^{i}$ feature matrix, with each row representing a node's feature. The subsequent section discusses the oversmoothing problem encountered in GNN models when stacking multiple GCN layers.

2.2. Oversmoothing problem

As discussed in the Introduction Section, most GNN models exhibit optimal performance with shallow depth, typically limited to 2 or 3 layers (Kipf and Welling 2017; Ud Din and Qureshi 2024; Veličković et al. 2018; Y. Wang et al. 2019). This limitation arises because adding more layers can result in performance degradation, often due to oversmoothing, where node feature representations become increasingly similar with additional layers. Recently, numerous researchers have been focusing on addressing these challenges in deep GNNs (M. Chen et al. 2020; G. Li et al. 2019; Xu et al. 2018; Zhao and Akoglu 2020). This



(a) Indian Pines dataset (b) Pavia University dataset

Figure 2. Performance of the GNN model across graph convolution layers, both when incorporating (solid lines) and not incorporating (dotted lines) oversmoothing-resistant techniques. OA, AA, and k represent accuracy metrics (overall accuracy, average accuracy, and kappa coefficient, respectively) before addressing oversmoothing, while OA*, AA*, and k* represent the metrics after addressing oversmoothing.

section examines the performance of GNN models in HS image classification as layers are added and discusses two approaches to mitigate the oversmoothing problem Figure 2.

For the experiment, a GCN model (Kipf and Welling 2017) was augmented with multiple graph convolution layers. The model was trained for HS image classification using the Indian Pines (2) and Pavia University (2(b)) datasets, with up to 50 layers added to explore variations in classification accuracy. Performance was assessed both with incorporating and without incorporating oversmoothing-resistant techniques. The figures reveal that, without oversmoothing-resistant techniques, shallow-layered models outperform deep-layered GNN models (dotted curves). The 50-layer GNN model failed to classify any HS images, indicating significant performance reduction with increased layers. Conversely, the model with oversmoothing-resistant techniques (solid lines) maintained consistent performance as layers increased, underscoring the critical impact of oversmoothing on deep-layered GNN models. Section 5.2 further analyzes the GNN's performance by increasing the number of layers and employing various oversmoothing-handling techniques. Given the GNN's limitation to deep architectures, which restricts its expressive power in classification, several studies have attempted to address the oversmoothing issue. This section briefly discusses two prominent techniques: Initial Residual and Identity Mapping (M. Chen et al. 2020) and PairNorm (Zhao and Akoglu 2020).

2.2.1. Initial residual and identity mapping technique (GCNII)

According to Wu et al. (2019) stacking numerous layers (m) on a graph convolution neural network replicates an m^{th} order polynomial filter with fixed coefficients over the graph spectral domain and these fixed coefficients limit the expressiveness of a multi-layer GCN model, resulting in oversmoothing. To extend the expressiveness of GCN models to deep architectures, the model must express the m^{th} order polynomial filter with any arbitrary number of coefficients. Chen et al. (2020) employed initial residual connection and identity mapping techniques (GCNII) to achieve this. The $(l+1)^{th}$ layer in the GCNII model can be defined as Equation (10)

$$\boldsymbol{H}^{l+1} = \sigma\big(((1-\alpha_l)\tilde{\boldsymbol{P}}\boldsymbol{H}^l + \alpha_l\boldsymbol{H}^0)((1-\beta_l)\boldsymbol{I}_n + \beta_l\boldsymbol{W}^l)\big). \tag{10}$$

where α and β are two hyper parameters, and $\tilde{\boldsymbol{P}}$ is the re-normalized graph convolution matrix, which equals $\left(\tilde{\mathcal{D}}^{-\frac{1}{2}}\mathcal{A}\left(\tilde{\mathcal{D}}^{-\frac{1}{2}}\right)\right)$. Here $\mathcal{A}$ is adjacency matrix and $\mathcal{D}$ is diagonal degree matrix. In GCNII, authors made two modification into the standard GCN model. The $(l+1)^{th}$ layer of standard GCN model can be expressed as $\mathbf{H}^{l+1} = \sigma\big(\big(\tilde{P}\mathbf{H}^l\big)W^l\big)\big)$ (M. Chen et al. 2020). The first modification is the addition of initial residual connection, which combines smoothed representation, $\tilde{\boldsymbol{P}}\,\mathbf{H}^l$, with the first layer, $\boldsymbol{H}^0$. The second modification is the introduction of identity mapping, that adds $\mathbf{I}_n$ to the weight matrix, $\boldsymbol{W}^l$ as illustrated in Equation (10). This strategy slows the performance drop and alleviates the problem of oversmoothing to some extent as network depth increases.

2.2.2. PairNorm

Li, Han, and Wu (2018) asserted that graph convolution is a form of Laplacian smoothing and the repeated application of convolution operations or Laplacian smoothing causes node representations to resemble one another. PairNorm is a normalization technique introduced by Zhao and Akoglu (2020) that prevents node representations from becoming identical to one another. It maintains the sum of pairwise distances across layers, prevents feature mixing, and guarantees that distant pairs have unique features. PairNorm requires no changes to network architecture and can be applied to any GNN model. It significantly improves the GNNs' resistance to oversmoothing and allows for deeper models to be trained without sacrificing efficiency. PairNorm normalization consists of two steps: centring the data points and scaling them. It can be applied after each graph convolution layer that centres and then scales the data points. The normalization procedure used Equation (11) for centring the data points and Equation (12) for scaling.

$$\mathbf{X}_i^c = \mathbf{X'}_i - \frac{1}{n}\sum_{i=1}^{n}\mathbf{X'}_i, \tag{11}$$

$$\tilde{\mathbf{X}}_i = s.\frac{\mathbf{X}_i^c}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}||\mathbf{X}_i^c||_2^2}}. \tag{12}$$

where $\mathbf{X'}_i$ denotes the input of PairNorm or the output of graph convolution layer, n represents the total number of elements in the feature matrix, $\tilde{\mathbf{X}}_i$ is the output of PairNorm, $\mathbf{X}_i^c$ is $\mathbf{X'}_i$ after centring, and s is the hyper parameter that decides the pairwise squared distance. HyperGCN utilizes the PairNorm approach to mitigate the oversmoothing issue and enhance the model's performance within a multi-layered architecture. Additionally, the proposed model integrates initial residual and identity mapping techniques to address the oversmoothing problem. A comparison is conducted to assess the performance of these two oversmoothing handling strategies.

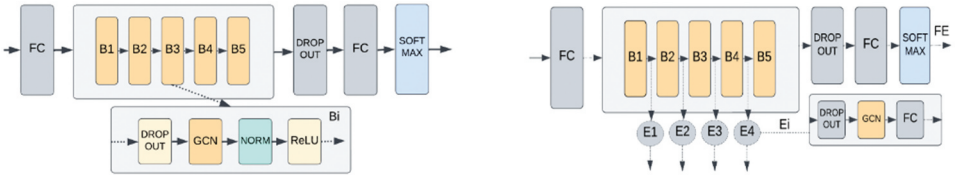
2.3. Multi-exit neural network

A multi-exit neural network incorporates multiple exit points within the intermediate layers of a standard neural network, enabling predictions at various stages. Unlike traditional DNNs, which process input data through all layers before producing a final output, multi-exit DNNs allow for predictions or decisions at these intermediate stages based on their prediction confidence. If the prediction confidence meets a predefined threshold at any exit point, the input data can stop propagating further and return the prediction label. As illustrated in Figure 1(b), input samples progress through the network, and upon reaching early-exit branches (Exi), classification labels are generated from the features extracted up to that point, and their confidence is assessed. Typically, entropy or max-probability of the generated output is used as a confidence measure (Kaya, Hong, and Dumitras 2019; Panda, Sengupta, and Roy 2017; Teerapittayanon, McDanel, and Kung 2016). For example, samples with lower complexity (e.g. A and B) exit early via E × 1 and Exi, while more complex samples (e.g. C) traverse the entire network before exiting at the final output layer (FE). Multi-exit networks reduce the computational load for confidently classified inputs, leading to faster responses and lower energy consumption, which is particularly beneficial in real-time or resource-constrained environments.

3. Methodology

This section demonstrates the methodology used to develop HyperGCN. First, HS image pixels undergo preprocessing to generate a graphical representation, including an input feature matrix, a ground truth matrix, and an adjacency matrix. These matrices are then inputted into the proposed model for training and validation. GCN employs a transductive learning approach, where the entire graph and training set labels are used as input for model training. The model then predicts the testing set labels. The labels, represented as Y, are divided into a train mask and a test mask. The train mask is used for model training, while the test mask is used for evaluation. Utilizing Adam optimizer and cross-entropy loss function, HyperGCN is trained over one thousand epochs.

The architecture of HyperGCN is depicted in Figure 3(a). The model includes a fully connected layer (FC), five graph convolution blocks (B_i), a dropout layer (DROPOUT) followed by a fully connected layer (FC) with softmax. Through empirical analysis, five graph convolution blocks were selected. The analysis results are discussed in Section 5.2. Please refer to the bolded scores in Tables 9 and 10,



(a) HyperGCN (b) Multi-exit HyperGCN

Figure 3. The architecture of HyperGCN and multi-exit HyperGCN.

which provide the layer-wise accuracy variations of Multi-GCN, GCNII, and HyperGCN on the Indian Pines and Pavia University datasets, respectively. Each convolution block (B_i) comprises four layers: a dropout layer (DROPOUT), a graph convolution layer (GCN), a normalization (NORM) and an activation (ReLU) layer. The PairNorm normalization technique described in Section 2.2.2 is employed in the NORM layer. Each HS image pixel can be viewed as a two-dimensional image with a height of one (M. Chen et al. 2020). Therefore, the size of the input layer is $(1, b)$. Before propagating to convolution block, a fully connected layer is applied on the input feature, $\mathbf{X}$, to obtain a lower-dimensional representation, $\mathbf{H}^1$, as shown in Equation (13):

$$\mathbf{H}^1 = \mathbf{W}^T\mathbf{X} + W_0, \tag{13}$$

where $\mathbf{W}$ denotes the weight matrix, $\mathbf{X}$ represents the input feature matrix, and W_0 denotes the bias term. The dimension of $\mathbf{H}^1$ is $(1, n2)$, where $n2$ is the reduced dimension. The hidden layers of the model, $\mathbf{H}^l$, can be represented as $g(\mathbf{H}^{l-1}, \mathcal{A})$, where $\mathbf{H}^{l-1}$ is the output of the $(l-1)$-th layer and g is the convolution rule or layer-wise propagation function, as described in Equation 9. Each layer aggregates feature representations using the propagation rule, g, to form the feature representations of the subsequent layer. These features gradually become more abstract as they propagate through subsequent layers.

Stacking multiple graph convolution layers simulates a polynomial filter with fixed coefficients, limiting the model's expressive power and resulting in oversmoothing (Wu et al. 2019). A recent study (Rong et al. 2020) also found that randomly removing some edges from the graph slows the rate of oversmoothing. Furthermore, performing multiple convolution and non-linearity operations on the feature matrix can lead to overfitting (Gasteiger, Bojchevski, and Günnemann 2019). To address these issues and mitigate the resulting performance drop, a dropout layer and a PairNorm normalization layer are incorporated immediately before and after each graph convolution layer, respectively. Equation (14) describes the output of the convolution block:

$$\mathbf{H}^l = \lambda\left(Pair\,Norm\left(\tilde{\boldsymbol{F}}_{drop}\mathbf{H}^{l-1}\mathbf{W}^l + \mathbf{W}_0^l\right)\right). \tag{14}$$

Here, $\tilde{F}_{drop}$ is equivalent to $\tilde{\boldsymbol{F}}$, except that some edges of the adjacency matrix are randomly removed (DROPOUT), PairNorm is the normalization function as described in Section 2.2.2, and $\lambda(\cdot)$ represents the activation function (ReLU). The details of constructing multi-exit HyperGCN are discussed in Section 4.3.

The performance of the proposed model is measured using four commonly used performance metrics: Cohen's kappa coefficient (k), average accuracy (AA), overall accuracy (OA), and individual class-wise accuracy. Cohen's kappa coefficient (Kraemer 2015) is an important performance metric in multi-label classification tasks where a simple accuracy score is insufficient to understand the model better. Also, it is a very useful statistic when the dataset is unevenly distributed across classes. It demonstrates how much better the model performs on random guesses as a function of the frequency of each class. The kappa coefficient can be mathematically defined as $k = \frac{P_o - P_e}{1 - P_e}$, where k is the kappa coefficient, P_o is the observed accuracy of the model, and P_e is the expected accuracy at random chance. Its value ranges from 0 to 1, with 0 indicating that the model is useless and 1 indicating that the model is perfect.

4. Experiments

This section discusses extensive experiments conducted on HyperGCN and its multi-exit version (M-HyperGCN) using four benchmark HS image datasets: Indian Pines, Pavia University, Salinas, and Botswana. To evaluate their effectiveness, a comparative analysis is performed with models including (Kipf and Welling 2017), Multi-GCN, GCNII, MiniGCN (Hong et al. 2020), FuNet-C (Hong et al. 2020), CEGCN (Liu et al. 2021), ConGCN (W. Yu et al. 2023). Section 4.2 details these baseline models. The evaluation employs standard performance measures such as overall and average accuracy, as well as Cohen's kappa coefficient. Individual category-wise accuracy is also compared. Additionally, performance degradation in HS image classification with increased layers is examined. HyperGCN is evaluated both with and without incorporating oversmoothing-resistant techniques. Additionally, Section 4.3 evaluates the performance of M-HyperGCN versus HyperGCN to assess the effectiveness of augmented exit branches in predicting outcomes, reducing computational cost and time, and maintaining accuracy.

4.1. Data sets

Four benchmark HS datasets – Indian Pines, Pavia University, Salinas, and Botswana – are used to evaluate the proposed model. The Indian Pines dataset (Marion, Baumgardner, and Landgrebe David 2015) is captured over northwestern Indiana, U.S.A., with 145×145 pixels and 220 spectral bands per pixel spanning 400 to 2450 nm at a spectral resolution of 10 nm. It includes 16 land-cover categories (Shahraki and Prasad 2018) as illustrated in 1. The Pavia University dataset (Hong et al. 2020) was captured over the city of Pavia using a ROSIS sensor. It comprises 42,776 images with dimensions of 610×340 pixels and 103 spectral bands ranging from 430 to 860 nm. The pixels have a spatial resolution of 1.3 m and are classified into nine classes, as illustrated in Table 4. The Salinas dataset (Q. Yu et al. 2023) covers the Salinas Valley in California, U.S.A., with 54,129 pixels, 224 bands (after removing noisy bands, 204 bands), and a spatial resolution of 3.7 m. It contains annotations for 16 different crops, as illustrated in Table 2. The Botswana dataset (Q. Yu et al. 2023), captured over the Okavango Delta, comprises 3248 pixels with dimensions of 1476×256 and a pixel resolution of 30 nm. It includes 242 bands spanning the spectral range from 400 to 2500 nm, with 145 bands retained after denoising. The dataset covers 14 land-cover categories, such as seasonal swamps and dry woodlands, as detailed in Table 3. The distribution of testing and training data points across different classes in the Indian Pines, Salinas, Botswana, and Pavia University datasets is illustrated in Tables 1–4 respectively. Smaller training datasets are used compared to the test set for two reasons. Firstly, this allows for a fair comparison with baseline models, as previous studies (Hong et al. 2020; Kipf and Welling 2017) evaluated their models (GCN, MiniGCN, Funet-C) with minimal training sets. Secondly, it enables an assessment of the proposed HyperGCN model's suitability for HS image classification under limited training samples. The performance of HyperGCN is further analysed by varying the size of the training set, as described in Section 5.3.

Table 1. Land-cover categories of the Indian Pines dataset with train-test data distribution.

Id	Class Name	#Train	#Test
1	Corn-notill	50	1384
2	Corn-mintill	50	784
3	Corn	50	184
4	Grass-pasture	50	447
5	Grass-trees	50	697
6	Hay-windrowed	50	439
7	Soybean-notill	50	918
8	Soybean-mintill	50	2418
9	Soybean-clean	50	564
10	Wheat	50	162
11	Woods	50	1244
12	Buildings-grass-trees-drives	50	330
13	Stone-steel-towers	50	45
14	Alfalfa	15	39
15	Grass-pasture-mowed	15	11
16	Oats	15	5

Table 2. Land-cover categories of the Salinas dataset with train-test data distribution.

Id	Class Name	#Train	#Test
1	Broccoli green weeds 1	30	1979
2	Broccoli green weeds 2	30	3696
3	Fallow	30	1946
4	Fallow rough plow	30	1364
5	Fallow smooth	30	2648
6	Stubble	30	3929
7	Celery	30	3549
8	Grapes untrained	30	11241
9	Soil vineyard develop	30	6173
10	Corn senesced green weeds	30	3248
11	Lettuce romaines, 4 wk	30	1038
12	Lettuce romaines, 5 wk	30	1897
13	Lettuce romaines, 6 wk	30	886
14	Lettuce romaines, 7 wk	30	1040
15	Vineyard untrained	30	7238
16	Vineyard vertical trellis	30	1777

4.2. Baseline models

HyperGCN and M-HyperGCN are evaluated against seven baseline models: GCN (Kipf and Welling 2017), MiniGCN (Hong et al. 2020), Multi-GCN, GCNII, FuNet-C (Hong et al. 2020), CEGCN (Liu et al. 2021), and ConGCN (W. Yu et al. 2023). The characteristics of these baseline models and motivations for their selection are explained below:

(1) **GCN**, proposed by Kipf and Welling (2017), consists of two graph convolution layers with 64 units. Batch normalization with a 0.9 momentum is applied before and after each graph convolution layer. ReLU activation is employed before feeding the samples into the softmax layer. This model is chosen as the baseline for its status as the standard graph convolution model proposed for classification on graph-structured data, and it forms the foundation for the proposed HyperGCN.
(2) **MiniGCN**, described by (Hong et al. 2020), consists of three graph convolution layers, each with 64 units. Batch normalization with a 0.9 momentum is applied

Table 3. Land-cover categories of the Botswana dataset with train-test data distribution.

Id	Class Name	#Train	#Test
1	Water	30	240
2	Hippo grass	30	71
3	Floodplain grasses 1	30	221
4	Floodplain grasses 2	30	185
5	Reeds	30	239
6	Riparian	30	239
7	Firescar	30	229
8	Island interior	30	173
9	Acacia woodlands	30	284
10	Acacia shrublands	30	218
11	Acacia grasslands	30	275
12	Short mopane	30	151
13	Mixed mopane	30	238
14	Exposed soils	30	65

Table 4. Land-cover categories of the Pavia dataset with train-test data distribution.

Id	Class Name	#Train	#Test
1	Asphalt	548	6304
2	Meadows	540	18146
3	Gravel	392	1815
4	Trees	524	2912
5	Metal Sheets	265	1113
6	Bare Soil	532	4572
7	Bitumen	375	981
8	Bricks	514	3364
9	Shadows	3921	40002

before and after each graph convolution layer, followed by a ReLU activation layer. MiniGCN shares the same architecture as GCN but distinguishes itself by its ability to be trained in batches, leading to a better local optimum of networks. MiniGCN is an exclusive GCN-based model proposed for classifying high-resolution images and is considered a baseline model.

(3) **Multi-GCN**, a multi-layered GCN model that does not address the oversmoothing problem. It shares a similar structure with the proposed HyperGCN model but lacks the PairNorm normalization layer. Multi-GCN serves as a baseline model for evaluating how a GCN-based model behaves when more layers are added without addressing oversmoothing issues.

(4) **GCNII**, a multi-layered GCN model that incorporates initial residual and identity mapping techniques (M. Chen et al. 2020) to mitigate oversmoothing when additional graph convolution layers are added. The structure of GCNII mirrors that of the proposed model, except it employs initial residual and identity mapping techniques instead of PairNorm normalization to address potential performance drops. Using GCNII as a baseline, the effectiveness of two oversmoothing handling techniques within the HyperGCN architecture can be compared: the initial residual and identity mapping technique and the PairNorm normalization technique.

(5) **FuNet-C**, a fusion model that concatenates MiniGCN and CNN, as demonstrated by (Hong et al. 2020). It preprocesses the HS dataset by generating compatible

structures for both MiniGCN and CNN separately. This ensures that the data is appropriately formatted for each network type, allowing for effective feature extraction and fusion. The FuNet model has a feature extraction module that extracts different kinds of features using CNN and MiniGCN and a fusion module that combines these features using different fusion strategies: additive (FuNet-A), element-wise multiplicative (FuNet-M), and concatenation (FuNet-C). Among these, FuNet-C achieves better classification performance. It is chosen as a baseline model to assess whether our proposed GCN-based model, HyperGCN, can achieve a similar performance gain as GCN-CNN combined models.

(6) **CEGCN**, an integrated model that combines both CNN and GCN branches, allowing for feature learning at both pixel and superpixel levels (Liu et al. 2021). It addresses the structural incompatibility between Euclidean data-oriented CNNs and non-Euclidean data-oriented GCNs by integrating a graph encoder and decoder. These components facilitate the propagation of features between image pixels and graph nodes, leading to enhanced performance in HS image classification. We selected CEGCN, a state-of-the-art CNN-GCN model for classifying HS images, as our baseline to investigate whether our HyperGCN models could achieve comparable performance improvements.

(7) **ConGCN**, a sophisticated GNN-based model for HS image classification (W. Yu et al. 2023). It incorporates contrastive learning, the SLIC algorithm (Achanta et al. 2012), adaptive graph augmentation techniques, and semi-supervised contrastive and graph generative loss functions into its GCN architecture. The model first segments the HS image using SLIC and constructs a graph where each segment is a node. Augmented graphs are then created using graph augmentation techniques, and localized, hierarchical graph convolutions generate node representations. Finally, semi-supervised contrastive and graph generative loss functions are used for training. ConGCN has shown outstanding performance in HS image classification, making it a suitable baseline to compare against our innovative M-HyperGCN for performance enhancements.

4.3. Multi-exit HyperGCN

As depicted in Figure 3(b), M-HyperGCN is constructed by adding side exit branches after every graph convolution block of HyperGCN model. Each side exit branch is composed of a dropout layer, a graph convolution layer, and a fully connected layer. Training the Modified HyperGCN involves a joint training strategy, treating both the HyperGCN and the additional branches as a single optimization problem and training them together. During training, the model calculates a loss at each exit branch and aims to minimize the total loss. This total loss is the sum of the losses generated at each exit point and the final output layer, as defined in Equation (15).

$$\mathcal{L} = \sum_{i=1}^{N+1} W_i L_i. \tag{15}$$

where $\mathcal{L}$ represents the overall loss, N signifies the total exit branches added to the model, W_i denotes the joint optimization weight that reflects the significance assigned to the output of a particular branch in calculating the overall loss. In this research, evenly distributed weights, W_i, ranging from 0.1 to 1 were utilized. L_i denotes the cross-entropy loss generated at the i^{th} exit branch, as shown in Equation (16):

$$L_i = -\frac{1}{|C|}\sum_{c\in C} y_c \log \hat{y}_c, \tag{16}$$

where y is the ground truth vector, C is the set of all possible labels, $\hat{y} = \text{softmax}(z) = \left(\frac{\exp(z)}{\sum_{c\in C}\exp(z_c)}\right)$, and z is the output of the i th exit branch. After training, the M-HyperGCN can be utilized to classify HS images. When HS images reach the initial early-exit branch, Ex1, a probability distribution across all potential labels is generated using the features extracted up to that layer. To assess confidence, the label with the highest probability is considered and compared to a predetermined threshold value. This determines whether to exit the classification process or proceed to the subsequent exit branch. If the confidence meets the criteria, the propagation and processing of the HS image cease, and the label with the highest probability is returned. Otherwise, the HS image continues to the next exit branch, repeating the process until it reaches the final output layer. Given that predetermined threshold value significantly influences the performance of multi-exit DNNs (Haseena Rahmath, Srivastava, and Chaurasia 2023), a brute force analysis was conducted on thresholds ranging from 0.6 to 0.9, with an incremental step of 0.5. The optimal threshold value was then selected based on this evaluation.

4.4. Experimental setup

The programs necessary for the experiment are scripted in Python, and the PyTorch framework is used to implement the proposed models, HyperGCN and M-HyperGCN. PyTorch simplifies the definition, optimization, and evaluation of DNNs on GPUs. Model parameters are optimized using the Adam optimizer, with a learning rate set to 0.005, and the model is trained for 1000 epochs. The proposed model consists of five graph convolution blocks with a 64-unit graph convolution layer. To mitigate overfitting, a dropout layer with a rate of 0.06 is applied within the convolution block, and a dropout layer with a rate of 0.4 is applied before the final fully connected layer. Experiments are conducted on a computer equipped with an Nvidia GeForce GTX graphics card and an Intel Core i5 processor operating at 2.8 GHz. Both the proposed and baseline models, except ConGCN, share identical hyperparameters, including learning rate, epochs, and hidden units, and are executed on the same computer. Due to the unavailability of implementation code for ConGCN, the original article's scores have been utilized for comparison (W. Yu et al. 2023).

5. Results and discussion

This section presents findings from comprehensive experiments conducted on the experimental datasets. Section 5.1 compares the performance of HyperGCN models with

baseline models. Section 5.2 evaluates the proposed model's performance with increasing layers, both with and without the incorporation of oversmoothing-resistant techniques. It explores HyperGCN's performance under two different oversmoothing handling techniques. Section 5.3 examines the model's behaviour with different training dataset sizes, and Section 5.4 visually illustrates the experimental results. Finally, Section 5.5 evaluates HyperGCN alongside its multi-exit version, M-HyperGCN.

5.1. Performance comparison with baseline models

Tables 5–8 provide a comparison of the classification performance among various models in terms of OA, AA, and k. These tables also detail the accuracies for individual classes across the Indian Pines, Pavia University, Salinas, and Botswana datasets, respectively. GCN, Multi-GCN, GCNII, and HyperGCN exhibit similar classification trends across datasets. GCN's performance decreases with more layers (Multi-GCN), while techniques like initial residual and identity mapping (GCNII) improve accuracy. HyperGCN outperforms GCNII,

Table 5. Performance comparison of HyperGCN and M-HyperGCN with baseline models on the Indian Pines dataset. Best and second-best scores are bolded and underlined, respectively.

Id	GCN	Mini GCN	Multi-GCN	GCNII	FuNet-C	CEGCN	ConGCN	Hyper GCN	M-Hyper GCN
1	68.21	62.86	6.79	64.45	67.92	92.14	92.07	74.21	**97.84**
2	55.23	56.12	77.3	67.35	75.64	83.47	**97.50**	76.92	96.24
3	85.87	92.39	93.48	91.85	96.20	93.55	**100**	88.59	90.30
4	87.92	92.39	73.60	87.47	95.08	93.43	94.50	95.53	**98.60**
5	88.95	95.98	76.18	82.93	95.41	95.84	98.99	94.54	**100**
6	98.63	98.86	97.95	90.66	99.32	99.83	**100**	97.72	99.77
7	65.47	78.76	59.91	78.00	75.93	83.5	93.57	83.55	**98.69**
8	67.62	63.98	52.36	68.36	73.04	96.66	97.12	74.32	**97.61**
9	50.71	56.38	4.08	56.38	69.5	90.17	97.57	84.74	**97.64**
10	98.15	99.38	96.3	95.68	**100**	98.09	**100**	99.38	**100**
11	94.21	94.61	94.77	93.57	88.18	99.13	**99.83**	94.77	99.57
12	42.42	47.58	23.94	44.85	90.30	76.22	**99.41**	69.64	74.64
13	95.56	100	93.33	93.33	**100**	85.82	87.14	97.78	**100**
14	43.59	61.54	2.56	87.18	94.87	91.14	**98.75**	91.31	93.56
15	72.73	72.73	27.27	81.82	**100**	99.23	**100**	92.91	93.75
16	**100**	60.00	40.00	**100**	**100**	98.89	**100**	**100**	**100**
AA	75.95	73	57.49	80.24	88.84	92.32	**97.28**	88.49	94.92
OA	72.86	74.36	56.49	74.45	80.09	75.90	96.74	81.85	**97.05**
k	0.6902	0.7058	0.5115	0.7089	0.7734	0.8134	0.9627	0.7905	**0.9665**

Table 6. Performance comparison of HyperGCN and M-HyperGCN with baseline models on the Pavia University dataset. Best and second-best scores are bolded and underlined, respectively.

Id	GCN	Mini GCN	Multi-GCN	GCNII	FuNet-C	CEGCN	ConGCN	Hyper GCN	M-Hyper GCN
1	74.98	83.49	69.99	70.54	85.31	74.85	93.11	87.99	**98.78**
2	75.87	76.42	87.50	92.43	91.94	90.16	96.55	85.21	**99.81**
3	59.23	73.39	50.96	52.95	69.92	**99.12**	97.24	76.86	89.05
4	79.43	90.59	88.46	69.71	96.53	87.67	93.91	97.69	**97.71**
5	99.01	99.37	99.55	99.46	**100**	**100**	98.80	99.91	99.94
6	57.87	83.49	16.40	35.15	90.62	89.38	**100**	93.68	96.61
7	81.14	86.54	71.46	79.61	76.15	**99.92**	99.12	92.74	96.20
8	82.67	87.60	67.48	78.42	97.38	79.74	94.76	89.98	**97.49**
9	97.48	**100**	**100**	99.87	99.75	93.52	82.81	99.37	**100**
AA	78.63	86.77	72.42	75.35	89.66	90.48	95.14	91.49	**96.69**
OA	74.95	81.53	73.53	77.84	90.53	87.73	95.97	93.68	**97.89**
k	0.6712	0.7604	0.6382	0.6638	0.8737	0.8401	0.9469	0.9159	**0.9709**

Table 7. Performance comparison of HyperGCN and M-HyperGCN with baseline models on the Salinas dataset. Best and second-best scores are bolded and underlined, respectively.

Id	GCN	Mini GCN	Multi-GCN	GCNII	FuNet-C	CEGCN	ConGCN	Hyper GCN	M-Hyper GCN
1	94.70	98.29	85.45	99.11	98.70	93.40	**100**	98.84	**100**
2	33.45	98.92	70.40	94.35	97.11	**100**	100	99.95	**100**
3	85.43	95.22	90.13	83.14	99.08	**100**	**100**	93.73	**100**
4	97.73	99.34	92.38	99.41	98.93	99.18	98.5	99.49	**100**
5	97.58	96.60	82.74	90.63	97.89	95.65	97.58	98.26	**99.65**
6	99.82	99.80	99.75	99.64	99.72	99.61	99.81	99.84	**100**
7	99.41	99.72	99.32	99.58	99.80	99.86	99.94	99.85	**100**
8	55.44	37.76	68.88	83.00	79.18	90.01	98.33	70.81	**98.65**
9	97.68	99.87	93.18	85.78	98.77	99.92	100	99.82	**100**
10	82.64	90.70	82.14	81.90	86.58	88.16	**99.28**	84.61	98.89
11	81.91	93.83	66.96	83.72	97.59	95.91	**99.74**	92.39	98.78
12	95.31	99.95	88.67	99.84	**100**	**100**	98.26	99.63	99.96
13	94.89	99.21	94.70	99.44	99.44	89.53	97.58	99.77	**100**
14	80.06	88.90	86.44	85.67	88.77	99.12	98.86	88.97	**99.27**
15	61.87	73.72	52.65	41.35	54.64	54.47	**99.63**	73.78	73.80
16	90.88	96.09	84.13	91.62	97.86	99.93	**100**	92.91	**100**
AA	84.30	91.75	83.62	88.64	93.38	94.05	**99.22**	93.29	98.06
OA	78.79	83.30	79.27	83.35	88.79	90.27	**99.25**	90.33	98.79
k	0.7652	0.8157	0.7694	0.8140	0.8750	0.8913	**0.9917**	0.9024	0.9891

Table 8. Performance comparison of HyperGCN and M-HyperGCN with baseline models on the Botswana dataset. Best and second-best scores are bolded and underlined, respectively.

Id	GCN	MiniGCN	Multi-GCN	GCNII	FuNet-C	CEGCN	HyperGCN	M-Hyper GCN
1	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
2	92.16	**100**	**100**	**100**	**100**	**100**	**100**	**100**
3	96.52	98.51	84.58	97.01	**100**	**100**	99.50	**100**
4	87.88	91.52	87.27	97.58	99.23	100	97.88	**100**
5	84.47	88.13	66.21	87.21	96.18	70.34	86.76	**96.8**
6	70.32	71.23	52.51	84.02	93.15	**97.41**	84.65	95.43
7	98.09	98.56	97.61	97.61	**100**	98.23	98.99	**100**
8	96.73	96.73	71.90	95.42	**100**	82.66	**100**	**100**
9	85.98	84.47	89.39	85.23	96.57	100	90.91	99.24
10	15.03	88.89	94.44	73.74	**100**	97.22	94.93	99.49
11	98.82	95.69	60.00	98.43	74.12	**100**	94.12	99.61
12	97.71	**100**	4.58	96.18	**100**	**100**	93.13	**100**
13	79.36	69.27	70.64	92.66	96.79	97.05	93.12	**99.08**
14	88.89	**100**	93.33	**100**	**100**	89.23	**100**	**100**
AA	85.14	91.64	76.60	93.22	96.86	95.15	95.29	**99.26**
OA	83.12	90.11	76.02	92.11	95.96	95.25	93.53	**99.60**
k	0.8165	0.8926	0.7392	0.9142	0.9561	0.9495	0.9347	**0.9955**

suggesting PairNorm normalization is more effective in combating oversmoothing. This is further investigated in Section 5.2. While MiniGCN generally outperforms GCN across all datasets due to its ability to train networks in batches and reach better local optima, there are specific classes such as Oats in the Indian Pines, Mixed mopane in the Salinas dataset, and Grapes untrained in the Botswana dataset where MiniGCN performs lower than expected, deviating from the typical trend. This divergence suggests that MiniGCN may struggle to capture the intricate details and variations within these particular classes during batch-wise training. Training with batches can sometimes lead to less exposure to classes in each batch, especially noticeable with classes like Oats which have relatively low sample counts. The sensitivity to batch size and noise introduced by mini-batches

might potentially hinder MiniGCN's ability to effectively learn their distinguishing features.

Remarkably, the integration of additional graph convolution layers and over-smoothing-resistant techniques empowers HyperGCN to effectively capture both spectral and long-range spatial relations inherent in HS images. This advancement surpasses basic GCN models like GCN, MiniGCN, Multi-GCN, and GCNII, achieving performance levels comparable to combined CNN-GNN models like FuNet-C and CEGCN. Specifically, compared to Funet-C, HyperGCN shows significant enhancements in overall accuracy in the Indian Pines (1.76%), University of Pavia (3.15%), and Salinas (1.54%) datasets. However, FuNet-C has a notable edge in the Botswana dataset (2.4%) may be due to its fusion of CNN and MiniGCN. Nevertheless, M-HyperGCN, the multi-exit version, outperforms FuNet-C by 3.64%. HyperGCN outperforms CEGCN, especially in the Indian Pines and Pavia University datasets, showing an improvement in overall accuracy of 5.95%. In the Salinas and Botswana datasets, both models perform similarly, demonstrating comparable results.

When considering M-HyperGCN, it consistently outperforms all other models across various datasets, including shallow-layer models like GCN and MiniGCN, as well as multi-layer models like Multi-GCN, GCNII, and HyperGCN. Even when compared to advanced models like Funet-C, CEGCN, and ConGCN, M-HyperGCN demonstrates superior performance. In the Salinas dataset (Table 7), ConGCN achieved a 1.15% average accuracy improvement over M-HyperGCN, primarily due to challenges in classifying the Vineyard untrained land cover type. Excluding this class from the calculation, M-HyperGCN surpasses all baseline models in the Salinas dataset as well. This exceptional performance is attributed to its innovative design and feature extraction capabilities. By integrating multiple exit points throughout the network, M-HyperGCN effectively captures diverse information representation at different processing stages, mitigating overfitting and overthinking. This allows the model to learn robust representations and exit early from the processing pipeline once it gains sufficient prediction confidence.

Another advantage of M-HyperGCN is its capability to address the oversmoothing problem commonly encountered in GNN models. It achieves this by allowing HS images to exit early through intermediate exits based on their characteristics. As illustrated in Figure 15(a), the vast majority of HS images exit through intermediate exit points, with only a few challenging HS images propagated through all the layers of the model. Moreover, M-HyperGCN integrates oversmoothing-resistant techniques, notably PairNorm normalization, into its architecture. This integration further safeguards against the loss of critical spectral and spatial information during multiple graph convolution operations, enhancing the model's efficiency and effectiveness in handling HS image classification tasks. These improvements enable M-HyperGCN to outperform traditional GCN models with higher accuracy and robustness.

The individual class performance analysis reveals consistent accuracy in certain classes across all models. Classes like Wheat in the Indian Pines dataset, Metal Sheets in the Pavia University dataset, Stubble and Celery in the Salinas dataset, and Water and Firescar in the Botswana dataset exhibit distinct spectral characteristics and well-defined boundaries, facilitating accurate classification. However, for classes with complex spatial patterns and mixed natural and man-made elements, such as Buildings-grass-trees-drives and Grass-

pasture-mowed in the Indian Pines dataset, the CNN-GNN combination proves effective due to CNN's edge detection and pattern recognition capabilities. Challenges arise with classes like Meadows in the Pavia University dataset, Grapes untrained in the Salinas dataset, and Reeds in the Botswana dataset, where basic GCN models struggle. Advanced GCN models show improved classification accuracy in these cases. Remarkably, the multi-exit version surpasses all baseline models and achieves high accuracy levels.

As shown in Table 7, all baseline models performed better on the Salinas dataset, possibly due to its more regular boundaries. M-HyperGCN achieved perfect accuracies in nine land cover categories. The Vineyard untrained category posed challenges for most baseline models, including Funet-C and CEGCN, likely due to similarities in spectral signatures and spatial distributions with the Grapes untrained category. HyperGCN and M-HyperGCN improved accuracy by 11.93% over GCN in this category, while contrastive learning approaches helped CONGCN achieve the best score. In many other cases, such as Grass-trees and Stone-steel-towers in the Indian Pines dataset, Asphalt, Bricks, Meadows, Trees, and Shadows in the Pavia University dataset, and Lettuce romaines, 6 wk, Lettuce romaines, 5 wk, and Fallow smooth in the Salinas dataset, M-HyperGCN achieved significant improvements in accuracy, even achieving perfect accuracy in many instances. In contrast, CONGCN encountered challenges in accurately classifying these classes. The multi-layer graph convolution with oversmoothing-resistance and multi-exit strategy contributed significantly to these improvements.

5.2. Performance evaluation: additional layers and varied oversmoothing strategies

Experiments were conducted to analyse fluctuations in HS image classification accuracy as additional graph convolution layers were incorporated into the GNN model. The experiment included three GNN models: Multi-GCN (without oversmoothing-resistant techniques), GCNII (with initial residual and identity mapping), and HyperGCN (with PairNorm normalization). In this context, adding a graph convolution layer refers to adding a convolution block (Bi), as outlined in the proposed model's architecture. Each convolution block (Bi) consists of a dropout layer, a graph convolution layer, a normalization layer, and an activation layer, as shown in Figure 3(a). Examining the results of layer 1 for Multi-GCN, GCNII, and HyperGCN reveals minor differences in performance metrics. This discrepancy arises from the absence of oversmoothing-resistant techniques in Multi-GCN's first convolution block, while GCNII's first block employs initial residual and identity mapping instead of PairNorm.

Tables 9 and 10 illustrate the layer-wise accuracy fluctuations of Multi-GCN, GCNII, and HyperGCN on the Indian Pines and Pavia University datasets, respectively, highlighting performance degradation with added layers. As illustrated in Table 9, for the Indian Pines dataset, Multi-GCN's overall accuracy decreased from 68% at layer 1 to 25% at layer 50, and average accuracy dropped from 70% to 6%. The kappa coefficient fell from 0.6323 to 0.0472. GCNII initially improved up to 7 layers, with overall and average accuracy increasing from 71% and 64% to 71% and 73%, respectively. However, its performance declined thereafter, with overall and average accuracy at layer 50 recorded as 27% and 13%, respectively, and the kappa coefficient falling to 0.1703. Conversely, HyperGCN consistently demonstrated an enhancement and sustained high classification accuracy

Table 9. Evaluating the impact of increased layers on performance in Multi-GCN, GCNII, and HyperGCN using Indian Pines dataset.

	Multi-GCN			GCNII			HyperGCN		
Layers	OA	AA	k	OA	AA	k	OA	AA	k
1	67.62	69.57	0.6323	71.06	64.09	0.6287	71.56	65.39	0.6785
2	55.68	62.39	0.5059	76.12	73.28	0.729	76.33	75.91	0.7308
3	61.3	68.9	0.5615	74.81	78.73	0.7142	78.58	79.3	0.7441
4	57.14	60.73	0.525	73.27	78.57	0.6956	80.75	83.15	0.7775
5	56.26	57.49	0.5115	70.97	78.37	0.6725	**81.63**	**88.06**	**0.7905**
6	53.3	51.75	0.4795	68.82	76.7	0.6487	79.21	82.5	0.7512
7	51.89	51.64	0.4629	70.14	72.69	0.6613	78.22	80.4	0.7398
8	53.51	50.69	0.4814	66.25	68.59	0.6171	77.14	77.13	0.7378
9	51.87	47.17	0.4508	52.32	49.22	0.4467	77.95	82.11	0.7488
10	45.88	27.62	0.375	51.06	34.44	0.4117	76.98	75.7	0.7372
11	46.53	38.43	0.3873	51.77	47.63	0.431	77.66	76.05	0.7447
12	46.95	45.38	0.3977	47.19	33.53	0.3524	76.63	78.31	0.7333
15	49.13	31.72	0.3921	46.17	36.89	0.3067	77.13	79.06	0.739
20	20.67	5.92	0.024	43.11	24.31	0.3047	75.74	77.92	0.7243
35	24.98	6.28	0.0257	38.7	19.01	0.2229	76.22	80.44	0.729
50	24.99	6.25	0.0472	28.63	13.33	0.1703	74.84	78.28	0.7133

Table 10. Evaluating the impact of increased layers on performance in Multi-GCN, GCNII, and HyperGCN using Pavia University dataset.

	Multi-GCN			GCNII			HyperGCN		
Layers	OA	AA	k	OA	AA	k	OA	AA	k
1	78.98	80.28	0.7145	81.82	80.97	0.7289	89.1	81.18	0.8522
2	75.59	79.86	0.6758	79.02	80.97	0.7147	92.98	90.12	0.9068
3	75.39	78.03	0.6695	76.35	80.41	0.6879	93.56	90.76	0.9144
4	74.16	75.64	0.6863	78.05	75.50	0.6979	93.03	90.19	0.9076
5	73.53	72.42	0.6382	77.11	78.08	0.6896	**93.68**	**91.49**	**0.9159**
6	77.56	76.02	0.6898	77.58	76.05	0.6896	91.53	90.23	0.9008
7	73.89	73.6	0.6469	75.65	74.69	0.6655	91.52	89.19	0.8872
8	75.22	75.48	0.6629	73.44	77.35	0.6571	92.53	89.92	0.9005
9	73.52	74.38	0.6442	76.26	77.60	0.6797	91.29	88.69	0.8836
10	74.03	66.36	0.6463	76.14	74.63	0.6749	91.67	89.36	0.8891
11	76.91	77.01	0.6844	75.00	74.64	0.6577	91.37	88.97	0.8854
12	75.56	74.69	0.6640	74.37	71.87	0.6472	90.44	87.82	0.8722
15	73.85	70.03	0.6478	73.92	74.49	0.6490	90.80	88.62	0.8779
20	68.17	48.83	0.555	67.81	69.57	0.6343	91.45	89.12	0.8863
35	45.36	11.11	0.0215	50.39	41.16	0.4133	89.70	86.62	0.8629
50	35.36	09.17	0.0199	40.82	33.66	0.3287	89.75	86.74	0.8637

irrespective of the number of layers added. For instance, at layer 1, HyperGCN achieved an overall accuracy of 72% and an average accuracy of 65%, with a gradual improvement reaching 82% and 88% at layer 5. The kappa coefficient at layer 5 was 0.7905, reflecting its robust performance. This pattern persisted even at layer 50, where HyperGCN recorded an overall accuracy of 75% and an average accuracy of 78%. A similar trend is observed with the Pavia University dataset, as illustrated in Table 10. Multi-GCN's overall accuracy dropped from 79% at layer 1 to 35% at layer 50, and average accuracy decreased from 80% to 9%. The kappa coefficient fell from 0.7145 to 0.0199. GCNII experienced a modest decline, with overall and average accuracy decreasing from 82% and 81% at layer 1 to

76% and 75% at layer 7, and further to 41% and 34% at layer 50, with a kappa coefficient of 0.3287. HyperGCN improved steadily, starting at 89% overall and 81% average accuracy at layer 1, reaching 94% and 91% at layer 5, with a kappa coefficient of 0.9159. At layer 50, it maintained 90% overall and 87% average accuracy. In conclusion, the findings demonstrate that while Multi-GCN and GCNII experience significant performance degradation with increased layers, GCNII deteriorates at a slower rate than Multi-GCN. In contrast, HyperGCN consistently maintains high classification accuracy, effectively handling the oversmoothing problem. These results confirm HyperGCN's superior stability and reliability in deep-layered graph convolutional networks, making it a more suitable choice for applications requiring deep architectures. Additionally, the study highlights that PairNorm techniques perform better than initial residual and identity mapping techniques in handling the oversmoothing problem.

5.3. Performance evaluation with varying training set sizes

As illustrated in Table 11, HyperGCN demonstrates exceptional performance, particularly notable when trained with larger datasets. Evaluation metrics recorded on testing samples reveal the model's proficiency across various training set sizes. Training HyperGCN with just 5% of the Indian Pines dataset yields impressive results, achieving 82% overall accuracy, 88% average accuracy, and a kappa coefficient of 0.7905. Notably, as the training set size increases to 10% and 15%, the classification accuracy improves by 2% and 7%, respectively. Scaling further, when trained with 45% of the Indian Pines dataset, HyperGCN achieves remarkable results: 93% accuracy and a kappa coefficient of 0.9136. These trends are consistent when applying the model to variable-sized training samples from the Pavia University dataset. At 5% of the total dataset, HyperGCN attains a kappa coefficient of 0.9159, 94% overall accuracy, and 91% average accuracy. As the size of the training set grows, so does the classification performance, culminating in HyperGCN's outstanding achievement of 96% overall accuracy, 94% average accuracy, and a kappa coefficient of 0.9415 when trained with 45% of the training data.

5.4. Visual analysis

This section provides a visual comparison of evaluation results, focusing on classification maps and the overall accuracy achieved by various baseline models on the experimental

Table 11. Evaluating HyperGCN's performance by varying the training set sizes on the Indian Pines and Pavia University datasets.

#Training Set (%)	Indian Pines Dataset			Pavia University Dataset		
	OA	AA	k	OA	AA	k
5	81.85	88.49	0.7905	93.68	91.49	0.9159
10	83.37	88.75	0.8100	94.25	92.27	0.9237
15	86.1	88.47	0.8414	94.53	93.27	0.9276
20	87.30	89.53	0.8550	94.82	92.67	0.9313
25	87.43	88.80	0.8567	94.91	93.07	0.9324
30	88.02	89.23	0.8634	95.04	92.92	0.9343
35	89.45	89.99	0.8792	95.40	93.89	0.9393
40	89.97	90.33	0.8846	95.06	93.51	0.9345
45	92.50	92.49	0.9136	95.61	94.02	0.9415

dataset. It also includes a visual analysis of class-wise accuracy using a heatmap. Additionally, it presents an analysis of metrics such as Cohen's kappa coefficient, precision, recall, and F1 score obtained from both HyperGCN models and other basic GCN models across the experimental datasets. The impact of additional graph convolution layers on model performance is illustrated, followed by a discussion on the correlation between accuracy and training time.

5.4.1. Comparison of classification maps

Figures 4–7 show false-colour images, ground truth maps, and classification maps generated by HyperGCN models and baseline models for the Indian Pines, Pavia University, Botswana, and Salinas datasets, respectively. The classification maps vary in clarity across models, from shallow-layered GCN and MiniGCN to deeper models like Multi-GCN, GCNII, and HyperGCN. HyperGCN's classification maps exhibit reduced noise and superior clarity, comparable to advanced models like FuNet-C and CEGCN, due to its ability to capture high-order spectral features from neighbouring nodes through additional layers and oversmoothing-resistant techniques. The multiple exits in HyperGCN further aid in noise suppression, producing accurate maps similar to ConGCN. In Figure 4, classes such as Corn-notill, Grass-pasture, Grass-trees, and Stone-steel-towers exhibit more misclassifications in ConGCN's map compared to M-hyperGCN. However, regions like Buildings-grass-trees-drives pose challenges for smooth classification by HyperGCN and M-hyperGCN, often resulting in misclassifications as woods or Stone-steel-towers. The contrastive learning approach in ConGCN helps differentiate objects within this land cover type,

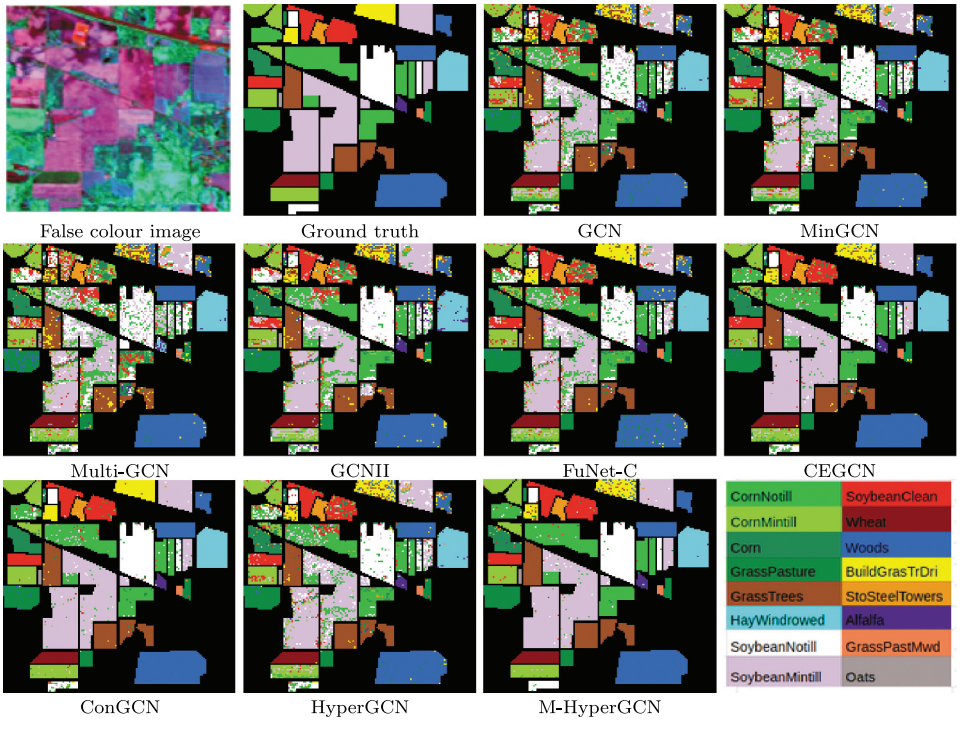


Figure 4. Classification maps of various baseline models with Indian Pines dataset.

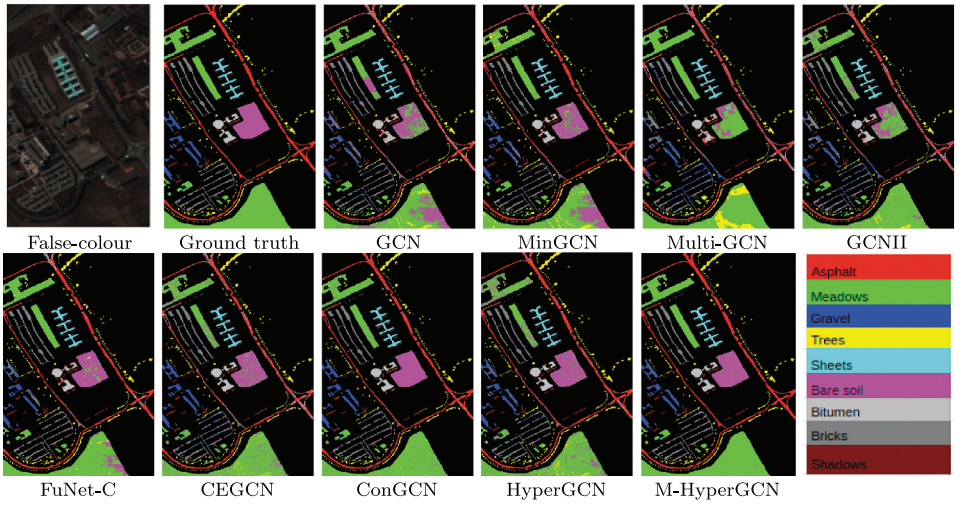


Figure 5. Classification maps of various baseline models with Pavia University dataset.

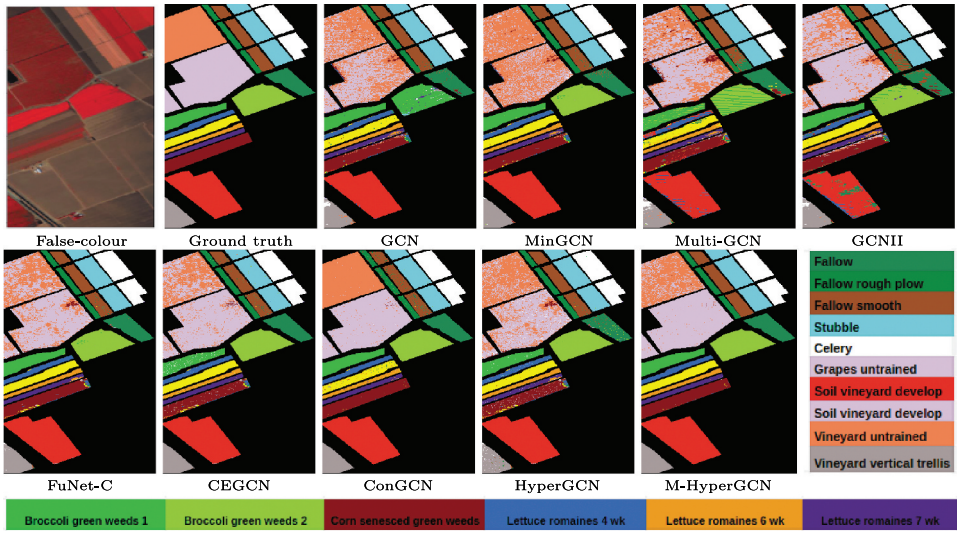


Figure 6. Classification maps of various baseline models with Salinas dataset.

resulting in a clearer classification map. Classes like Grass-trees, Wheat, and Oats in M-hyperGCN's classification map are entirely error-free and closely resemble the ground truth map.

In the context of the Pavia University dataset, both HyperGCN and M-HyperGCN demonstrated superior performance in generating less noisy and smoother classification maps compared to other baseline models. M-HyperGCN's classification map closely aligned with the ground truth (see Figure 5). Notably, M-HyperGCN produced a completely error-free classification map for the Shadows class, a challenge that advanced models such as CEGCN and ConGCN faced. However, challenges arose in classifying the Gravel region, where most baseline models yielded erroneous and noisy

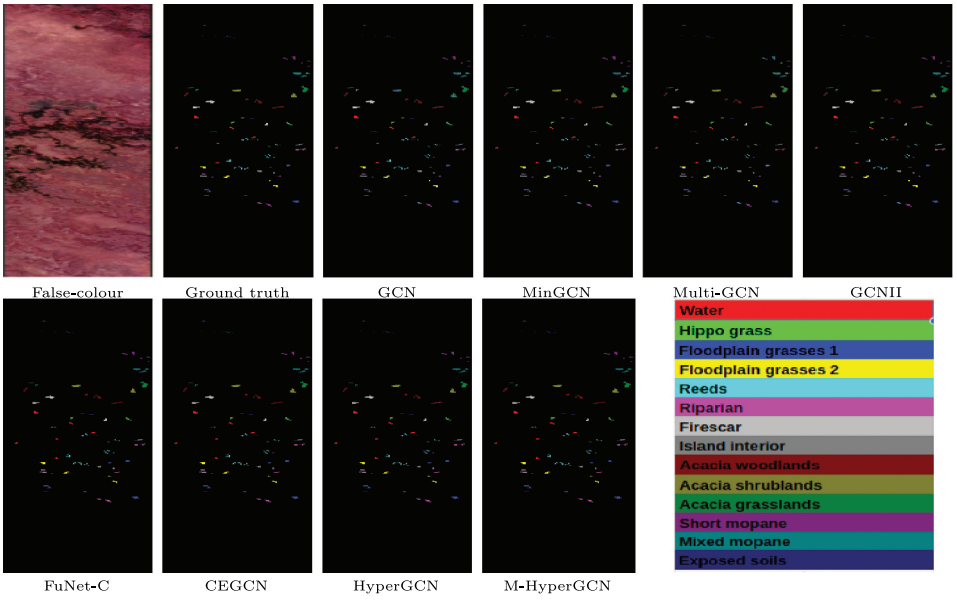


Figure 7. Classification maps of various baseline models with Botswana dataset.

results. Despite this, M-HyperGCN's Gravel region exhibited fewer errors compared to basic GCN models. Misclassification of some Gravel pixels as bricks was common among many models, except for CEGCN and ConGCN. This misclassification is attributed to the similar spectral signatures and spatial distribution patterns of Gravel and bricks. However, CEGCN and ConGCN models struggled to accurately classify bricks, as evident in the classification map where many bricks pixels were misclassified as Gravel. In contrast, M-HyperGCN's bricks region appeared clear and relatively error-free.

In the Salinas dataset, the classification map generated by M-HyperGCN is generally error-free, except for the Vineyard untrained land cover type. Some Vineyard untrained pixels are misclassified as Grapes untrained, a trend observed in most baseline models. However, both HyperGCN and M-HyperGCN generated relatively less noisy regions for this class. The use of positive and negative pairing of samples in ConGCN may have contributed to its better performance in this category. Figure 7 illustrates the classification maps generated by different models in the Botswana dataset. Since ConGCN's accuracy scores are from its original publication, which did not include the Botswana dataset, HyperGCN and M-HyperGCN are compared with other baseline models. With the incorporation of additional graph convolution layers and multiple exits, M-HyperGCN generated a classification map that closely matches the ground truth. In four land cover types, HyperGCN achieved completely error-free regions, while M-HyperGCN achieved 8 out of 14 error-free regions in the classification map.

5.4.2. Overall accuracy comparison

Figure 8 displays the overall classification accuracy achieved by the baseline models on the experimental datasets. The baseline GCN models, such as GCN and MiniGCN, delivered moderate performance but were surpassed by more advanced models with deeper architectures. Multi-GCN consistently showed poor performance across all datasets due to

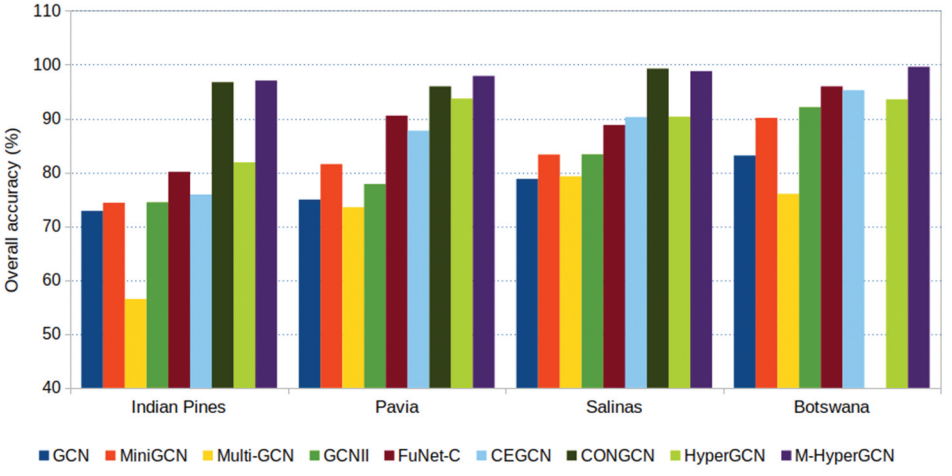
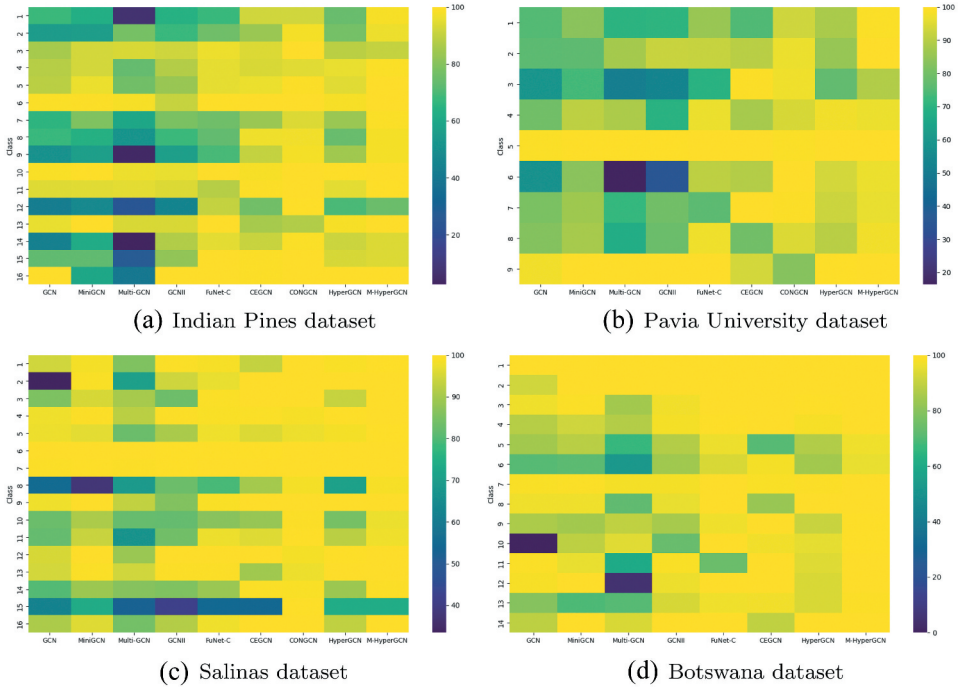


Figure 8. Overall accuracy of HyperGCN models and other baseline models on the experimental datasets.

the oversmoothing issue it encountered. FuNet-C and CEGCN exhibited stable and reliable performance, highlighting the efficacy of their CNN-GCN fusion approach. GCNII's performance varied across datasets, achieving moderate to good accuracy levels, with notable performance in the Botswana dataset (OA: 92.11% and AA: 93.22%). The incorporation of multiple graph convolutional layers and techniques to mitigate oversmoothing contributed to the strong performance of HyperGCN across datasets. It consistently outperformed basic GCN models such as GCN and MiniGCN, as well as FuNet-C and CEGCN in the Indian Pines, Pavia University, and Salinas datasets, and demonstrated comparable performance in the Botswana dataset. M-HyperGCN, an extension of HyperGCN with additional intermediate exits, consistently outperformed all other models across all datasets, specifically in the Indian Pines, Pavia University, and Botswana datasets. In the Salinas dataset, the most advanced model ConGCN showed a slight increment (0.46%) in overall accuracy over M-HyperGCN. The multi-exit nature of M-HyperGCN allows it to leverage intermediate prediction points, enhancing its adaptability to varying data complexities and structures. Overall, the inclusion of intermediate exits in M-HyperGCN has proven to be a valuable enhancement, contributing to its superior performance in HS image classification tasks.

5.4.3. Comparison on class-wise accuracy

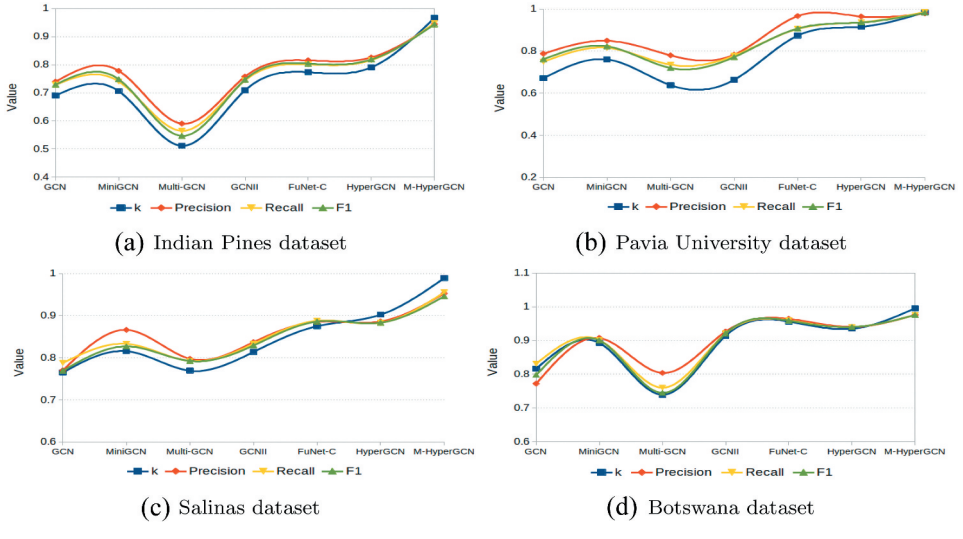
Figures 9(a–d), visually compare classification accuracy across individual classes on the Indian Pines, Pavia University, Salinas, and Botswana datasets, respectively, using heatmaps. These visualizations reveal the performance of different models on specific classes. For instance, classes like Stone-steel-towers (Class 13) in Indian Pines dataset and Shadows (Class 9) in Pavia University dataset showed discrepancies in performance among models. ConGCN struggled with these classes, while HyperGCN models performed well. The dark regions in the heatmaps for the multi-GCN model indicate oversmoothing issues, whereas the consistently bright areas for M-HyperGCN across most classes highlight its effectiveness in HS image classification.



(a) Indian Pines dataset (b) Pavia University dataset

(c) Salinas dataset (d) Botswana dataset

Figure 9. Comparison of individual class-wise accuracy of HyperGCN models with baseline models.



(a) Indian Pines dataset (b) Pavia University dataset

(c) Salinas dataset (d) Botswana dataset

Figure 10. Comparing Cohen's kappa coefficient, precision, recall, and F1 score of HyperGCN models with baseline models.

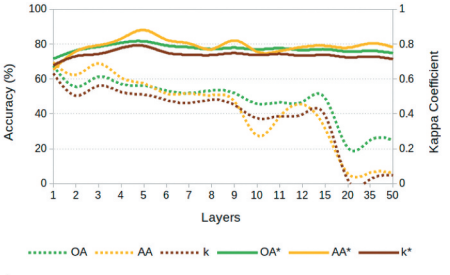
5.4.4. Comparison on other metrics

Figures 10(a–d), graphically depict the Cohen's kappa coefficient, precision, recall, and F1 score obtained from HyperGCN models and other baseline models on the experimental datasets. The curves for all performance metrics, including the kappa coefficient,

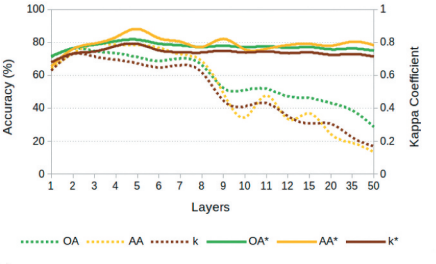
precision, recall, and F1 score, decline with Multi-GCN, suggesting that adding more layers to the GNN model without addressing the issue of oversmoothing degrades model performance. The slope of the curve increases with GCNII and HyperGCN, confirming that when the model is integrated with oversmoothing handling techniques, the performance metrics improve, and the curves ascend. Furthermore, the curve peaks with M-HyperGCN, emphasizing that the augmentation of intermediate exits into the HyperGCN architecture performed well across all datasets.

5.4.5. *Effect of additional graph convolution layers on model performance*

Figure 11 demonstrates how various models react to additional graph convolution layers. Figures 11(a,b) show how the performance of Multi-GCN and HyperGCN differs with additional layers on the Indian Pines and the Pavia University datasets, respectively. The graphs emphasize that multi-layered GNN models perform poorly in HS image classification when the oversmoothing issue is not addressed. Figures 11(c,d) demonstrate the performance of two oversmoothing handling techniques in HS image classification on the Indian Pines and the Pavia University datasets, respectively. The graphs illustrate how the oversmoothing resistant techniques based on the PairNorm approach (HyperGCN) outperform those based on initial residual and identity mapping techniques (GCNII).

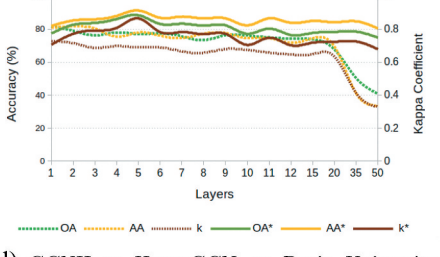


(a) Multi-GCN vs HyperGCN on Indian Pines dataset



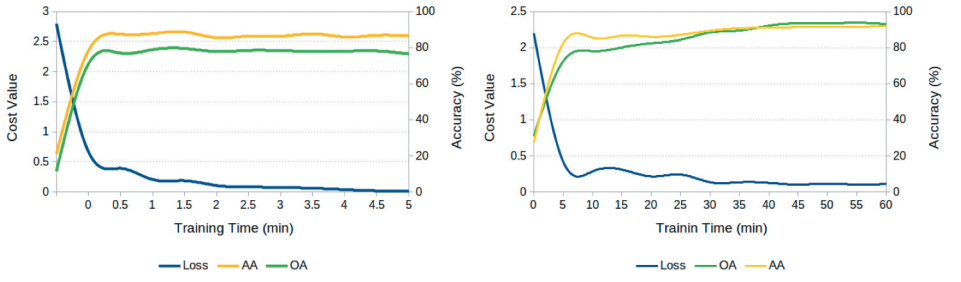
(b) Multi-GCN vs HyperGCN on Pavia University dataset

(c) GCNII vs HyperGCN on Indian Pines dataset



(d) GCNII vs HyperGCN on Pavia University dataset

Figure 11. Comparing the performance of Multi-GCN, GCNII, and HyperGCN with added graph convolution layers. Solid lines represent HyperGCN's performance (OA*, AA*, k*), while dotted lines represent Multi-GCN and GCNII (OA, AA, k).



(a) Indian Pines dataset (b) Pavia University dataset

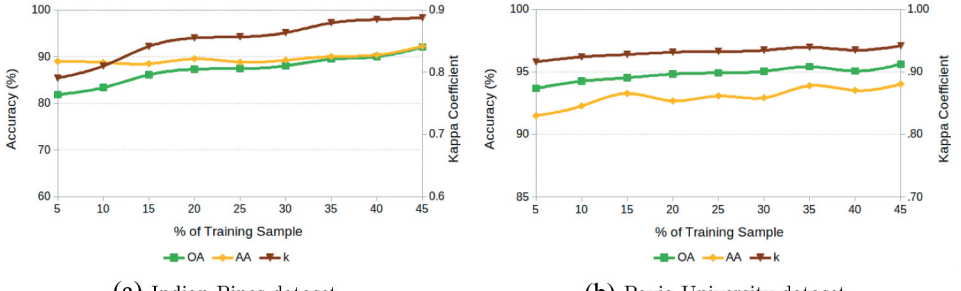
Figure 12. Comparing classification accuracies versus training time and cost-value versus training time on the Indian Pines and Pavia University datasets.

5.4.6. Correlation between accuracy and training time

The overall and average accuracy trends with training time are depicted in Figure 12 for the Indian Pines and Pavia University datasets. For the Indian Pines dataset (Figure 12(a)), the curve illustrates that as training progresses, classification accuracy steadily increases, reaching more than 85%. Within the first minute, average and overall accuracy reached 78% and 71%, respectively, and after 2 min, accuracy peaked at 89% and 80%. In contrast, Figure 12(b) demonstrates the accuracy versus training time for the Pavia University dataset, which required more time to achieve high accuracy compared to the Indian Pines dataset. In 5 min, overall and average accuracies of 72% and 82% were achieved, respectively, while the highest overall and average accuracy (88% and 91%) was attained after 45 min of training. Additionally, Figure 12 illustrates the cost value (cross-entropy loss) reduction with training time for both datasets. The value of the loss function decreases as the number of training iterations increases, indicating the convergence of the proposed model, HyperGCN. After 5 min of training, the loss value decreased to 0.0193 and 0.4179 on the Indian Pines and Pavia University datasets, respectively.

5.4.7. HyperGCN performance with varying training sizes

HyperGCN's performance on the Indian Pines and Pavia University datasets is evaluated with varying training set sizes, as illustrated in Figures 13(a,b).



(a) Indian Pines dataset (b) Pavia University dataset

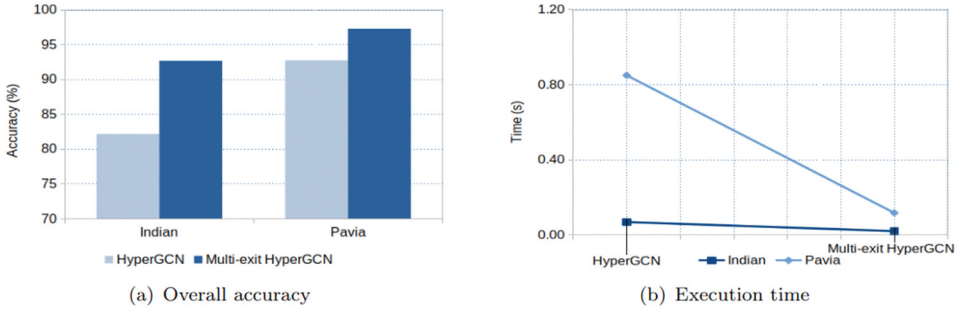
Figure 13. Performance evaluation with varying training set sizes. Here, OA, AA, and k represent the overall accuracy, average accuracy, and kappa coefficient, respectively.

HyperGCN performed well even with 5% training data samples, confirming its suitability for classification when training samples are limited. HyperGCN consistently provides higher accuracy and kappa coefficients as the training dataset size grows.

5.5. Comparative evaluation: HyperGCN vs. Multi-exit HyperGCN

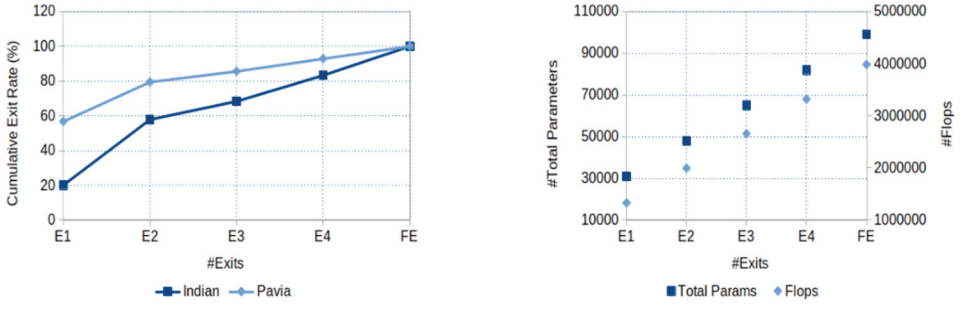
Figure 14(a) compares the overall accuracy of HyperGCN and M-HyperGCN models on the Indian Pines and Pavia University datasets. The multi-exit version outperforms the single-exit model, achieving higher accuracy rates by 10.52% and 4.53%, respectively. This highlights the effectiveness of additional side exit branches in improving accuracy in HS classification. The added flexibility of side exit branches enhances overall performance, offering a practical advantage for more accurate classifications. Figure 14(b), presents the execution time of HyperGCN and its multi-exit version on the Indian Pines and Pavia University datasets. Execution time refers to the total duration required for both models to complete the inference process on test samples. The multi-exit version demonstrates substantial speedup in execution time by 3.24× and 7.2×, respectively, on the Indian Pines dataset and the Pavia University dataset. These results underscore the remarkable efficiency gains achieved by integrating multi-exit strategies. The multi-exit version significantly accelerates inference by exiting most HS images early in the processing pipeline. The reduced execution times for multi-exit HyperGCN indicate that the flexibility introduced by side exit branches contributes to expedited processing, providing a practical advantage in scenarios where faster inference is crucial, such as real-time applications or resource-constrained environments.

Figure 15(a) presents the cumulative exit rate of all four exit branches and the final output layer, showing an upward trend in exit rates. This trend highlights the importance of early prediction in HS image classification. In the Indian Pines and Pavia University datasets, 20.09% and 56.72% of samples exit at the initial branch (E1), demonstrating the model's ability for early confident predictions. By the third exit branch (E3), 63.38% and 82.56% of samples have already terminated, emphasizing the model's efficiency without needing to reach the final stage. These high



(a) Overall accuracy (b) Execution time

Figure 14. The overall accuracy and execution time recorded by HyperGCN and M-HyperGCN on the Indian Pines and Pavia University datasets.



(a) Cumulative exit rate (b) Total parameters and Flops

Figure 15. The cumulative exit rate, total parameters, and floating-point operations (Flops) at each side exit branch for the Indian Pines and Pavia University datasets.

intermediate exit rates show the effectiveness of adding early-exit branches to standard GNN models, improving efficiency in HS image classification. This approach not only optimizes resource utilization but also reduces overall computational workload, leading to quicker inference and lower resource demands. Figure 15 shows the total parameters and floating-point operations (Flops) required by each exit branch (E1 to E4) and the final output layer (FE). Exiting at intermediate points, such as E2 and E3, reduces total parameters and Flops, enhancing computational efficiency and speeding up inference. Strategically exiting at these points balances accuracy and computational workload. The increase in total parameters and Flops from early to final exit points reflects the model's flexibility, allowing users to tailor computational resources to their needs, whether for real-time processing or accuracy-focused applications.

6. Conclusion

GNNs have demonstrated exceptional performance in various applications, including HS image classification. However, most studies favour shallow-layered GNN models due to performance degradation in deeper layers. This degradation is mainly caused by oversmoothing, where node embeddings of neighbouring nodes become too similar with repeated graph convolutions. Recognizing the potential of deeper models in extracting complex spectral and spatial information from HS images, this study introduces HyperGCN, an oversmoothing-resistant multi-layer GNN model that mitigates performance degradation by incorporating oversmoothing-resistant techniques. Evaluation on benchmark datasets such as the Indian Pines, Pavia University, Salinas, and Botswana demonstrates HyperGCN's effectiveness and superiority over basic GCN-based models. Additionally, this study integrates side exit branches into the intermediate layers of HyperGCN, resulting in M-HyperGCN. These intermediate exit branches enhance both efficiency and accuracy. M-HyperGCN consistently outperforms its single-exit counterpart, achieving higher overall accuracy rates. It also surpasses baseline models across multiple datasets. By allowing most samples to exit through intermediate exits,

M-HyperGCN offers a solution to the oversmoothing issue of GCNs. Our analysis reveals the intricate nature of HS image classification, where certain land cover types pose inherent challenges due to spectral similarities and spatial complexities. While basic GCN models struggled with such cases, HyperGCN and M-HyperGCN showcased remarkable adaptability and performance enhancements. Moreover, M-HyperGCN significantly accelerates execution time by 3.24× on the Indian Pines dataset and 7.2× on the Pavia University dataset. This highlights its practical advantage in scenarios requiring faster inference, such as real-time applications or resource-constrained environments. These results underscore the effectiveness of augmenting GNN models with additional early-exit branches, providing a balanced solution between accuracy and computational efficiency in HS image classification tasks.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Haseena Rahmath P http://orcid.org/0000-0003-1128-5964

Data availability statement

We have utilized publicly available datasets that can be found at https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes.

Preprint citation

A preprint version of this work was previously shared on Research Square [https://doi.org/10.21203/rs.3.rs-3679445/v1].

References

Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. 2012. "SLIC Superpixels Compared to State-of-The-Art Superpixel Methods." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 34 (11): 2274–2282. https://doi.org/10.1109/TPAMI.2012.120.

Bronstein, M. M., J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. 2017. "Geometric Deep Learning: Going Beyond Euclidean Data." *IEEE Signal Processing Magazine* 34 (4): 18–42. https://doi.org/10.1109/MSP.2017.2693418.

Chen, M., Z. Wei, Z. Huang, B. Ding, and Y. Li. 2020. "Simple and Deep Graph Convolutional Networks." *International Conference on Machine Learning*, Originally scheduled in Vienna, Austria, held virtually, 1725–1735. PMLR.

Chen, Y., H. Jiang, C. Li, X. Jia, and P. Ghamisi. 2016. "Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks." *IEEE Transactions on Geoscience & Remote Sensing* 54 (10): 6232–6251. https://doi.org/10.1109/TGRS.2016.2584107.

Chen, Y., Z. Lin, X. Zhao, G. Wang, and Y. Gu. 2014. "Deep Learning-Based Classification of Hyperspectral Data." *IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing* 7 (6): 2094–2107. https://doi.org/10.1109/JSTARS.2014.2329330.

Gasteiger, J., A. Bojchevski, and S. Günnemann. 2019. "Predict Then Propagate: Graph Neural Networks Meet Personalized Pagerank." The 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA. https://doi.org/10.48550/arXiv.1810.05997

Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. 2017. "Neural Message Passing for Quantum Chemistry." *International conference on machine learning*, Sydney, Australia, 1263–1272. PMLR.

Gori, M., G. Monfardini, and F. Scarselli. 2005. "A New Model for Learning in Graph Domains." In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005*, Montreal, QC, Canada, 729–734.

Hamilton, W. L., R. Ying, and J. Leskovec. 2017. "Inductive Representation Learning on Large Graphs." The 31st International Conference on Neural Information Processing Systems (NIPS'17), Red Hook, NY, USA, 1025–1035. Curran Associates Inc.

Hammond, D. K., P. Vandergheynst, and R. Gribonval. 2011. "Wavelets on Graphs via Spectral Graph Theory." *Applied and Computational Harmonic Analysis* 30 (2): 129–150. https://doi.org/10.1016/j.acha.2010.04.005.

Hang, R., F. Zhou, Q. Liu, and P. Ghamisi. 2020. "Classification of Hyperspectral Images via Multitask Generative Adversarial Networks." *IEEE Transactions on Geoscience & Remote Sensing* 59 (2): 1424–1436. https://doi.org/10.1109/TGRS.2020.3003341.

Harsanyi, J. C., and C.-I. Chang. 1994. "Hyperspectral Image Classification and Dimensionality Reduction: An Orthogonal Subspace Projection Approach." *IEEE Transactions on Geoscience & Remote Sensing* 32 (4): 779–785. https://doi.org/10.1109/36.298007.

Haseena Rahmath, P., V. Srivastava, and K. Chaurasia. 2023. "A Strategy to Accelerate the Inference of a Complex Deep Neural Network." In *Proceedings of Data Analytics and Management: ICDAM 2022*, Virtual conference, 57–68. Springer.

Hong, D., L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot. 2020. "Graph Convolutional Networks for Hyperspectral Image Classification." *IEEE Transactions on Geoscience & Remote Sensing* 59 (7): 5966–5978. https://doi.org/10.1109/TGRS.2020.3015157.

Hong, D., N. Yokoya, J. Chanussot, and X. Xiang Zhu. 2019. "CoSpace: Common Subspace Learning from Hyperspectral-Multispectral Correspondences." *IEEE Transactions on Geoscience & Remote Sensing* 57 (7): 4349–4359. https://doi.org/10.1109/TGRS.2018.2890705.

Huang, K., S. Li, X. Kang, and L. Fang. 2016. "Spectral–Spatial Hyperspectral Image Classification Based on KNN." *Sensing and Imaging* 17 (1): 1–13. https://doi.org/10.1007/s11220-015-0126-z.

Hwang, H., S. Lee, C. Park, and K. Shin. 2022. "AHP: Learning to Negative Sample for Hyperedge Prediction." The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022), Madrid, Spain, 2237–2242. https://doi.org/10.1145/3477495.3531836 .

Jia, S., S. Jiang, S. Zhang, M. Xu, and X. Jia. 2024. "Graph-In-Graph Convolutional Network for Hyperspectral Image Classification." *IEEE Transactions on Neural Networks and Learning Systems* 35:1157–1171. https://doi.org/10.1109/TNNLS.2022.3182715.

Kaya, Y., S. Hong, and T. Dumitras. 2019. "Shallow-Deep Networks: Understanding and Mitigating Network Overthinking." *International conference on machine learning (ICML-2019)*, Long Beach, California, USA, 3301–3310. PMLR.

Khodadadzadeh, M., J. Li, A. Plaza, and J. M. Bioucas-Dias. 2014. "A Subspace-Based Multinomial Logistic Regression for Hyperspectral Image Classification." *IEEE Geoscience & Remote Sensing Letters* 11 (12): 2105–2109. https://doi.org/10.1109/LGRS.2014.2320258.

Kipf, T. N., and M. Welling. 2017. "Semi-Supervised Classification with Graph Convolutional Networks." The 5th International Conference on Learning Representations (ICLR 2017), Toulon, France. https://doi.org/10.48550/arXiv.1609.02907.

Kraemer, H. C. 2015. "Kappa Coefficient." Wiley StatsRef: Statistics Reference Online, edited by N. Balakrishnan, T. Colton, and B. Everitt, 1–4. Hoboken, NJ, USA: Wiley.

Landgrebe, D. 2002. "Hyperspectral Image Data Analysis." *IEEE Signal Processing Magazine* 19 (1): 17–28. https://doi.org/10.1109/79.974718.

Li, G., M. Muller, A. Thabet, and B. Ghanem. 2019. "Deepgcns: Can Gcns Go As Deep As Cnns?" *Proceedings of the IEEE/CVF international conference on computer vision*, Seoul, Korea, 9267–9276.

Li, Q., Z. Han, and X.-M. Wu. 2018. "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning." In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, edited by S. A. McIlraith and K. Q. Weinberger, 3538–3545. New Orleans, Louisiana, USA: AAAI Press.

Li, Y., D. Tarlow, M. Brockschmidt, and R. Zemel. 2016. "Gated Graph Sequence Neural Networks." The 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico. https://doi.org/10.48550/arXiv.1511.05493.

Liu, Q., L. Xiao, J. Yang, and Z. Wei. 2021. "CNN-Enhanced Graph Convolutional Network with Pixel- and Superpixel-Level Feature Fusion for Hyperspectral Image Classification." *IEEE Transactions on Geoscience & Remote Sensing* 59 (10): 8657–8671. https://doi.org/10.1109/TGRS.2020.3037361.

Liu, Q., F. Zhou, R. Hang, and X. Yuan. 2017. "Bidirectional-Convolutional LSTM Based Spectral-Spatial Feature Learning for Hyperspectral Image Classification." *Remote Sensing* 9 (12): 1330. https://doi.org/10.3390/rs9121330.

Marion, F., L. B. L. Baumgardner, and A. Landgrebe David. 2015. "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3." https://purr.purdue.edu/publications/1947/1.

Melgani, F., and L. Bruzzone. 2004. "Classification of Hyperspectral Remote Sensing Images with Support Vector Machines." *IEEE Transactions on Geoscience & Remote Sensing* 42 (8): 1778–1790. https://doi.org/10.1109/TGRS.2004.831865.

Panda, P., A. Sengupta, and K. Roy. 2017. "Energy-Efficient and Improved Image Recognition with Conditional Deep Learning." *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 13 (3): 1–21. https://doi.org/10.1145/3007192.

Patel, H., and K. P. Upla. 2022. "A Shallow Network for Hyperspectral Image Classification Using an Autoencoder with Convolutional Neural Network." *Multimedia Tools & Applications* 81 (1): 695–714. https://doi.org/10.1007/s11042-021-11422-w.

Pham, T., T. Tran, D. Phung, and S. Venkatesh. 2017. "Column Networks for Collective Classification." *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17*, San Francisco, California, USA, 2485–2491. AAAI Press.

Qin, A., Z. Shang, J. Tian, Y. Wang, T. Zhang, and Y. Yan Tang. 2018. "Spectral–Spatial Graph Convolutional Networks for Semisupervised Hyperspectral Image Classification." *IEEE Geoscience & Remote Sensing Letters* 16 (2): 241–245. https://doi.org/10.1109/LGRS.2018.2869563.

Rahimi, A., T. Cohn, and T. Baldwin. 2018. "Semi-Supervised User Geolocation via Graph Convolutional Networks." The 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), edited by I. Gurevych and Y. Miyao, Melbourne, Australia, 2009–2019. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1187.

Rong, Y., W. Huang, T. Xu, and J. Huang. 2020. "Dropedge: Towards Deep Graph Convolutional Networks on Node Classification." International Conference on Learning Representations (ICLR 2020), Virtual Conference. https://openreview.net/pdf?id=Hkx1qkrKPr .

Shahraki, F. F., and S. Prasad. 2018. "Graph Convolutional Neural Networks for Hyperspectral Data Classification." In *2018 IEEE global conference on signal and information processing (GlobalSIP)*, Anaheim, CA, USA, 968–972. IEEE. https://doi.org/10.1109/GlobalSIP.2018.8645969.

Tang, L., and H. Liu. 2009. "Relational Learning via Latent Social Dimensions." In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France, 817–826.

Teerapittayanon, S., B. McDanel, and H.-T. Kung. 2016. "Branchynet: Fast Inference via Early Exiting from Deep Neural Networks." In *2016 23rd International Conference on Pattern Recognition (ICPR)*, Cancun, Mexico, 2464–2469. IEEE.

Ud Din, A. M., and S. Qureshi. 2024. "Limits of Depth: Over-Smoothing and Over-Squashing in GNNs." *Big Data Mining & Analytics* 7 (1): 205–216. https://doi.org/10.26599/BDMA.2023.9020019.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. 2018. "Graph Attention Networks." International Conference on Learning Representations (ICLR 2018), Vancouver, Canada. https://openreview.net/pdf?id=rJXMpikCZ.

Wan, S., C. Gong, P. Zhong, S. Pan, G. Li, and J. Yang. 2020. "Hyperspectral Image Classification with Context-Aware Dynamic Graph Convolutional Network." *IEEE Transactions on Geoscience & Remote Sensing* 59 (1): 597–612. https://doi.org/10.1109/TGRS.2020.2994205.

Wang, L., and X. Wang. 2022. "Dual-Coupled CNN-GCN-Based Classification for Hyperspectral and LiDAR Data." *Sensors* 22 (15): 5735. https://doi.org/10.3390/s22155735.

Wang, Y., Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. 2019. "Dynamic Graph Cnn for Learning on Point Clouds." *ACM Transactions on Graphics (TOG)* 38 (5): 1–12. https://doi.org/10.1145/3326362.

Wu, F., A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. 2019. "Simplifying Graph Convolutional Networks." In *International conference on machine learning*, Long Beach, CA, USA, 6861–6871. PMLR.

Xu, K., C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka. 2018. "Representation Learning on Graphs with Jumping Knowledge Networks." In *International conference on machine learning*, Stockholm, Sweden, 5453–5462. PMLR.

Yan, L., M. Zhao, X. Wang, Y. Zhang, and J. Chen. 2021. "Object Detection in Hyperspectral Images." *IEEE Signal Processing Letters* 28:508–512. https://doi.org/10.1109/LSP.2021.3059204.

Ying, R., R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. 2018. "Graph Convolutional Neural Networks for Web-Scale Recommender Systems." In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, London, United Kingdom, 974–983.

Yu, Q., W. Wei, Z. Pan, J. He, S. Wang, and D. Hong. 2023. "GPF-Net: Graph-Polarized Fusion Network for Hyperspectral Image Classification." *IEEE Transactions on Geoscience & Remote Sensing* 61:1–22. https://doi.org/10.1109/TGRS.2023.3304311.

Yu, W., S. Wan, G. Li, J. Yang, and C. Gong. 2023. "Hyperspectral Image Classification with Contrastive Graph Convolutional Network." *IEEE Transactions on Geoscience & Remote Sensing* 61:1–15. https://doi.org/10.1109/TGRS.2023.3240721.

Zhang, M., Z. Cui, M. Neumann, and Y. Chen. 2018. "An End-To-End Deep Learning Architecture for Graph Classification." In *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA, 32.

Zhao, L., and L. Akoglu. 2020. "Pairnorm: Tackling Oversmoothing in Gnns." The Eighth International Conference on Learning Representations (ICLR 2020), Virtual Conference. https://openreview.net/pdf?id=rkecl1rtwB .

Zhuang, H., C. Wang, C. Li, Q. Wang, and X. Zhou. 2017. "Natural Language Processing Service Based on Stroke-Level Convolutional Networks for Chinese Text Classification." In *2017 IEEE international conference on web services (ICWS)*, Honolulu, Hawaii, USA, 404–411. IEEE.