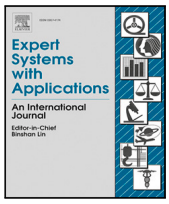
Expert Systems With Applications 275 (2025) 127046

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Joint U-Nets with hierarchical graph structure and sparse Transformer for hyperspectral image classification

Pengfei Zhu, Jinglei Liu *

School of Computer and Control Engineering, Yantai University, Yantai 264005, Shandong, China

ARTICLE INFO

Keywords:
Hyperspectral image (HSI) classification
Hierarchical graph merging block
Sparse transformer block
Sparse self-attention mechanism
U-net

ABSTRACT

Hyperspectral image (HSI) classification is a critical task in remote sensing, but existing U-Net and transformer-based models encounter significant challenges. Traditional U-Net architectures struggle with multi-scale feature extraction due to their fixed convolutional kernels, limiting their effectiveness in capturing complex spatial distributions. Transformer models, while adept at capturing global context, suffer from high computational complexity and inadequate sensitivity to local features in HSI. To address these limitations, we propose a novel joint U-Nets with hierarchical graph structure and sparse transformer (HGSTNet). HGSTNet introduces hierarchical graph merging blocks and incremental merging methods to dynamically extract and fuse multi-scale features, leveraging superpixel segmentation and hierarchical graph topology to enhance spatial correlation. Furthermore, to enhance the model's global context perception, we integrate a sparse transformer block in the first four encoder–decoder. Unlike traditional transformers, the sparse transformer reduces computational complexity and enhances feature capture by incorporating the sparse self-attention (SPA) module, which utilizes the sparse self-attention mechanism to suppress low-relevance or redundant information, thereby improving the capture of both local and global features. Experiments conducted on multiple HSI datasets, along with comparisons to other deep learning methods, demonstrate that HGSTNet exhibits strong competitiveness.

1. Introduction

Hyperspectral imaging, which captures hundreds or even thousands of spectral bands, provides a unique perspective by combining spatial and spectral information. Unlike traditional imaging methods, HSI enables the identification and differentiation of materials and objects based on their unique spectral signatures. This rich spectral data facilitates accurate classification, identification, and monitoring of diverse land cover types and environmental conditions. With its ability to offer unprecedented levels of detail, HSI has become indispensable across a wide range of applications, including earth observation, military intelligence, medical diagnostics, and more (Adão et al., 2017; Munipalle, Nelakuditi, CVSS, & Nidamanuri, 2024; Oswald et al., 2024; Ramakrishnan & Bharti, 2015).

Convolutional neural networks (CNNs) and graph convolutional networks (GCNs) have emerged as powerful tools for extracting semantic features from HSIs. CNNs are particularly adept at encoding spectral–spatial information (Zhong, Li, Luo, & Chapman, 2018), making them suitable for capturing the unique characteristics of HSIs. Typically, CNN-based models employ multi-layer perceptrons as classifiers to process the extracted features. However, CNNs face challenges such as vanishing and exploding gradients, which can hinder their training on deep architectures. To address these issues, researchers have introduced residual connection mechanisms (He, Zhang, Ren, & Sun, 2016; Huang, Liu, Van Der Maaten, & Weinberger, 2017) and dense connection mechanisms (Wang, Dou, Jiang, & Sun, 2018) into CNN frameworks. These enhancements allow for more efficient gradient flow during backpropagation, enabling deeper network structures to be effectively trained. Further advancements in CNNs include deformable convolutional networks (DCNNs) (Xu, Ren, Liu, & Jia, 2014), which improve the precision of local feature extraction by dynamically adjusting the convolution kernel shape based on the spatial structure of HSIs. Additionally, dual-branch dual-attention (DBDA) networks (Li, Zheng, Duan, Yang, & Wang, 2020) enhance CNN performance by employing dual-attention mechanisms. These mechanisms selectively focus on relevant spatial and spectral information, thereby improving classification accuracy in complex HSI datasets. On the other hand, graph convolutional networks (GCNs) have gained significant attention for their ability to model relationships between data points through graph structures. GCNs effectively learn representations of nodes and edges, making them well-suited for datasets with arbitrary and non-Euclidean structures (Hong et al., 2021; Mou, Lu, Li, & Zhu, 2020; Qin

* Corresponding author.
E-mail addresses: 15293128265@s.ytu.edu.cn (P. Zhu), liujinglei@ytu.edu.cn (J. Liu).

https://doi.org/10.1016/j.eswa.2025.127046
Received 3 December 2024; Received in revised form 18 January 2025; Accepted 23 February 2025
Available online 5 March 2025

et al., 2019). A key innovation in this domain is the dynamic graph convolutional operation (Wan et al., 2020), which continuously updates the similarity measures between pixels based on their current feature embeddings. By constructing an adaptive graph structure, this approach captures spatial and spectral correlations more comprehensively, leading to improved classification accuracy and robustness in diverse HSI datasets.

With the continuous evolution of network architectures, U-Net-based models have demonstrated significant potential in HSI classification. For example, the nested U-Net architecture (Liu, Yu et al., 2022), which integrates global contextual information, achieves high-precision classification performance when applied to datasets with limited samples. However, its generalization capability is limited when processing datasets with diverse spatial and spectral patterns or a large number of samples. To address this issue, a dual-branch U-Net architecture (Zhang, Liu, Yang, & Wu, 2023) was proposed. This model separately processes low-resolution HSIs and high-resolution multispectral images, incorporating a spectral–spatial feature interaction module to effectively fuse cross-modal information. Further advancements include the CAGU model (Lin, Jing, Di, Chen, & Song, 2022), which was developed to tackle classification challenges by learning and leveraging intra-class variability and inter-class similarity in HSIs. This model improves the understanding of complex relationships within HSIs, resulting in enhanced classification accuracy. However, the superpixel-based GCN utilized in the CAGU model generates features at the superpixel level, which limits its ability to capture fine-grained pixel-level details, thus constraining its classification performance.

In recent years, transformer-based network models have been widely applied to HSI due to their capability to capture long-range dependencies, which is particularly beneficial for modeling spatial and spectral relationships between pixels. For example, the MSTNet model (Yang et al., 2022) employs a transformer encoder to effectively capture long-range dependencies, addressing limitations in earlier methods regarding the utilization of long-range information in HSIs. Similarly, the swin transformer architecture has been adopted as the main network backbone (Yang, Wang, Zhao, Song, & Zhang, 2023), leveraging global feature extraction capabilities to capture spatial information in HSI. Furthermore, (Sun, Zhao, Zheng, & Wu, 2022) introduced a spectral–spatial feature extraction module integrated with a transformer encoder to efficiently extract both low-level spectral–spatial features and high-level semantic features. However, despite these advancements, transformer-based models still face challenges, including insensitivity to local feature information, high computational complexity, and a large number of parameters.

To effectively address the aforementioned challenges, we propose a novel model called Joint U-Nets with hierarchical graph structure and sparse transformer (HGSTNet). HGSTNet combines the strengths of hierarchical graph merge blocks and sparse transformer blocks to tackle the complexities of HSI classification. The hierarchical graph merge blocks employ a hierarchical graph incremental merging method to dynamically build multi-scale graph topologies. This enables the model to capture local spatial–spectral relationships and enhance spatial correlation, forming a solid foundation for further feature learning. Complementing this, the sparse transformer blocks leverage a sparse self-attention mechanism to capture long-range dependencies and global spectral relationships. Together, these modules form a hybrid architecture that collaboratively extracts local and global features. The workflow of HGSTNet begins with principal component analysis (PCA) for dimensionality reduction of HSIs, followed by hyperpixel segmentation using superpixels on the primary components (Wei, Yang, Gong, Ahuja, & Yang, 2018). Subsequently, the core method incrementally merges adjacent region superpixels to construct hierarchical graph topologies, enabling the extraction of multi-scale spatial–spectral relationships. These graph-based operations are augmented by CNN-based pixel-level feature extraction within an encoder–decoder framework, preserving fine-grained pixel-level details while enriching them with contextual information. By skip connections between the encoder and decoder further ensure that pixel-level details are retained during the feature fusion process, preventing the loss of valuable information. Furthermore, after feature extraction, the sparse transformer blocks refine and fuse features through a sparse self-attention mechanism, resulting in a more comprehensive feature representation. As depicted in Fig. 1, the input HSI is first processed by the encoder, where a series of operations extract image features. These features are then passed through the sparse transformer blocks to obtain more detailed multi-scale representations. The decoder integrates these features through fusion, with skip connections between the encoder and decoder ensuring the preservation of image details and preventing valuable information loss. Additionally, the feature fusion process is optimized to adaptively integrate features across different scales, effectively enhancing spatial adaptivity for pixel classification tasks. This hybrid U-Net framework not only retains the local feature extraction and fusion capabilities of traditional U-Nets but also incorporates the global information learning capabilities of transformers. By seamlessly integrating multi-level feature extraction, HGSTNet achieves a high level of efficiency and adaptability in HSI classification. Comprehensive experiments conducted on multiple benchmark datasets demonstrate the superior performance and robustness of HGSTNet.

The main contributions of this work are summarized as follows.

(1) We propose the HGSTNet model for HSI classification, which innovatively integrates the local feature extraction capabilities of the hierarchical graph U-Net with the global context awareness of the sparse transformer. This fusion enables HGSTNet to effectively capture both local and global features at multiple spatial scales, improving classification performance compared to conventional models, such as standard U-Net and transformer-based approaches.
(2) Considering the limitations of fixed convolutional kernels in U-Net and the need for extracting global land cover features in HSI, we introduce hierarchical graph merging blocks and incremental merging methods into the U-Net architecture. By extracting multi-scale features, HGSTNet can flexibly captures the spatial distribution of different land cover types, forming superpixel regions $\mathbf{S}^{(l)}$ as shown in Eq. (1), and simultaneously build the corresponding hierarchical graph topology $\mathbf{G}^{(l)}$ as shown in Eq. (2), effectively enhance spatial correlation and context representation.
(3) To address the issues of traditional transformer models being insensitive to local features and having high computational complexity in HSI processing, we design a sparse self-attention (SPA) module. The SPA module introduces a sparse self-attention mechanism that selectively suppresses low-relevance information and reduces computational cost, capturing long-range dependencies at the minimal possible computational expense.
(4) To validate the effectiveness of the HGSTNet model, we conducted extensive experiments on several publicly available HSI datasets. The results, as shown in Table 5, 6, 7, demonstrate that HGSTNet consistently achieved superior performance across a range of experiments. Compared to other state-of-the-art models, HGSTNet exhibits improvements in overall accuracy (OA), average accuracy (AA), and kappa coefficient (KPP), showcasing its effectiveness in HSI classification tasks.

The rest of this article is organized as follows. In Section 2, we introduce works related to HSIs, graph U-Net, and vision transformers, providing a foundation for the subsequent research. Section 3 briefly describes the entire network architecture of HGSTNet, emphasizing the construction of hierarchical graph merge blocks, the design of the sparse transformer block, the details of the sparse self-attention mechanism, the loss function, the optimization process, and the algorithm description. Section 4 presents various experimental results and analyses. Finally, Section 5 summarizes the conclusions and discusses future work.

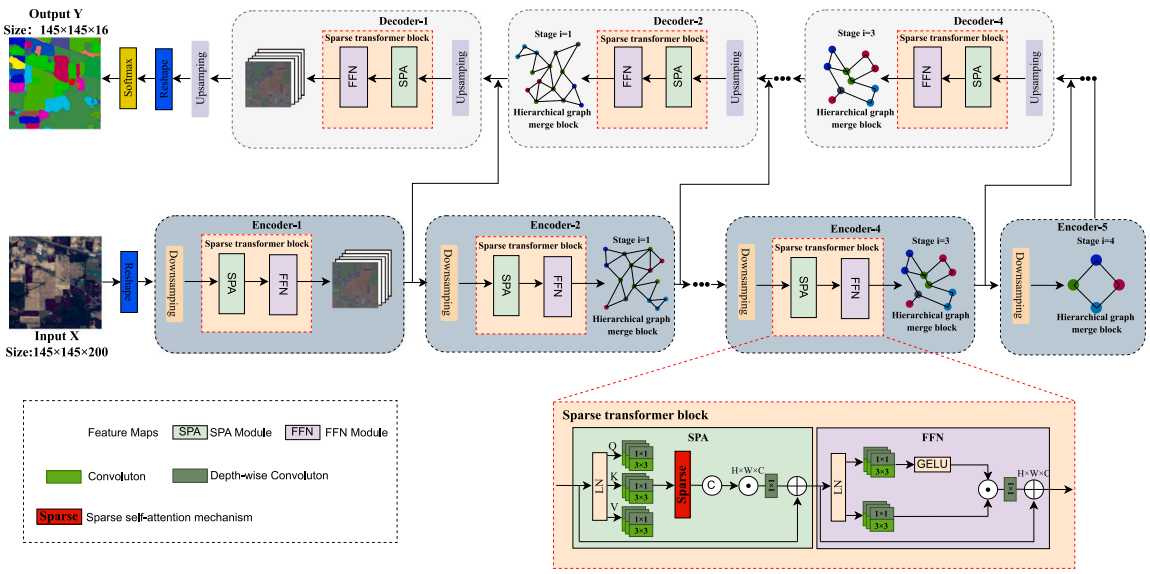


Fig. 1. The flowchart of our HGSTNet is as follows: an input HSI (X) is processed through the network to generate an output probability map (Y). The model takes the entire HSI as input (e.g., the IP dataset). During training, only a subset of labeled pixels is dynamically selected to construct the training set for each iteration, while the remaining labeled and unlabeled pixels are reserved for validation and testing. To adapt to various HSI datasets, the number of nodes in the second-level encoder is set to 2048, with a feature map size of [128 × 128]. For subsequent encoder stages, both the node count and feature map size are halved sequentially, resulting in [2048, 1024, 512, 256] nodes and corresponding sizes of [128 × 128], [64 × 64], [32 × 32], and [16 × 16]. Symmetrically, the decoder stages restore the feature maps with node counts of [256, 512, 1024, 2048] and corresponding feature map sizes, ultimately recovering the original spatial dimensions of [145 × 145]. HGSTNet consists of five encoder stages (Encoder 1 to Encoder 5) and five decoder stages (Decoder 1 to Decoder 5), with corresponding hierarchical graph merge blocks at each stage. Each encoder–decoder pair includes a sparse transformer block and a hierarchical graph merge block. The sparse transformer block, consisting of the SPA and FFN modules, is responsible for enhancing spatial and spectral feature dependencies. The construction process of the hierarchical graph merge block is illustrated in Fig. 2.

2. Related work

In this section, we will introduce three topics closely related to this work: HSI classification, graph U-Net, and vision transformer. Additionally, to facilitate a better understanding of the subsequent content, Table 1 provides a complete definition of each symbol used throughout this paper.

2.1. Hyperspectral image classification

Hyperspectral image (HSI) classification methods are generally categorized into pixel-based and object-based approaches. Pixel-based methods (Varma, Rao, Raju, & Varma, 2016) classify each pixel independently, often ignoring spatial relationships. This can lead to inaccurate classifications, especially due to noise and lack of contextual information. To address these limitations, researchers have explored strategies to incorporate spatial information and reduce noise sensitivity. For instance, MSSGU (Liu, Xiao, Yang, & Wei, 2022) uses a multi-scale hierarchical approach to learn spatial features from different levels. However, pixel-based methods remain sensitive to noise, which can negatively impact accuracy. To mitigate this, some studies propose preprocessing techniques, such as the A-HySN network (Fan, Zhang, Chen, & Sun, 2023), which performs pixel-level similarity analysis to replace dissimilar pixels, reducing interference. Additionally, the SGAT module (Li, Liu, Fan, Bai, & Xin, 2023) uses spectral sparsity to trim graphs, eliminating noise and redundant information. Object-based classification (Hong & Zhang, 2020) segments the image into regions or objects by grouping adjacent pixels with similar semantics, and classifies these objects. This method is advantageous over pixel-based approaches, as it better accounts for spatial relationships and reduces noise sensitivity. By grouping semantically similar pixels, object-based methods exploit spatial continuity and correlation, leading to improvements in classification accuracy. For example, LMF-BBNet (Shi, Chen, & Wang, 2024) employs late fusion to integrate low- and high-level features at multiple stages, enhancing classification performance. The RDTN model (Li, Yang, Tang & Zhou, 2024), by combining the CSCA and LRTB modules, improves object-level classification accuracy and generalization through local and global feature modeling.

Meanwhile, clustering methods play an important auxiliary role in HSI classification. By generating consistent spatial object regions, clustering methods provide more robust input features for classification models, thereby indirectly improving classification performance. For instance, the AMKSC model (Cai et al., 2024) proposes an anchor-based multiview kernel subspace clustering method, which effectively integrates contributions from different modalities by learning scalable anchor graphs in the kernel space and introduces spatial regularization to improve clustering accuracy and efficiency. Moreover, the BGPC (Zhang, Jiang, Cai, & Zhou, 2024) method further introduces local region guidance and bipartite graph-based projection learning, effectively leveraging the spatial information in HSI. These clustering-based methods not only aim to enhance the spatial consistency and denoising ability of hyperspectral data but also establish semantically consistent spatial regions, thereby laying a solid foundation for classification tasks.

2.2. Graph U-Net

The U-Net architecture, originally developed for medical image segmentation (Ronneberger, Fischer, & Brox, 2015), features a symmetrical encoder–decoder structure. The encoder progressively reduces feature map dimensions through pooling, extracting contextual information. The decoder then restores spatial details using upsampling to reconstruct the segmentation result. Skip connections between the encoder and decoder transfer low-level features directly, minimizing spatial information loss. This design enables U-Net to capture both high-level context and low-level details, making it highly effective for image segmentation tasks.

Graph U-Net extends the U-Net architecture to graph data, maintaining the advantages of U-Net while enabling application to non-Euclidean spaces (Asif et al., 2021). A key difference in graph U-Net is the use of GCN instead of traditional CNN in both the encoder

Table 1
Key notational conventions.

Notations	Meaning
X	The input of HSI
Y	The output classification label
N	Superpixels numbers of $\mathbf{S}$
T	Trees of $\mathbf{S}$
F	Auxiliary forest of T
$\mathbf{S}$	Superpixel segmentation obtained after the encoder–decoder stage of U-Net
$\mathbf{G}$	Hierarchical graph structure topology corresponding to $\mathbf{S}$ of the encoder–decoder stage of U-Net
$\mathbf{S}^{(l)}$	Superpixel segmentation obtained after the lth encoder–decoder stage of U-Net
$\mathbf{G}^{(l)}$	Hierarchical graph structure topology corresponding to $\mathbf{S}^{(l)}$ of the encoder–decoder stage of U-Net
v'	Nodes of $\mathbf{S}$
v	Nodes of $\mathbf{G}$
c	Superpixels of $\mathbf{S}$
ε	Edges of $\mathbf{S}$ and $\mathbf{G}$
$\mathbf{A}^{(l)}$	Adjacency matrix of the lth layer
$\mathbf{W}^{(l)}$	Weight matrix of the lth layer
$\mathbf{H}^{(l)}$	Node matrix of the lth layer of the encoder
$\hat{\mathbf{H}}^{(l)}$	Node vector of the node in the lth layer of the decoder
SPA	The sparse self-attention module
FFN	The feedforward network module
$\boldsymbol{A}$	Attention maps within self-attention mechanisms
$\mathcal{A}'$	Attention maps within sparse self-attention mechanisms
$\mathbf{H}_a^{(l)}$	Attention results of the obtain
$\mathbf{H}^{(l)}_{SPA}$	Gated attention features output of the SPA module
$\mathbf{H}^{(l)}_{f1}$	Depthwise convolved features
$\mathbf{g}_{\mathbf{H}^{(l)}}$	Gating values
$\mathcal{H}_{\mathbf{g}}^{(l)}$	Gated result
δ	Layer normalization
$\mathcal{L}$	Loss function
LN(·)	Layer normalization
Re(·)	Process of reshaping and transposing the dimension
ReLU(·)	Rectified Linear Unit activation function

and decoder (Lin et al., 2022). Another distinction lies in the pooling operation. In traditional CNNs, pooling reduces the spatial dimensions of feature maps, improving computational efficiency and robustness to spatial transformations. However, applying this to graph data is challenging due to the lack of fixed spatial arrangements of nodes. This makes defining consistent pooling operations difficult. To address this, graph U-Net employs specialized pooling and unpooling operations tailored to the unique characteristics of graph data. HSI contain rich spatial information, where the interaction between spectral features and spatial relationships plays a crucial role in classification. By constructing graph structures to model spatial relationships between pixels (Liu et al., 2022), graph U-Net can more accurately capture these complex relationships, resulting in improved classification performance.

2.3. Vision transformer

Vision Transformers (ViT) (Han et al., 2022) are attention-based deep learning models that have gained significant popularity in image processing and computer vision. They leverage attention mechanisms to identify associations between different parts of an image, enabling them to capture long-range dependencies and achieve state-of-the-art performance across various tasks. ViT learns these associations dynamically, which is particularly useful for feature extraction. This capability is especially valuable for HSI classification, where both spatial patterns within individual spectral bands and spectral correlations across bands are critical. Using the self-attention mechanism (Vaswani et al., 2017), ViT adaptively focuses on significant areas, producing feature representations rich in semantic information and improving HSI classification performance.

Recently, ViT has shown great potential in HSI classification due to its ability to extract both spatial and spectral features effectively. For instance, the SSFTT model (Sun et al., 2022) extracts shallow spectral and spatial features and uses a transformer encoder for feature learning. Other models, such as SST (He, Chen, & Lin, 2021), integrate spectral attention mechanisms to capture continuous spectral information. UFMS-LN employ lightweight multi-head attention mechanisms to compress spatial features (Li, Xu, Liu, Sheng & Wan, 2024). However, these approaches often focus more on spatial relationships within individual spectral bands, rather than fully modeling broader spectral relationships across bands. In contrast, we propose a novel sparse transformer block that captures spectral relationships, leading to a more comprehensive understanding of HSI data.

3. The proposed methods

This section first introduces the incremental merging method used to construct hierarchical graph merge blocks. In Section 3.1.1, multilevel graphs are progressively built using adjacency and association matrices, as illustrated in Fig. 2. Next, Section 3.1.2 analyzes the core structure of HGSTNet's sparse transformer block. We then propose a novel sparse self-attention mechanism, shown in Fig. 3. Section 3.1.3 discusses the classification loss used in HGSTNet. Following this, Sections 3.2 and 3.3 provide a comprehensive analysis of all modules within HGSTNet, including their respective losses, and explain the overall optimization process of the model. Finally, Section 3.4 outlines the algorithmic procedure implemented by the HGSTNet model.

3.1. Overall architecture

Based on the classic U-Net architecture (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017) and the design principles of traditional vision transformer (Han et al., 2022), we propose a method for HSI classification called HGSTNet. Our entire network model, illustrated in Fig. 1, follows a like U-Net structure, comprising an encoder–decoder architecture. Initially, the input to the network is a HSI denoted as $\mathrm{X} \in \mathbb{R}^{H\times W\times C_{in}}$, with height H, width W, and C_{in} spectral bands. We construct multilevel graphs gradually based on superpixels to obtain hierarchical graph merge block and affinity matrices that play a certain role. We preprocess the HSI to compress it. At each stage of the encoder–decoder, we incorporate sparse transformer blocks to replace some GCN layers, effectively capturing global dependencies through SPA module and FFN module extracting feature information more meticulously. The encoder is responsible for extracting features from the input X, while introducing sparse transformer blocks to alter the feature scale for enlarging the receptive field. Symmetrically, the decoder utilizes features extracted from the encoder along with its own feature generation capabilities to gradually reconstruct features and restore images. Skip connections between the encoder and decoder ensure that feature information is preserved during the upsampling process. Finally, we employ the softmax activation function to assign a probability to each class for every pixel, yielding the final classification output $\mathrm{Y} \in \mathbb{R}^{H\times W\times C'}$ with numbers of channel C'.

3.1.1. Hierarchical graph merge block

To construct the hierarchical graph merge block at each stage (Stage i) of the merging process, we first preprocess the HSI using principal component analysis (PCA) to reduce dimensionality and minimize the impact of irrelevant information on subsequent segmentation. We then perform superpixel segmentation using a super-hierarchy approach. Next, we introduce a hierarchical graph incremental merging method. This method aggregates adjacent identical superpixels into new superpixels through region merging (Wei et al., 2018). The result is a

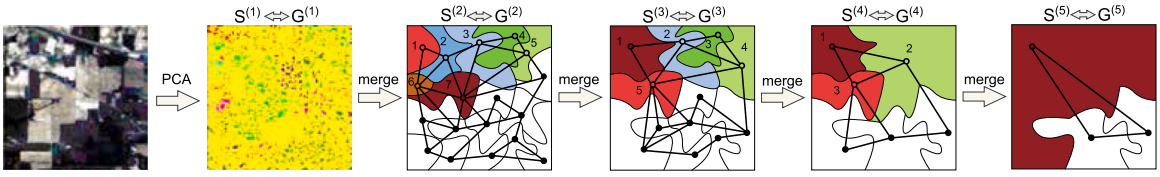


Fig. 2. Hierarchical graph structure fusion for multilevel feature learning in U-Net. This figure illustrates the process of constructing the hierarchical graph merge block by incrementally merging superpixels in U-Net across different stages, where $\mathbf{S}^{(l)}$ and $\mathbf{G}^{(l)}$ represent the superpixel segmentation and corresponding hierarchical graph structure at layer l, respectively. $\mathbf{S}^{(1)}$ represents the superpixel segmentation obtained after the first encoder–decoder stage, while $\mathbf{G}^{(l)}$ represents the corresponding hierarchical graph structure of $\mathbf{S}^{(1)}$. Subsequent stages generate progressively representations: $\mathbf{S}^{(2)}$ and $\mathbf{G}^{(2)}$ to correspond Stage 1, …, $\mathbf{S}^{(5)}$ and $\mathbf{G}^{(5)}$ to Stage 4. This demonstrates the iterative construction of the hierarchical graph structure. Each stage contributes a level to the hierarchical graph structure, capturing spatial details at different scales. Meanwhile, each superpixel is assigned a serial number, e.g., 1, 2, …, 7. Adjacent superpixels are merge to generate at superpixels at the next layer l +1, e.g., $\{1,2\} \rightarrow \{1\}, \{4,5\} \rightarrow \{4\}, \{6,7\} \rightarrow \{5\}$.

Algorithm 1 Hierarchical graph incremental merging method

Input:
(1) Original graph $\mathbf{G}(v, \varepsilon)$;
(2) Initial $\mathbf{S}(v' = v, \varepsilon = NULL)$;
(3) Set auxiliary forest F with n trees T, each initialized with nodes v from $\mathbf{G}$;
Output: $\mathbf{S}(v' = v, \varepsilon = \varepsilon')$.

```
Initialize the number of superpixels to N and minimum spanning tree (MST) = NULL;
while n>N do
  // Manhattan distance to calculate the weight of the e_{i,j} two adjacent nodes
  for v_{i,j} ∈ v do
    if v_i ≠ v_j then
      e_{i,j} = ||v_i − v_j||_1
    end if
  end for
  // Minimum spanning tree (MST) algorithm
  for T ⊆ F do
    if e_{i,j} < T_{e_{i,j}} then
      Update the minimum edge of T with e_{i,j} and add edge e_{i,j} to MST;
    end if
    for edge e_{i,j} ⊆ MST do
      Get the two nodes of the edge e_{i,j}; // node v_i and v_j are connected
    end for
  end for
end while
return S(v' = v, ε = ε').
```

hierarchical graph structure, where each stage (i) corresponds to a level in the hierarchy, with each level representing spatial details at different scales.

Specifically, given an undirected connected graph $\mathbf{G} = (v, \varepsilon)$ representing the HSI, where v is the set of nodes and ε is the set of edges ($\varepsilon \subseteq v \times v$), we first apply PCA to the HSI to obtain the three principal components as node features. A hierarchical graph structure is then constructed through the incremental merging method, progressively creating new superpixels and building a hierarchical graph with an arbitrary number of nodes. This process is detailed in the pseudocode of Algorithm 1 and visually depicted in Fig. 2, which illustrates the dynamic merging and step-by-step creation of the hierarchical graph.

Let $\left\{\mathbf{S}^{(l)}, \mathbf{G}^{(l)}\right\}_{l=1}^{L}$ represent the lth layer superpixel segmentation and the corresponding hierarchical graph structure topology respectively. The specific form is as follows:

$$\mathbf{S}^{(l)} = \left\{c_i^{(l)}\right\}_{i=1}^{N} \tag{1}$$

$$\mathbf{G}^{(l)} = \left\{v^{(l)}, \varepsilon^{(l)}\right\} = \left\{\left\{v_i^{(l)}\right\}_{i=1}^{n}, \left\{e_j^{(l)}\right\}_{j=1}^{m}\right\} \tag{2}$$

where $\mathbf{S}^{(l)}$ represents a set of N superpixels and $c_i^{(l)}$ represents ith superpixels of the lth layer superpixel segmentation, $\mathbf{G}^{(l)}$ represents the lth layer hierarchical graph structure topology composed of n nodes $v \in v$ and m edges $e \in \varepsilon \subseteq v \times v$. In particular, $v^{(l)}$ represents a node matrix $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times D}$ (D is the feature dimension), and $v^{(l)}$ represents the adjacency matrix $\mathbf{A}^{(l)} \in \mathbb{R}^{N \times N}$. To represent the connectivity of the powerless graph, we define its adjacency matrix as:

$$\mathbf{A}_{i,j}^{(l)} = \begin{cases} 1, & \text{if} T_{i,j}^{(l)} \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where the attention coefficient $T_{i,j}^{(l)}$ represents the node characteristics of each pair of connected nodes $v_i \in v^{(l)}$ and $v_j \in v^{(l)}$ of $\mathbf{A}^{(l)}$ on lth layer.

The update process of nodes in the encoder from layer l to layer $l+1$ can be written as:

$$\text{Encoder} \rightarrow \mathbf{H}_i^{(l+1)} = \text{ReLU}\left(\sum_{j \in v} T_{i,j}^{(l)} \mathbf{W}^{(l)} \mathbf{H}_j^{(l)}\right) \tag{4}$$

where $\mathbf{W}^{(l)}$ represents the weight matrix of layer l, and ReLU(·) represents the activation function.

Correspondingly, in the decoder, the conversion process of the graph feature from layer $l+1$ to layer l is similar to that of the encoder as follows:

$$\text{Decoder} \rightarrow \hat{\mathbf{H}}_i^{(l)} = \text{ReLU}\left(\sum_{j \in v} T_{i,j}^{(l)} \mathbf{W}^{(l)} \hat{\mathbf{H}}_j^{(l+1)}\right) \tag{5}$$

3.1.2. Sparse transformer block

Considering the spatial characteristics of HSI and the limitations of GCNs in capturing long-range dependencies and fully utilizing spatial and spectral information, we propose a novel sparse transformer block. This block addresses the locality constraint of GCNs by leveraging a sparse self-attention mechanism to compute global relationships between features. Specifically, it selectively attends to high-relevance features across non-adjacent regions of the spatial domain, effectively capturing long-range spatial dependencies. The sparse transformer block enhances spectral feature modeling by calculating inter-band correlations, allowing the model to capture complex global relationships within the spectral domain. By integrating both spatial and spectral information, the sparse transformer block refines feature representations, improving classification performance. The overall architecture of our proposed sparse transformer block is described in Algorithm 2.

Specifically, the entire sparse transformer block comprises two modules: SPA and FFN. The SPA module, as illustrated in Fig. 1, consists of sparse self-attention mechanism combined with a residual connections (Szegedy et al., 2017), gating mechanism (Yu et al., 2019), and layer normalization (LN). First, for SPA in each sparse transformer block, we input the features $\mathbf{H}^{(l-1)}$ into the SPA of multiple heads to quickly obtain the attention results $\mathbf{H}_a^{(l)}$, which can be written as:

$$\mathbf{H}_a^{(l)} = \text{SPA}\left(\text{LN}(\mathbf{H}^{(l-1)})\right) \tag{6}$$

where LN(·) denotes layer normalization applied to the input features $\mathbf{H}^{(l-1)}$. The gating mechanism uses a spatial convolutional network to

Algorithm 2 Sparse transformer block

Input:
(1) The input is $\mathbf{H}^{(l-1)} \in \mathbb{R}^{N\times D}$;
(2) The number of attention head is $h \in [1,4]$;
Output: $\mathbf{H}^{(l)}$. // Output feature map of the l-th layer.
for $l \in h$ **do**
  // SPA module
  Obtain the attention result $\mathbf{H}_a^{(l)}$ in Eq. (6);
  Guide spatial information and obtain gating values $\mathbf{g}_{\mathbf{H}^{(l)}}$ for each spatial position in Eq. (7);
  Perform the hadamard product to get the feature attention $\mathbf{H}_{\mathbf{g}}^{(l)}$ in Eq. (8);
  Use a residual shortcut to connect the input feature $\mathbf{H}^{(l-1)}$ and the gated attention feature $\mathbf{H}_{\mathbf{g}}^{(l)}$ to gain the output feature $\mathbf{H}_{SPA}^{(l)}$ in Eq. (9);
end for
// FFN module
For step 6, use two 3×3 depthwise convolutions: one to obtain the gating values $\mathbf{g}_{\mathbf{H}_{SPA}^{(l)}}$ and the other to obtain feature $\mathbf{H}_{f1}^{(l)}$ in Eq. (10), Eq. (11);
Compute the gated result $\mathcal{H}_{\mathbf{g}}^{(l)}$ in Eq. (12);
Use a residual connection to link the input feature $\mathbf{H}_{SPA}^{(l)}$ with the convoluted gated feature $\mathrm{Conv}(\mathcal{H}_{\mathbf{g}}^{(l)} \odot \mathbf{g}_{\mathbf{H}_{SPA}^{(l)}})$, outputting the final feature $\mathbf{H}^{(l)}$ in Eq. (13);
return $\mathbf{H}^{(l)}$.

generate gating values that modulate the attention results, allowing the model to focus on relevant spatial regions and suppress irrelevant information. Subsequently, spatial gating mechanisms are employed to guide the attention results using spatial information. This is followed by applying 1×1 convolution and activation function to obtain the gating values $\mathbf{g}_{\mathbf{H}^{(l)}}$ for each spatial position. We formulate this process as:

$$\mathbf{g}_{\mathbf{H}^{(l)}} = \delta\left(\mathrm{Conv}\left(\mathrm{LN}\left(\mathbf{H}^{(l-1)}\right)\right)\right) \tag{7}$$

where $\mathrm{Conv}(\cdot)$ denotes a 1×1 convolution operation, and $\delta(\cdot)$ represents an activation function. Residual connections enable the gradient to flow more easily through the network, preventing the vanishing gradient problem and improving training stability. Subsequently, the attention results are modulated by performing the hadamard product between the gating values and the attention results, obtaining the gated attention features, which can be expressed as:

$$\mathbf{H}_{\mathbf{g}}^{(l)} = \mathbf{H}_a^{(l)} \odot \mathbf{g}_{\mathbf{H}^{(l)}} \tag{8}$$

where $\odot$ denotes the hadamard product operation. Finally, we employ a residual connection between the input features $\mathbf{H}^{(l-1)}$ and the gated attention features $\mathbf{H}_{\mathbf{g}}^{(l)}$, resulting in the output of the SPA module. This entire process can be expressed as:

$$\mathbf{H}_{SPA}^{(l)} = \mathbf{H}^{(l-1)} + \mathbf{H}_{\mathbf{g}}^{(l)} \tag{9}$$

The FFN module integrates layer normalization (LN), residual connections (Szegedy et al., 2017), and a gating mechanism (Yu et al., 2019), as depicted in Fig. 1. The FFN module consists of layer normalization, a gating mechanism, residual connections, and two convolutional layers: one for channel expansion and one for feature reduction. We further process the output $\mathbf{H}_{SPA}^{(l)}$ from the SPA module through the FFN module. Initially, the normalized input features $\mathbf{H}_{SPA}^{(l)}$ undergo channel expansion via two 1×1 convolutions, followed by two 3×3 depthwise convolutions (Howard et al., 2019). Depthwise convolutions operate on each input channel independently, performing convolutions along the spatial dimension only. One of these depthwise convolutions passes through an activation function to obtain gate values $\mathbf{g}_{\mathbf{H}_{SPA}^{(l)}}$, while the other directly yields the depthwise-convolved features $\mathbf{H}_{f1}^{(l)}$. This can be expressed as:

$$\mathbf{g}_{\mathbf{H}_{SPA}^{(l)}} = \delta\left(\mathrm{Convs}\left(LN\left(\mathbf{H}_{SPA}^{(l)}\right)\right)\right) \tag{10}$$

$$\mathbf{H}_{f1}^{(l)} = \mathrm{Convs}\left(\mathrm{LN}\left(\mathbf{H}_{SPA}^{(l)}\right)\right) \tag{11}$$

where $\mathrm{Convs}(\cdot)$ denotes the convolution operation. The gating mechanism in the FFN module uses a spatial convolutional network to generate gating values that modulate the feature maps based on spatial information, ensuring that relevant features are emphasized and irrelevant features are suppressed. Subsequently, we perform the hadamard product on the two obtained values to derive the gated result $\mathcal{H}_{\mathbf{g}}^{(l)}$. The process can be described by the following equation:

$$\mathcal{H}_{\mathbf{g}}^{(l)} = \mathbf{H}_{f1}^{(l)} \odot \mathbf{g}_{\mathbf{H}_{SPA}^{(l)}} \tag{12}$$

Finally, we perform a 1×1 convolution again to reduce the feature channels. Then, a residual connection is applied to link the input features $\mathbf{H}_{SPA}^{(l)}$ with the gated features, yielding the final output features $\mathbf{H}^{(l)}$. The entire process can be simplified by the following formula:

$$\mathbf{H}^{(l)} = \mathbf{H}_{SPA}^{(l)} + \mathrm{Conv}(\mathcal{H}_{\mathbf{g}}^{(l)} \odot \mathbf{g}_{\mathbf{H}_{SPA}^{(l)}}) \tag{13}$$

Sparse self-attention mechanism. The conventional self-attention mechanism (Vaswani et al., 2017) calculates relationships between queries and key–value pairs, capturing interdependencies within an input sequence. It transforms the query $\boldsymbol{Q}$ and key $\boldsymbol{K}$ to obtain an attention map, representing the strength of correlations between all input elements. The self-attention mechanism then performs a weighted summation of the value $\boldsymbol{V}$ based on the attention map, yielding the attention output. As depicted in Fig. 3(a), this process can be formulated as:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{Q}, \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{K}, \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{V} \tag{14}$$

$$\mathrm{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \mathrm{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{T}}{\sqrt{\boldsymbol{d_k}}})\boldsymbol{V} \tag{15}$$

where $\boldsymbol{Q} \in \mathbb{R}^{N\times D}$, $\boldsymbol{K} \in \mathbb{R}^{N\times D}$, and $\boldsymbol{V} \in \mathbb{R}^{N\times D}$ are derived from the input feature map $\mathbf{H}^{(l)}$, with $N = H \times W$ representing the length/resolution of the input features. H, W, and C denote the height, width, and number of channels of the input feature map, respectively. D signifies the dimensionality of the embedded features, typically equal to the dimensions of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$, usually $N \gg D$. $\boldsymbol{W}^{Q}, \boldsymbol{W}^{K}, \boldsymbol{W}^{V} \in \mathbb{R}^{C\times D}$ represent learnable linear transformation matrices obtained through linear transformations of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$. The scaling parameter $\sqrt{d_k}$ normalizes the dot product between $\boldsymbol{Q}$ and $\boldsymbol{K}$, obtain attention map $\boldsymbol{A}$, it is expressed as follows:

$$\boldsymbol{A} = \mathrm{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{T}}{\sqrt{\boldsymbol{d_k}}}) \tag{16}$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and its computational complexity $\mathcal{O}(N^2 D)$, is shown in Fig. 3(a). Finally, through a softmax operation and subsequently multiplied attention map $\boldsymbol{A}$ with the value $\boldsymbol{V}$ to yield the final attention output.

However, self-attention mechanism face challenges due to their quadratic complexity and global focus, which limit real-time processing and local context understanding. To overcome these issues, we propose sparse self-attention mechanism, which applies the ReLU activation (Glorot, Bordes, & Bengio, 2011) function to filter out negative values, inducing sparsity. This approach significantly reduces computational complexity and memory consumption, making the model more efficient for real-time applications.

Specifically, the input feature map is represented as $\mathbf{H}^{(l)} \in \mathbb{R}^{B\times H\times W\times C}$, where B represents the batch size, C is the number of channels, H and W correspond to the height and width of the feature map, respectively. Similar to the self-attention mechanism, linear transformations are applied using three linear transformation $\boldsymbol{W}^{Q}, \boldsymbol{W}^{K}, \boldsymbol{W}^{V}$ to obtain $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$. Subsequently, the shapes of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are transformed. These

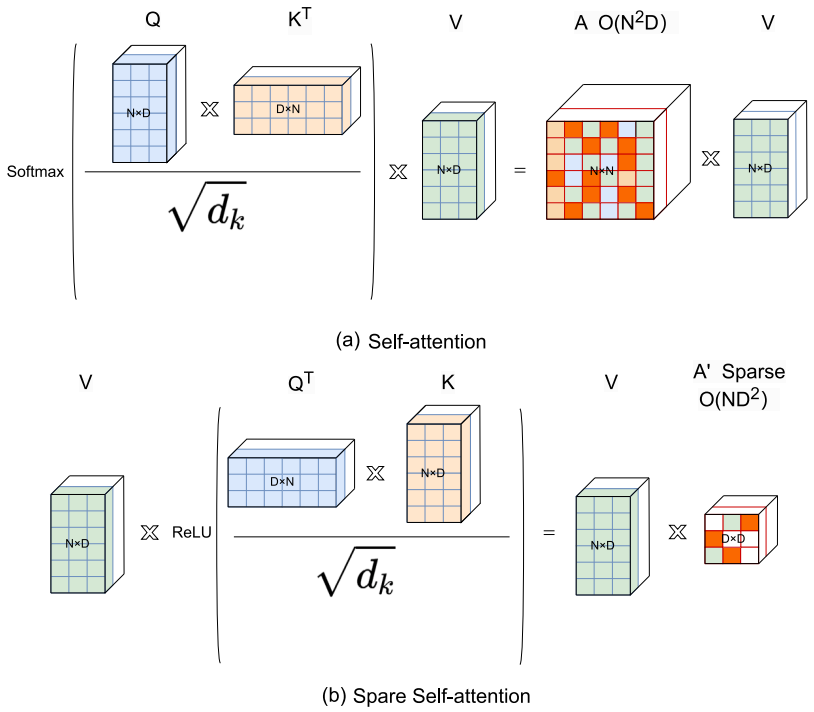


Fig. 3. Architectural overview: self-attention and sparse self-attention.

matrices are then reshaped and transposed to align their dimensions for the dot product calculation. The process can be formulated as follows:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{Q}, \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{K}, \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{V} \tag{17}$$

Then, $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are transformed, with the specific formulas as follows:

$$\begin{aligned} \boldsymbol{Q} &= \mathrm{Re}(\boldsymbol{Q}, \boldsymbol{B}\cdot\boldsymbol{C}\cdot HW \rightarrow \boldsymbol{B}\cdot head\cdot\boldsymbol{c}\cdot HW) \\ \boldsymbol{K} &= \mathrm{Re}(\boldsymbol{K}, \boldsymbol{B}\cdot\boldsymbol{C}\cdot HW \rightarrow \boldsymbol{B}\cdot head\cdot\boldsymbol{c}\cdot HW) \\ \boldsymbol{V} &= \mathrm{Re}(\boldsymbol{V}, \boldsymbol{B}\cdot\boldsymbol{C}\cdot HW \rightarrow \boldsymbol{B}\cdot head\cdot\boldsymbol{c}\cdot HW) \end{aligned} \tag{18}$$

where $\mathrm{Re}(\cdot)$ signifies the process of reshaping and transposing the dimensions of $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$. $\boldsymbol{B}$ represents the batch size, H and W denote the height and width of the feature map. The total number of channels in the feature map is represented by $\boldsymbol{C}$, and the number of attention heads is indicated by $head$. Each attention head processes $\boldsymbol{c} = \boldsymbol{C}/head$ channels, making the total dimensionality of the embedded features $D = \boldsymbol{c}\times head$. Notably, D also represents the combined dimensionality of all attention heads. Typically, D is equal to or close to $\boldsymbol{C}$, but this is a design choice and can be set such that $D \neq \boldsymbol{C}$. This relationship ensures that $\boldsymbol{c}$ not only describes the allocation of channels per head but also contributes to determining the total embedding dimensionality D. We then perform a dot product operation on the dot product of query $\boldsymbol{Q}$ and key $\boldsymbol{K}$, resulting in an attention map $\mathcal{A}$. Each element $\mathcal{A}_{i,j}$ in the map represents the correlation between the vector $\boldsymbol{Q}_i^T \in \mathbb{R}^{1\times N}$ of the ith channel and the vector $\boldsymbol{K}_j \in \mathbb{R}^{N\times 1}$ of the jth channel. It can be formulated as:

$$\mathcal{A} = \mathrm{Softmax}\left(\frac{\boldsymbol{Q}^T\boldsymbol{K}}{\sqrt{\boldsymbol{d_k}}}\right) \tag{19}$$

where $\boldsymbol{Q}^T \in \mathbb{R}^{D\times N}$, $\mathcal{A} \in \mathbb{R}^{D\times D}$. Subsequently, we resparse the attention map $\mathcal{A}$ by employing the ReLU activation function (Glorot et al., 2011), filtering out positive correlations and generating the sparse attention map $\mathcal{A}' = \mathrm{ReLU}(\frac{\boldsymbol{Q}^T\boldsymbol{K}}{\sqrt{\boldsymbol{d_k}}})$, whose computational complexity is $\mathcal{O}(ND^2)$. Last, it is multiplied with $\boldsymbol{V}$ to obtain the weighted sum output. As depicted in Fig. 3(b), the formula is expressed as follows:

$$\mathrm{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{V}\mathcal{A}' \tag{20}$$

3.1.3. Classification loss

During the training phase, we utilize a cross-entropy loss function to train our network model. However, in practical training datasets, an imbalance often exists in the number of samples belonging to different classes. This class imbalance can lead to the model overemphasizing the majority class samples while neglecting the minority class samples. To address this issue, we define the weight for each class as the normalized reciprocal of its sample count, ensuring a balanced contribution from each class. The calculation of the class weight and the classification loss is as follows:

$$W_c = \frac{1/n_c}{\sum_{k=1}^{C_t}(1/n_k)} \tag{21}$$

$$\mathcal{L}_{\mathrm{cls}} = -\frac{1}{N_t}\sum_{1}^{N_t}\sum_{1}^{C_t} W_c L_c^p \log(\hat{L}_c^p) m_c \tag{22}$$

where W_c represents the normalized weight assigned to each class c, n_c is the number of training samples for class c. C_t is the total number of classes. L_c^p and $\hat{L}_c^p$ are scalar values that represent the predicted probability and true probability, respectively, for a single pixel belonging to class c. N_t denotes the total number of samples, while m_c corresponds to the mask value for each class c (taking a value of 1 for valid pixels and 0 otherwise).

3.2. Loss function

Our overall HGSTNet loss consists of four components: hierarchical graph merge block loss $\mathcal{L}_{\mathrm{hg}}$, sparse transformer block loss $\mathcal{L}_{\mathrm{st}}$ and classification loss $\mathcal{L}_{\mathrm{cls}}$, The total loss $\mathcal{L}_{\mathrm{HGSTNet}}$ can be expressed as follows:

$$\mathcal{L}_{\mathrm{HGSTNet}} = \mathcal{L}_{\mathrm{hg}} + \mathcal{L}_{\mathrm{st}} + \mathcal{L}_{\mathrm{cls}} \tag{23}$$

The first part involves computing the hierarchical graph merge block loss by applying smoothness regularization on the adjacency matrix, enabling HGSTNet to learn a smoother hierarchical graph structure. The specific hierarchical graph merge block loss can be formulated as follows:

$$\mathcal{L}_{\mathrm{hg}} = \sum_{i=1}^{L}\sum_{j=1}^{n}\sum_{k=1}^{n}\left|A_{i,j,k} - \frac{1}{n}\sum_{l=1}^{n}A_{i,j,l}\right| \tag{24}$$

where $A_{i,j,l}$ represents the weight between node i and node j in the kth layer of the graph. L is the number of layers in the graph, and n is the number of nodes in each graph.

The second part, the loss of our sparse transformer block, can be expressed as follows:

$$\mathcal{L}_{\text{st}} = \mathcal{L}_{\text{SPA}} + \mathcal{L}_{\text{FFN}} \tag{25}$$

The sparse transformer block primarily calculates losses from the SPA module and FFN module. These losses are calculated as follows:

$$\mathcal{L}_{\text{SPA}} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(A_{ij} - A_{ij}^{*})^2 \tag{26}$$

$$\mathcal{L}_{\text{FFN}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^{*})^2 \tag{27}$$

where A_{ij} represents the attention weight matrix output by the SPA, with dimensions $N \times N$. A_{ij}^{*} represents the target attention weight matrix, with dimensions $N \times N$. y_i represents the output after the FFN; y_i^{*} represents the target output.

The final part, we employ a cross-entropy loss function for training our network model, utilizing back propagation and an adaptive moment estimation gradient descent algorithm to update the network weights and biases. The specific formula is given as Eq. (21).

3.3. Optimization process

The overall loss function of the HGSNet model is shown in Eq. (22). Now, we are going to optimize it using a multi-task learning strategy. The overall objective function to be optimized is defined as follows:

$$\mathcal{L} = \lambda_{hg}\mathcal{L}_{\text{hg}} + \lambda_{st}\mathcal{L}_{\text{st}} + \lambda_{cls}\mathcal{L}_{\text{cls}} \tag{28}$$

After our experimental testing, the optimal weights settings were found to be: $\lambda_{hg}, \lambda_{st}, (\lambda_{cls}) = (0.1, 0.1, 1)$. This setting prioritizes the U-net loss while balancing the contributions of the graph-based, transformer-based, and regularization losses.

To optimize the HGSNet model and achieve robust classification performance, we adopt a staged training strategy, inspired by Cycada and CMFT, CMFF (Hoffman et al., 2018; Zhang et al., 2021). This strategy consists of three stages: (1) Backbone pre-training: training is initiated by pre-training the U-Net on a classification task. For the classification loss in Eq. (21), we employ the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 5×10^{-4} and a weight decay of 1×10^{-3}. A warm-up strategy (He et al., 2016) is incorporated in the initial stage to avoid performance instability. Furthermore, a cosine annealing (Loshchilov, 2017) and dynamic learning rate scheduling scheme are adopted to enhance the learning process; (2) Hierarchical graph learning: we integrate the hierarchical graph merge block by optimizing the hierarchical graph merge block loss in Eq. (23). Simultaneously, we apply Xavier initialization to the hierarchical graph merge block to accelerate model convergence. This can be mathematically represented as:

$$W^{(l)} \sim U^{(l)}\left[-\sqrt{\frac{6}{H^{(l)} + H^{(l+1)}}}, \sqrt{\frac{6}{H^{(l)} + H^{(l+1)}}}\right] \tag{29}$$

where $U^{(l)}$ denotes the uniform distribution probability in the lth layer; (3) Sparse transformer adaptation: for Eq. (23), we independently train the sparse transformer block using the Adam optimizer with a learning rate of 5×10^{-4}, a weight decay of 1×10^{-3}, a batch size of 16, and for a total of 600 epochs.

3.4. Algorithm description

In this section, we present the complete workflow of the proposed HGSTNet method, as illustrated in Algorithm 3.

Specifically, first initialize the HGSTNet model with the specified number of convolutional layers l and set the Adam optimizer learning rate to 5×10^{-4}. Then, input the initial HSI data $\mathrm{X} \in \mathbb{R}^{H \times W \times C}$ and perform dimensionality reduction using the PCA method, transforming it into an image X_{pca}. Next, apply Algorithm 1 to construct hierarchical graph merge blocks at different scales. This process generates the superpixel segmentation map $\mathbf{S}^{(l)}$ and its corresponding hierarchical graph structure topology $\mathbf{G}^{(l)}$ as shown in Fig. 2. The encoder initiates by applying a convolution to the input HSI, generating the first-level feature map. Multilevel graph convolution, exploiting the hierarchical graph merge blocks, then extracts features across diverse spatial scales. This yields the subsequent-level nodes for each encoder stage represented as superpixels and their corresponding feature maps (superpixel maps). These features are further refined by employing Algorithm 2, which produces the subsequent-level feature map $\mathbf{H}^{(l)}$ while progressively downscaling the feature map size.

Correspondingly, in the decoder stage, progressively enlarge the feature map size in the decoder stage, combining the feature information from the encoding stage. This process finally generates the superpixel segmentation map $\mathbf{S}^{(l)}$ and the corresponding hierarchical graph structure topology $\mathbf{G}^{(l)}$ that match the input image size. The final step involves loading the best pre-trained model and obtaining the predicted labels Y for each layer feature map $\mathbf{H}^{(l)}$ through a softmax function, we have $\mathrm{Y} = \mathrm{Softmax}(\mathbf{H}^{(l)})$.

Algorithm 3 HGSTNet Model

Input:
(1) Input original HSI data $\mathrm{X} \in \mathbb{R}^{H \times W \times C_{in}}$;
(2) The number of PCA bands b;
(3) Learning rate lr;
Output: Predicted labels $\mathrm{Y} \in \mathbb{R}^{H \times W \times C'}$;

```
Initialize the number of convolutional layers l ∈ [1, 5];
Set optimizer Adam (learning rate = 5e-4), max epoch E to 600;
Obtain the X_pca after PCA transform;
Superpixels are merged using the MST algorithm by updating trees T in an auxiliary forest F and get superpixels S^(l) and their corresponding hierarchical graph structure topology G^(l);
for max epoch = 1 to E do
    Initialize feature graph H^(0);
    for convolution layer i to l do
        Use graph convolution in Eq. (4) to obtain the feature graph H^(l−1);
        Exploit sparse transformer block to get output feature graph H^(l);
        Execute graph convolution in Eq. (5) to derive the feature graph H^(l);
        Output and recover S^(l) and G^(l);
    end for
    Employ the softmax function to obtain the predicted labels Y =Softmax(H^(l));
    Calculate the loss and update the model parameters in Eq. (21);
end for
return Predicted labels Y ∈ R^{H×W×C'};
```

4. Experimental verification

This section evaluates the performance of the HGSTNet model and compares it with other state-of-the-art deep learning methods for HSI classification. Experiments were conducted on three publicly available HSI datasets. To assess the classification performance on these HSI datasets, we employed three widely used metrics: OA, AA and KPP.

4.1. Experimental settings

In this section, we explain the dataset and network configuration to prepare for the subsequent experiments.

Table 2
Relevant parameters used in this article.

Dataset	Image size	Band	Class	Total	Train./Val./Test.
IP	145 × 145	200	16	10366	5%/ 1%/ 94%
PU	610 × 310	103	9	42776	1%/ 1%/ 98%
SA	512 × 217	204	16	54129	1%/ 1%/ 98%
XZ	500 × 260	436	9	68877	5%/ 1%/ 94%
HS	349 × 1905	144	15	15029	5%/ 1%/ 94%

Table 3
Category information of IP.

No.	Class	Train. /Val. /Test.	Total.
1	Alfalfa	3 /1 /50	54
2	Corn-notill	72 /15 /1347	1434
3	Corn-mintill	42 /9 /783	834
4	Corn	12 /3 /219	234
5	Pasture	25 /5 /467	497
6	Trees/Grass	38 /8 /701	747
7	Pasture-mowed	2 /1 /23	26
8	Hay-windrowed	25 /5 /459	489
9	Oats	1 /1 /18	20
10	Soybeans-no till	49 /10 /909	968
11	Soybeans-min till	124 /25 /2319	2468
12	Soybeans-clean till	31 /7 /576	614
13	Wheat	11 /3 /198	212
14	Woods	65 /13 /1216	1294
15	Building-Grass	19 /4 /357	380
16	Stone-steel	5 /1 /89	95
	Total	524 /111 /9731	10366

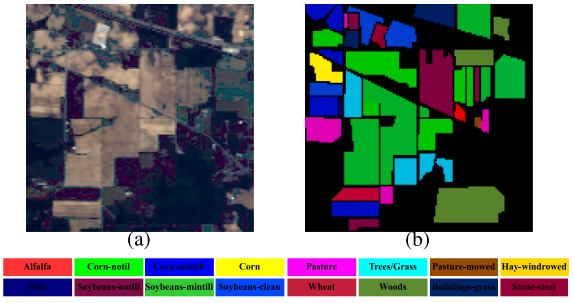


Fig. 4. (a) False-color image of the IP dataset. (b) Ground truth (GT) map of IP.

4.1.1. Datasets

We evaluated the performance of our proposed method on three publicly available HSI datasets: Indian Pines (IP), University of Pavia (PU), Salinas (SA), Xuzhou (XZ) and Houston (HS). Table 2 summarizes the relevant parameters, including the allocation of training, validation, and labeled samples for each dataset.

The IP dataset is a widely used HSI dataset acquired by the AVIRIS sensor in northwestern Indiana. It features a 145 × 145 pixel image with 200 spectral bands, providing rich spectral information for classification tasks. The dataset comprises 10,366 labeled samples representing 16 different land cover classes. The distribution of samples for training, validation, and testing is presented in Table 3 (see Fig. 4).

The PU dataset, acquired by the ROSIS sensor during a flight campaign over Pavia, northern Italy, was used in our experiments. It comprises a 610 × 310 pixel image with 103 spectral bands, featuring 42,776 labeled samples categorized into 9 distinct land cover classes. Table 4 provides a detailed breakdown of the samples used for training, validation, and testing (see Fig. 5).

The SA dataset, captured by the AVIRIS sensor over Salinas Valley, California, is a widely-used HSI dataset for agricultural applications. It features a 512 × 217 pixel image with 204 spectral bands, providing rich spectral information about various crops. The dataset includes 54,129 labeled samples representing 16 different crop classes. The specific distribution of training, validation, and test samples for each class is detailed in Table 5 (see Fig. 6).

The XZ dataset is a newly incorporated HSI dataset captured by a hyperspectral sensor over Xuzhou, China. It features a 500 × 260 pixel image with 436 spectral bands, offering detailed spectral information for land-cover classification. The dataset comprises 68877 labeled samples, categorized into 9 distinct land-cover classes, representing a mix of urban, vegetation, and bare-soil regions. The specific distribution of training, validation, and test samples for each class is presented in Table 6 (see Fig. 7).

Table 4
Category information of PU.

No.	Class	Train. /Val. /Test.	Total.
1	Asphalt	67 /67 /6497	6631
2	Meadows	187 /187 /18276	18649
3	Gravel	21 /21 /2057	2099
4	Trees	31 /31 /3002	3064
5	Metal sheets	14 /14 /1317	1345
6	Bare soil	51 /51 /4927	5029
7	Bitumen	14 /14 /1302	1330
8	Bricks	37 /37 /50	3608
9	Shadows	10 /10 /927	947
	Total	432 /432 /9731	42776

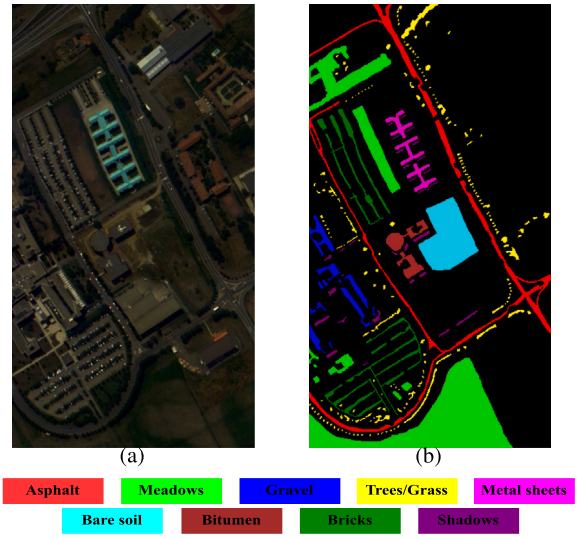


Fig. 5. (a) False-color image of the PU dataset. (b) GT map of PU.

Table 5
Category information of SA.

No.	Class	Train. /Val. /Test.	Total.
1	Brocoli_green_weeds_1	21 /21 /1967	2009
2	Brocoli_green_weeds_2	38 /38 /3650	3726
3	Fallow	20 /20 /1936	1976
4	Fallow_rough_plow	14 /14 /1366	1394
5	Fallow_smooth	27 /27 /2624	2678
6	Stubble	40 /40 /3879	3959
7	Celery	36 /36 /3507	3579
8	Grapes_untrained	113 /113 /11045	11271
9	Soil_vinyard_develop	63 /63 /6077	6203
10	Corn_senesced_green_weeds	33 /33 /3212	3278
11	Lettuce_romaine_4wk	11 /11 /1046	1068
12	Lettuce_romaine_5wk	20 /20 /1887	1927
13	Lettuce_romaine_6wk	10 /10 /896	916
14	Lettuce_romaine_7wk	11 /11 /1058	1070
15	Vinyard_untrained	73 /73 /7122	7268
16	Vinyard_vertical_trellis	19 /19 /1769	1807
	Total	549 /549 /53031	54129

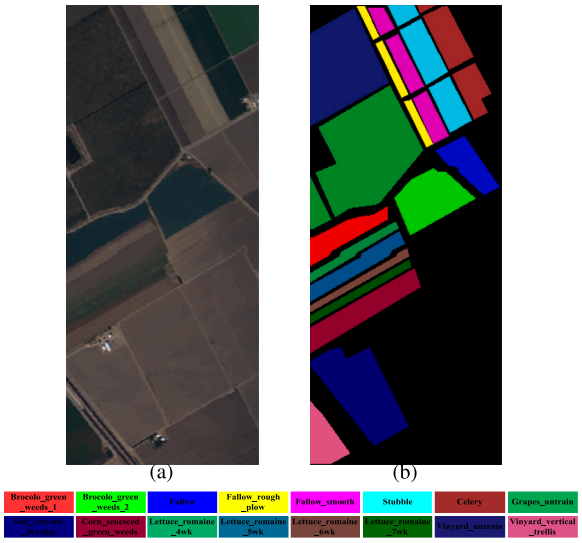


Fig. 6. (a) False-color image of the SA dataset. (b) GT map of SA.

Table 6
Category information of XZ.

No.	Class	Train. /Val. /Test.	Total.
1	Bareland-1	132 /132 /26132	26396
2	Lakes	21 /21 /3985	4027
3	Coals	14 /14 /2755	2783
4	Cement	27 /27 /5160	5214
5	Crops-1	66 /66 /13052	13184
6	Tress	13 /13 /2410	2436
7	Bareland-2	35 /35 /6920	6990
8	Crops-2	24 /24 /4729	4777
9	Red-tiles	16 /16 /3038	3070
	Total	348 /348 /68181	68877

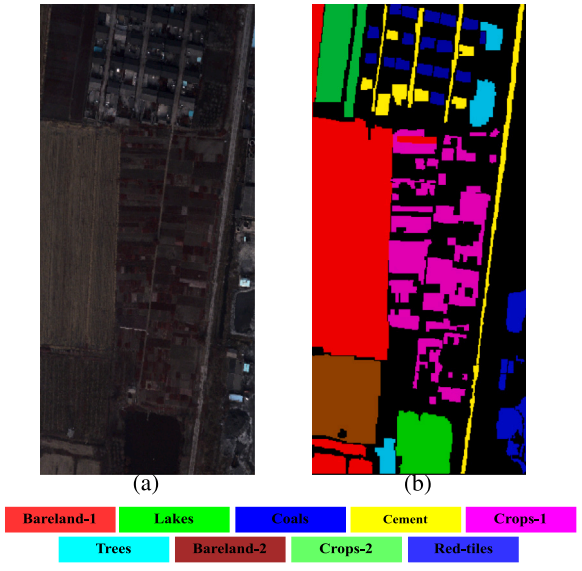


Fig. 7. (a) False-color image of the XZ dataset. (b) GT map of XZ.

The HS dataset, captured by an ITRES CASI-1500 sensor over the University of houston campus and its surrounding urban area, is a benchmark HSI dataset frequently utilized for urban land-cover classification tasks. It consists of a 349 × 1905 pixel image with 144 spectral bands, providing detailed spectral–spatial information for a variety of classification challenges. The dataset includes 15 distinct land-cover classes, with a total of 15,029 labeled samples representing diverse urban and natural categories such as vegetation, water bodies, road networks, and residential areas. The specific allocation of training, validation, and testing samples for each class is provided in Table 7 (see Fig. 8).

Table 7
Category information of HS.

No.	Class	Train. /Val. /Test.	Total.
1	Healthy Grass	63 /63 /1125	1251
2	Stressed Grass	63 /63 /1128	1254
3	Synthetic Grass	35 /35 /627	697
4	Tree	63 /63 /1118	1244
5	Soil	63 /63 /1116	1242
6	Water	17 /17 /291	325
7	Residential	64 /64 /1140	1268
8	Commercial	63 /63 /1118	1244
9	Road	63 /63 /1126	1252
10	Highway	62 /62 /1103	1227
11	Railway	62 /62 /1111	1235
12	Parking Lot 1	62 /62 /1109	1233
13	Parking Lot 2	24 /24 /421	469
14	Tennis Track	22 /22 /384	428
15	Running Track	33 /33 /594	660
	Total	348 /348 /68181	15029

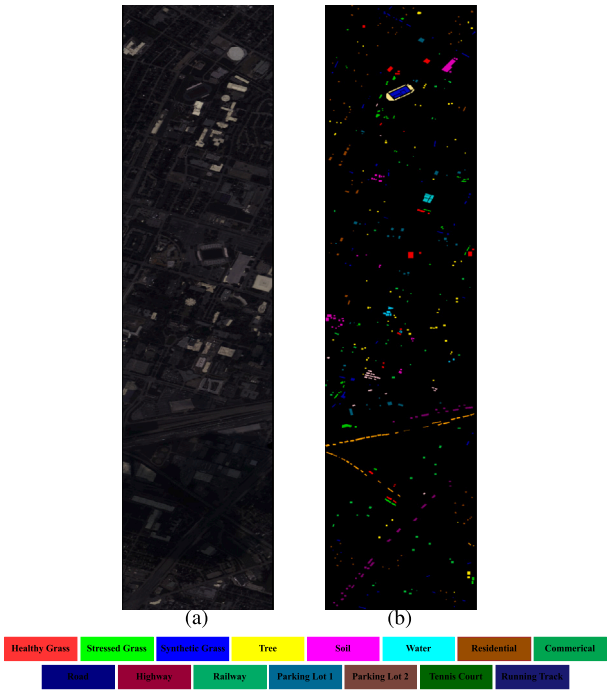


Fig. 8. (a) False-color image of the HS dataset. (b) GT map of HS.

4.1.2. Network configuration

We implemented the HGSTNet model using a python 3.7 environment with pytorch 1.7.0 and cuda toolkit 11.3. Experiments were conducted on a 64-bit windows operating system equipped with a 20-GB RTX3080 GPU, a 10-core intel xeon gold 6148 cpu, and 60 GB of memory. All comparative model experiments were tested under the same conditions. The HGSTNet model was trained using the adam optimizer with a learning rate of 0.0001 and a maximum of 600 training iterations. We adopted a five-stage multilevel encoder–decoder structure to demonstrate the effectiveness of the proposed approach even with deeper architectures.

Table 8
Classification results for the IP dataset (5% training samples)

Class	DBDA TGRS 2020 Li et al. (2020)	CEGCN TGRS 2021 Liu, Xiao, Yang, and Wei (2021)	MSSGU TGRS 2022 Liu et al. (2022)	A-HySN IJRS 2023 Fan et al. (2023)	UFMS-LN TGRS 2024 Li, Xu et al. (2024)	DATN EAAI 2024 Shu, Wang, and Yu (2024)	LMFFBNet ESWA 2024 Shi et al. (2024)	HGSTNet
1	99.53 ± 1.39	73.83 ± 15.69	92.40 ± 7.36	94.70 ± 4.00	97.63 ± 2.25	**100.00 ± 0.00**	93.90 ± 1.53	81.00 ± 14.12
2	93.63 ± 5.23	96.21 ± 1.35	**98.63 ± 0.64**	97.20 ± 0.64	87.81 ± 7.10	88.70 ± 6.30	97.04 ± 0.78	97.23 ± 0.89
3	94.36 ± 8.97	95.13 ± 2.26	**98.28 ± 1.57**	97.18 ± 2.53	91.46 ± 1.54	93.84 ± 4.43	96.26 ± 1.39	98.17 ± 1.07
4	96.41 ± 5.34	95.65 ± 2.62	**96.75 ± 2.46**	95.32 ± 5.24	89.20 ± 5.25	81.32 ± 10.34	95.16 ± 4.34	95.47 ± 6.30
5	96.94 ± 1.10	95.63 ± 2.58	96.63 ± 1.99	93.61 ± 5.71	84.94 ± 1.76	96.13 ± 2.56	**98.93 ± 0.22**	98.25 ± 2.88
6	96.57 ± 2.31	99.35 ± 0.40	98.92 ± 1.32	93.33 ± 6.13	97.91 ± 1.41	90.14 ± 7.34	98.92 ± 0.17	**99.57 ± 1.00**
7	89.60 ± 16.36	85.25 ± 18.30	91.73 ± 11.41	92.34 ± 1.84	**100.00 ± 0.00**	65.49 ± 29.57	55.89 ± 42.50	78.69 ± 23.53
8	98.70 ± 2.24	99.89 ± 0.20	**100.00 ± 0.00**	99.00 ± 1.40	99.43 ± 1.41	95.82 ± 3.34	**100.00 ± 0.00**	**100.00 ± 0.00**
9	64.84 ± 43.50	28.97 ± 22.14	70.00 ± 23.85	85.16 ± 0.92	**100.00 ± 0.00**	62.11 ± 10.21	68.05 ± 29.77	72.22 ± 25.34
10	89.94 ± 10.55	96.22 ± 2.45	97.07 ± 3.10	96.91 ± 2.87	85.80 ± 4.35	91.03 ± 4.29	97.47 ± 1.47	**97.69 ± 1.68**
11	94.83 ± 11.08	98.03 ± 1.23	98.64 ± 1.99	97.83 ± 7.04	93.89 ± 2.87	93.81 ± 3.34	96.95 ± 1.15	**98.87 ± 1.48**
12	97.26 ± 1.62	**98.00 ± 1.08**	96.61 ± 0.40	96.55 ± 2.38	91.99 ± 4.07	85.79 ± 5.18	95.17 ± 1.34	94.72 ± 4.20
13	97.45 ± 3.03	97.43 ± 1.08	98.49 ± 1.68	99.01 ± 0.98	99.06 ± 0.77	96.12 ± 1.38	99.14 ± 0.56	**99.29 ± 0.72**
14	98.58 ± 1.06	99.38 ± 0.28	99.91 ± 0.09	99.11 ± 1.73	97.45 ± 2.52	95.43 ± 3.50	99.88 ± 0.05	**99.96 ± 0.06**
15	96.28 ± 2.67	96.86 ± 2.33	99.10 ± 1.29	98.77 ± 2.21	81.20 ± 12.20	83.69 ± 3.27	97.98 ± 1.02	**99.35 ± 0.94**
16	86.66 ± 6.41	95.80 ± 5.03	95.28 ± 3.47	87.58 ± 0.22	98.75 ± 1.37	81.79 ± 3.66	**100.00 ± 0.00**	93.93 ± 4.85
OA(%)	94.33 ± 4.74	97.47 ± 0.31	98.25 ± 0.36	97.43 ± 4.37	92.11 ± 1.42	93.97 ± 2.76	97.47 ± 0.42	**98.47 ± 0.30**
AA(%)	93.22 ± 3.79	90.99 ± 2.87	95.69 ± 1.37	95.59 ± 2.33	93.53 ± 0.88	86.67 ± 7.73	93.20 ± 3.77	**95.74 ± 2.80**
KPP(×100)	93.51 ± 5.54	97.12 ± 0.35	98.01 ± 0.41	97.00 ± 8.74	90.98 ± 1.62	95.39 ± 3.17	97.11 ± 0.13	**98.14 ± 0.34**

Table 9
Classification results for the PU dataset (1% training samples)

Class	DBDA TGRS 2020 Li et al. (2020)	CEGCN TGRS 2021 Liu et al. (2021)	MSSGU TGRS 2022 Liu et al. (2022)	A-HySN IJRS 2023 Fan et al. (2023)	UFMS-LN TGRS 2024 Li, Xu et al. (2024)	DATN EAAI 2024 Shu et al. (2024)	LMFFBNet ESWA 2024 Shi et al. (2024)	HGSTNet
1	96.73 ± 1.60	99.02 ± 0.83	98.41 ± 1.54	96.41 ± 2.12	99.36 ± 0.60	94.31 ± 3.74	98.52 ± 1.16	**99.38 ± 0.55**
2	99.32 ± 0.38	99.92 ± 0.15	99.89 ± 0.10	99.00 ± 1.02	**99.97 ± 0.05**	97.63 ± 1.53	99.40 ± 0.53	**99.97 ± 0.05**
3	90.04 ± 6.67	88.44 ± 6.87	97.31 ± 5.93	97.11 ± 2.89	96.26 ± 0.57	80.66 ± 14.71	97.65 ± 0.27	**97.68 ± 4.08**
4	**97.97 ± 1.15**	94.56 ± 1.65	96.37 ± 1.09	97.00 ± 2.17	97.36 ± 0.97	97.80 ± 3.84	97.49 ± 1.55	97.58 ± 5.73
5	99.12 ± 0.79	99.97 ± 0.03	**100.00 ± 0.00**	97.11 ± 2.89	**100.00 ± 0.00**	99.62 ± 0.55	99.65 ± 0.27	**100.00 ± 0.00**
6	97.97 ± 2.23	99.90 ± 0.10	**100.00 ± 0.00**	98.88 ± 0.22	99.99 ± 0.02	93.42 ± 4.29	99.54 ± 0.36	**100.00 ± 0.00**
7	96.84 ± 3.01	99.52 ± 0.98	**100.00 ± 0.00**	95.47 ± 5.48	99.26 ± 1.10	88.48 ± 11.12	99.70 ± 0.17	97.94 ± 2.64
8	89.46 ± 5.97	98.67 ± 2.27	98.40 ± 1.45	95.00 ± 2.91	97.39 ± 0.86	79.64 ± 9.74	92.54 ± 3.78	**99.55 ± 0.56**
9	98.39 ± 1.27	**99.58 ± 0.42**	98.74 ± 0.94	80.66 ± 9.44	99.20 ± 0.74	98.97 ± 1.28	97.57 ± 1.54	98.04 ± 1.20
OA(%)	97.12 ± 0.69	98.43 ± 0.17	98.70 ± 0.21	97.33 ± 1.99	99.30 ± 0.10	93.65 ± 2.38	98.26 ± 1.29	**99.36 ± 0.25**
AA(%)	96.21 ± 0.96	97.42 ± 0.82	97.32 ± 1.16	95.37 ± 1.20	98.82 ± 0.20	93.66 ± 11.08	97.61 ± 1.77	**98.90 ± 0.57**
KPP(×100)	96.19 ± 0.88	98.25 ± 0.23	98.52 ± 0.24	96.55 ± 4.87	99.07 ± 0.19	91.57 ± 3.18	97.70 ± 0.54	**99.28 ± 0.34**

Table 10
Classification results for the SA dataset (1% training samples)

Class	DBDA TGRS 2020 Li et al. (2020)	CEGCN TGRS 2021 Liu et al. (2021)	MSSGU TGRS 2022 Liu et al. (2022)	A-HySN IJRS 2023 Fan et al. (2023)	UFMS-LN TGRS 2024 Li, Xu et al. (2024)	DATN EAAI 2024 Shu et al. (2024)	LMFFBNet ESWA 2024 Shi et al. (2024)	HGSTNet
1	99.62 ± 1.13	99.86 ± 0.36	**100.00 ± 0.00**	99.62 ± 0.35	**100.00 ± 0.00**	99.51 ± 1.07	99.54 ± 0.36	**100.00 ± 0.00**
2	99.14 ± 2.36	**100 ± 0**	99.99 ± 0.22	99.55 ± 0.44	99.99 ± 0.02	97.99 ± 3.53	99.77 ± 0.23	**100.00 ± 0.00**
3	97.45 ± 1.49	99.87 ± 0.38	99.97 ± 0.07	99.78 ± 0.19	**100.00 ± 0.00**	95.45 ± 1.42	**100.00 ± 0.00**	**100.00 ± 0.00**
4	94.77 ± 5.01	**99.51 ± 0.65**	98.92 ± 0.99	98.66 ± 0.10	99.50 ± 0.32	97.71 ± 1.80	96.83 ± 2.84	98.71 ± 1.36
5	98.02 ± 3.36	98.86 ± 0.80	99.68 ± 0.20	99.67 ± 0.38	99.50 ± 0.32	97.32 ± 2.48	99.66 ± 0.08	**99.69 ± 0.42**
6	97.99 ± 1.01	99.95 ± 0.07	99.90 ± 0.21	99.93 ± 0.03	99.91 ± 0.15	99.94 ± 0.10	99.00 ± 1.00	**100.00 ± 0.00**
7	97.63 ± 3.85	**99.98 ± 0.02**	99.88 ± 0.17	99.68 ± 0.32	**99.98 ± 0.02**	97.16 ± 1.36	95.59 ± 4.37	99.92 ± 0.09
8	89.57 ± 8.56	98.18 ± 1.86	**99.74 ± 0.24**	99.62 ± 0.37	99.02 ± 0.47	87.48 ± 6.56	95.59 ± 4.37	99.67 ± 0.46
9	96.18 ± 3.67	99.99 ± 0.01	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	98.16 ± 1.84	98.45 ± 1.43	**100.00 ± 0.00**
10	96.55 ± 2.96	97.97 ± 1.43	**99.51 ± 0.48**	99.35 ± 0.62	98.18 ± 1.25	96.81 ± 3.14	98.99 ± 1.00	99.36 ± 0.88
11	94.17 ± 2.96	99.86 ± 0.16	**99.96 ± 0.08**	99.66 ± 0.21	99.64 ± 0.20	92.10 ± 2.92	98.48 ± 1.37	99.84 ± 0.20
12	98.27 ± 0.84	**100.00 ± 0.00**	99.96 ± 0.08	99.95 ± 0.05	99.96 ± 0.08	98.79 ± 1.80	**100.00 ± 0.00**	99.96 ± 0.08
13	98.42 ± 0.71	**99.89 ± 0.10**	99.05 ± 1.15	99.56 ± 0.10	99.31 ± 0.65	99.18 ± 0.82	98.95 ± 1.03	99.63 ± 1.01
14	91.44 ± 2.89	98.48 ± 1.60	99.27 ± 0.76	99.16 ± 0.93	98.89 ± 0.39	99.21 ± 0.42	98.04 ± 0.40	**99.32 ± 0.64**
15	88.80 ± 7.42	99.07 ± 0.48	98.89 ± 1.13	99.42 ± 0.35	98.92 ± 0.75	82.38 ± 10.14	98.90 ± 0.89	**99.52 ± 0.66**
16	96.95 ± 1.09	99.71 ± 0.37	99.68 ± 0.72	99.30 ± 0.18	98.89 ± 0.83	**100.00 ± 0.00**	99.94 ± 0.07	98.63 ± 1.84
OA(%)	97.70 ± 0.93	99.25 ± 0.41	99.45 ± 0.10	99.31 ± 0.19	99.39 ± 0.10	95.57 ± 3.82	98.71 ± 0.27	**99.59 ± 0.13**
AA(%)	97.57 ± 0.82	98.49 ± 4.52	99.28 ± 0.32	99.24 ± 0.24	99.45 ± 0.10	97.71 ± 1.28	99.14 ± 0.10	**99.51 ± 0.11**
KPP(×100)	96.09 ± 2.17	99.16 ± 0.40	99.39 ± 0.16	99.48 ± 0.24	99.33 ± 0.10	97.62 ± 1.21	98.57 ± 1.27	**99.55 ± 0.13**

Table 11
Classification results for the XZ dataset (1% training samples)

Class	DBDA TGRS 2020 Li et al. (2020)	CEGCN TGRS 2021 Liu et al. (2021)	MSSGU TGRS 2022 Liu et al. (2022)	A-HySN IJRS 2023 Fan et al. (2023)	UFMS-LN TGRS 2024 Li, Xu et al. (2024)	DATN EAAI 2024 Shu et al. (2024)	LMFFBNet ESWA 2024 Shi et al. (2024)	HGSTNet
1	96.33 ± 0.06	98.17 ± 0.77	98.41 ± 0.43	**98.69 ± 0.67**	97.83 ± 1.60	96.26 ± 2.58	94.01 ± 5.77	98.47 ± 0.61
2	97.94 ± 1.61	99.44 ± 0.35	99.49 ± 0.43	95.05 ± 5.18	99.62 ± 0.22	97.49 ± 2.47	99.40 ± 0.22	**99.74 ± 0.20**
3	93.75 ± 2.57	98.69 ± 0.13	98.96 ± 0.73	95.01 ± 0.98	98.45 ± 1.45	98.16 ± 1.42	98.26 ± 0.66	**99.30 ± 1.00**
4	95.07 ± 3.07	99.00 ± 0.25	99.31 ± 0.63	97.00 ± 1.71	99.48 ± 0.29	98.98 ± 0.56	**99.64 ± 0.23**	97.75 ± 2.54
5	96.22 ± 0.95	96.34 ± 0.05	94.44 ± 0.74	99.10 ± 0.80	98.87 ± 0.08	98.96 ± 0.58	99.64 ± 0.69	**99.70 ± 0.26**
6	85.38 ± 7.41	97.65 ± 0.42	98.86 ± 0.33	97.55 ± 2.75	**99.16 ± 0.61**	98.32 ± 1.12	95.28 ± 3.35	94.60 ± 7.64
7	92.77 ± 6.13	93.52 ± 3.51	97.55 ± 0.45	97.50 ± 1.08	99.19 ± 0.51	98.43 ± 1.00	96.53 ± 3.51	**99.35 ± 4.73**
8	90.17 ± 7.37	97.70 ± 0.29	99.47 ± 0.12	95.00 ± 2.91	98.64 ± 0.25	99.46 ± 0.34	98.74 ± 0.78	**99.70 ± 3.84**
9	95.82 ± 0.27	97.48 ± 2.42	**97.54 ± 2.94**	97.34 ± 3.28	97.20 ± 1.47	97.07 ± 1.34	94.86 ± 4.41	97.48 ± 3.67
OA(%)	95.12 ± 2.64	97.92 ± 0.75	98.40 ± 1.77	97.85 ± 2.10	98.43 ± 0.07	98.22 ± 0.78	97.68 ± 2.19	**98.72 ± 3.94**
AA(%)	94.45 ± 3.22	97.72 ± 0.62	97.88 ± 1.45	96.27 ± 3.25	98.30 ± 0.21	95.63 ± 4.08	98.22 ± 1.58	**98.74 ± 7.98**
KPP(×100)	94.29 ± 2.88	97.25 ± 0.43	98.15 ± 2.24	97.05 ± 0.78	98.07 ± 0.90	97.79 ± 1.71	98.04 ± 0.41	**98.38 ± 4.98**

Table 12
Classification results for the HS dataset (5% training samples)

Class	DBDA TGRS 2020 Li et al. (2020)	CEGCN TGRS 2021 Liu et al. (2021)	MSSGU TGRS 2022 Liu et al. (2022)	A-HySN IJRS 2023 Fan et al. (2023)	UFMS-LN TGRS 2024 Li, Xu et al. (2024)	DATN EAAI 2024 Shu et al. (2024)	LMFFBNet ESWA 2024 Shi et al. (2024)	HGSTNet
1	90.54 ± 4.47	98.00 ± 0.11	98.02 ± 0.04	98.80 ± 0.10	97.30 ± 2.25	91.31 ± 6.47	96.41 ± 3.28	**99.64 ± 0.16**
2	98.69 ± 0.27	98.36 ± 0.08	96.54 ± 0.61	97.00 ± 1.10	98.25 ± 0.44	97.67 ± 2.23	**99.93 ± 0.03**	98.87 ± 0.20
3	**100.00 ± 0.00**	99.95 ± 0.10	**100.00 ± 0.00**	95.91 ± 2.89	99.98 ± 0.02	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
4	99.36 ± 0.15	99.51 ± 0.18	98.56 ± 0.58	98.35 ± 1.07	98.54 ± 0.66	97.80 ± 3.84	98.98 ± 1.02	**100.00 ± 0.00**
5	98.12 ± 0.41	99.48 ± 0.87	99.39 ± 0.87	91.91 ± 8.09	99.02 ± 0.47	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
6	98.76 ± 0.23	99.90 ± 0.15	**100.00 ± 0.00**	98.60 ± 1.34	97.18 ± 1.27	99.26 ± 0.73	**100.00 ± 0.00**	92.43 ± 6.58
7	89.95 ± 5.10	96.07 ± 0.64	96.80 ± 1.21	96.14 ± 1.45	96.05 ± 1.08	97.48 ± 2.22	96.74 ± 1.71	**98.15 ± 0.38**
8	86.19 ± 8.79	96.17 ± 1.20	97.55 ± 1.51	97.10 ± 1.94	97.35 ± 2.64	**99.74 ± 0.18**	**99.74 ± 0.18**	97.87 ± 1.58
9	89.07 ± 4.79	97.46 ± 0.74	**97.86 ± 0.93**	84.66 ± 9.44	95.18 ± 2.52	94.73 ± 4.08	97.46 ± 1.14	93.25 ± 5.97
10	85.43 ± 8.67	99.01 ± 0.19	98.69 ± 1.00	98.94 ± 0.62	95.48 ± 2.22	94.76 ± 3.41	97.68 ± 2.13	**100.00 ± 0.00**
11	90.15 ± 3.46	99.89 ± 0.16	99.86 ± 0.28	97.58 ± 2.01	98.75 ± 1.53	94.31 ± 3.24	97.20 ± 2.07	**99.90 ± 0.09**
12	91.12 ± 6.48	95.89 ± 0.37	97.77 ± 0.65	98.34 ± 0.60	99.31 ± 0.65	95.69 ± 3.21	99.37 ± 0.55	**99.63 ± 0.37**
13	88.52 ± 7.11	91.73 ± 1.02	96.23 ± 0.52	91.56 ± 2.33	96.45 ± 0.22	91.18 ± 4.27	98.24 ± 1.74	**98.25 ± 2.41**
14	97.20 ± 1.19	**100.00 ± 0.00**	**100.00 ± 0.00**	98.80 ± 0.56	97.45 ± 1.20	96.22 ± 2.42	95.77 ± 4.23	**100.00 ± 0.00**
15	96.70 ± 2.24	99.92 ± 0.09	99.98 ± 0.03	99.79 ± 0.20	98.97 ± 0.59	97.52 ± 2.12	99.18 ± 0.36	**100.00 ± 0.00**
OA(%)	92.53 ± 4.96	98.07 ± 0.19	98.31 ± 0.17	97.64 ± 1.79	97.41 ± 0.59	96.37 ± 2.48	98.37 ± 1.51	**98.72 ± 0.47**
AA(%)	93.04 ± 3.64	97.85 ± 0.19	98.35 ± 0.16	95.50 ± 0.57	97.47 ± 0.64	96.56 ± 3.48	98.02 ± 1.94	**98.41 ± 0.39**
KPP(×100)	91.37 ± 4.38	97.91 ± 0.20	97.00 ± 2.84	96.26 ± 1.21	97.07 ± 0.24	96.21 ± 3.81	98.14 ± 0.79	**98.39 ± 0.40**

4.2. Experimental results

To thoroughly evaluate HGSTNet, we conduct a series of qualitative and quantitative experiments, exploring various aspects such as visual result, ablation study, feature extraction and fusion, graph learning ability, learning efficacy and computational efficiency. Unless otherwise specified, all experiments in this section adhere to the same configuration outlined in Section 3.

4.2.1. Experiments comparison

Comparative experiments were conducted to assess the effectiveness of HGSTNet. We evaluated its performance against seven established deep learning methods using five standard public datasets: IP, PU, SA, XZ and HS. These methods include: DBDA (Li et al., 2020), CEGCN (Liu et al., 2021), MSSGU (Liu et al., 2022), A-HySN (Fan et al., 2023), DATN (Shu et al., 2024), UFMS-LN (Li, Xu et al., 2024) and LMFFBNet (Shi et al., 2024). The average and standard deviation for each metric were calculated across ten independent repetitions of each experiment. Tables 8–12 provide detailed quantitative accuracies for each dataset using different methods, where the best result is highlighted in bold, and the second-best result is underlined. Figs. 9, 10, 11, 12 and 13 show the classification visualization results for the IP dataset, PU dataset, SA dataset, XZ dataset, and HS dataset, respectively.

As demonstrated in Table 6 and Fig. 9, HGSTNet model surpasses other comparative methods in classification performance on the IP dataset. Methods such as DBDA, LMFFBNet, DATN, while achieving high precision on specific categories through multi-layered convolution, pooling, and attention mechanisms, may be constrained by the fixed nature of convolutional kernels, potentially limiting their feature extraction capability on intricate datasets. UFMS-LN employs a compressed multi-head attention mechanism to extract features in both spatial and spectral dimensions, benefitting from a wider receptive field to effectively capture pixel-level characteristics. A-HySN, by enhancing pixel-level features through preprocessing, also contributes to improved classification performance. CEGCN and MSSGU exhibit excellent classification results, demonstrating that a refined network

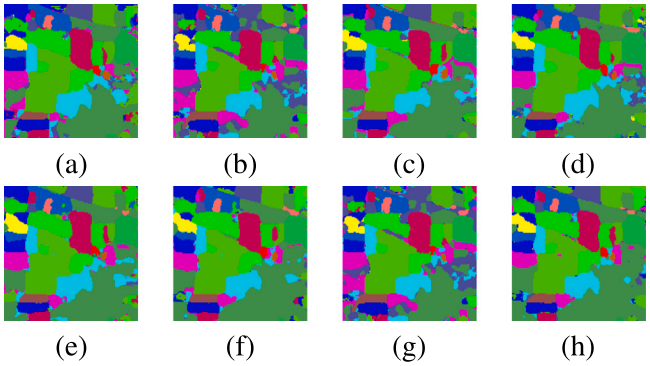


Fig. 9. Visualizing classification results on the IP dataset using different methods. (a) DBDA (OA = 94.22%). (b) CEGCN (OA = 97.05%). (c) MSSGU (OA = 97.93%). (d) A-HySN (OA = 97.28%). (e) DATN (OA = 96.58%). (f) UFMS-LN (OA = 97.40%). (g) LMFFBNet (OA = 97.14%). (h) HGSTNet (OA = 98.47%).

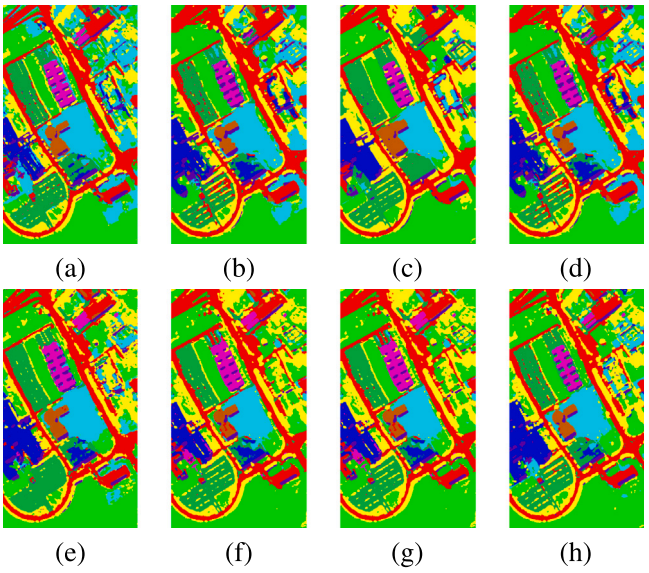


Fig. 10. Visualizing classification results on the PU dataset using different methods. (a) DBDA (OA = 94.66%). (b) CEGCN (OA = 98.27%). (c) MSSGU (OA = 99.01%). (d) A-HySN (OA = 98.27%). (e) DATN (OA = 98.03%). (f) UFMS-LN (OA = 99.30%). (g) LMFFBNet (OA = 98.38%). (h) HGSTNet (OA = 99.46%).

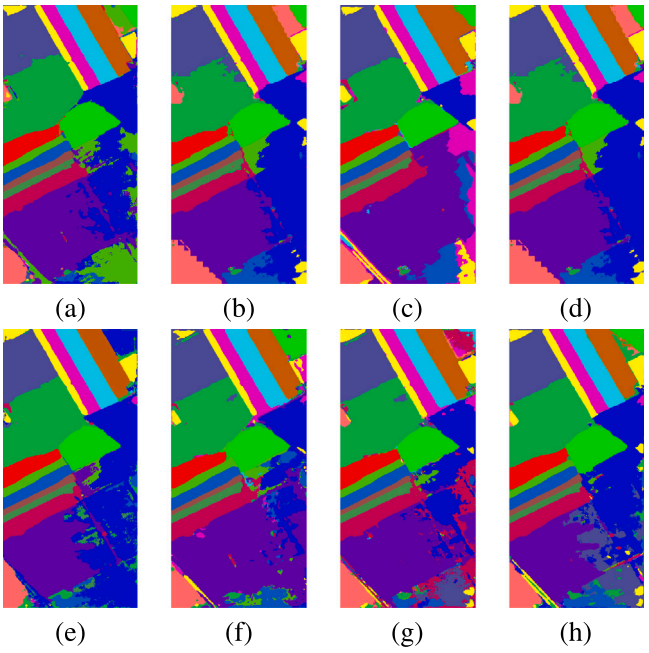


Fig. 11. Visualizing classification results on the SA dataset using different methods. (a) DBDA (OA = 94.66%). (b) CEGCN (OA = 98.27%). (c) MSSGU (OA = 99.01%). (d) A-HySN (OA = 98.27%). (e) DATN (OA = 98.03%). (f) UFMS-LN (OA = 99.34%). (g) LMFFBNet (OA = 98.38%). (h) HGSTNet (OA = 99.59%).

structure design can further enhance classification accuracy. However, ours HGSTNet achieves superior performance in categories 6, 8, 10, 11, 13, 14, and 15, illustrating its capability to adaptively fuse land cover features across different stages by effectively integrating diverse features. Visually, HGSTNet's classification results generating fewer errors and exhibiting boundary alignments more consistent with actual land cover boundaries. This observation reaffirms HGSTNet's adaptive multi-scale learning capacity.

Table 7 provides a comprehensive comparison of classification performance on the PU dataset, demonstrating the superior results

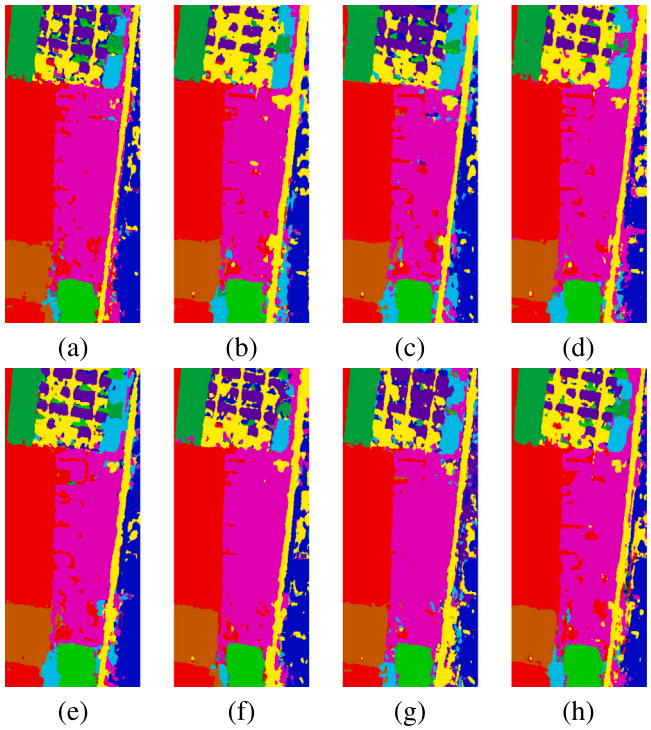


Fig. 12. Visualizing classification results on the XZ dataset using different methods. (a) DBDA (OA = 95.38%). (b) CEGCN (OA = 97.94%). (c) MSSGU (OA = 98.43%). (d) A-HySN (OA = 97.04%). (e) DATN (OA = 97.77%). (f) UFMS-LN (OA = 98.30%). (g) LMFFBNet (OA = 98.34%). (h) HGSTNet (OA = 99.24%).

achieved by our proposed method. UFMS-LN, possessing a wider receptive field, is able to extract more spectral information and therefore exhibits certain advantages. Notably, CEGCN and MSSGU achieve optimal accuracy in several categories. In comparison, HGSTNet showcases superior classification precision across numerous categories. HGSTNet demonstrates a distinct advantage in handling areas with uneven class distribution, exhibiting the flexibility to learn the distribution of surrounding environments for objects, thus ensuring regional consistency. Fig. 10 supports this observation, particularly evident in the asphalt road (red area). HGSTNet achieved continuous boundaries in the narrow path region (red) with complete clustering of surrounding trees (yellow), making each land cover category clearly discernible. This results in a low error rate, particularly within areas where categories are closely connected.

For the SA dataset, Table 8 and Fig. 11, reveal that HGSTNet outperforms all benchmark methods in classification performance. The SA dataset primarily comprises land cover, making the simulation of spatial land cover topology challenging, particularly for categories 9 and 12 (plant cultivation), where spectral similarity makes differentiation difficult. While CEGCN and MSSGU achieve optimal accuracy for these categories and demonstrate high accuracy in other categories, HGSTNet attains 100% accuracy in categories 1, 2, 3, 6, and 9, and maintains accuracy between 99%–100% for other categories. This outcome demonstrates HGSTNet's ability to capture multi-scale spatial and spectral information effectively, thus enabling precise pixel-level classification tasks.

Similarly, Tables 11 and 12 demonstrate that HGSTNet achieves superior classification performance on the XZ and HS datasets compared to all other methods. On the XZ dataset, HGSTNet attains the highest accuracy in categories 2, 3, 5, 7, and 8, surpassing the second-best models by 0.44% and 0.23% in categories 3 and 8, respectively. On the HS dataset, HGSTNet achieves 100% classification accuracy in categories 3, 4, 5, 10, 14, and 15, showcasing its ability to capture

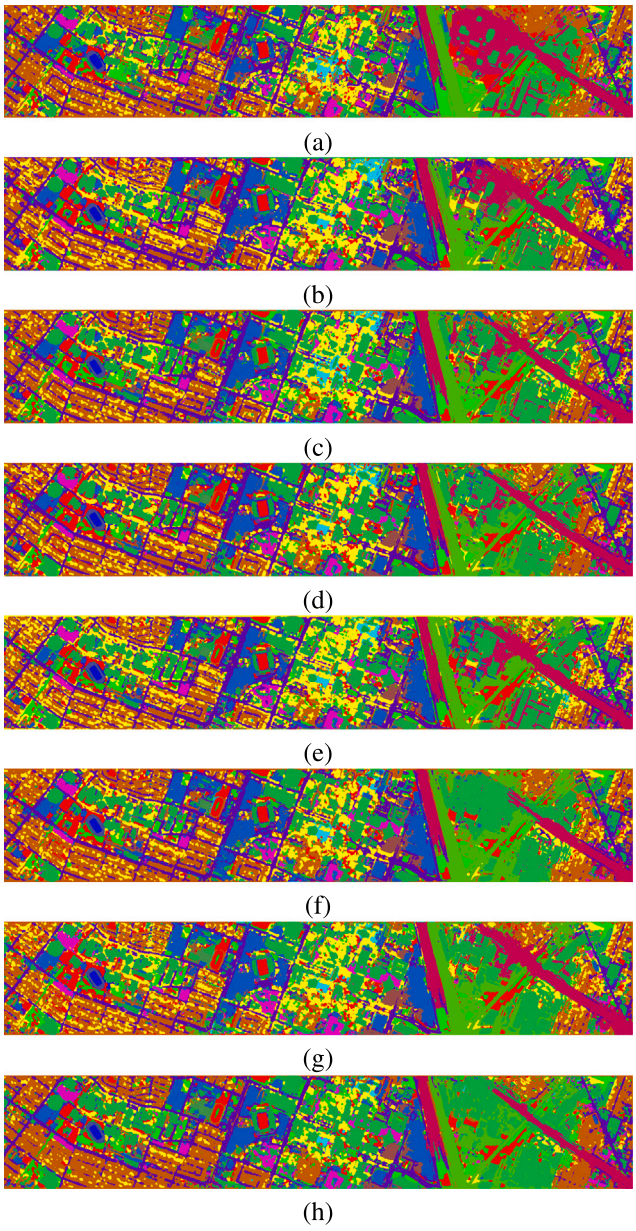


Fig. 13. Visualizing classification results on the HS dataset using different methods. (a) DBDA (OA = 92.62%). (b) CEGCN (OA = 97.22%). (c) MSSGU (OA = 98.34%). (d) A-HySN (OA = 98.27%). (e) DATN (OA = 96.51%). (f) UFMS-LN (OA = 97.80%). (g) LMFFBNet (OA = 98.35%). (h) HGSTNet (OA = 98.73%).

diverse land cover features and adapt to complex terrains. Figs. 12 and 13 further illustrate HGSTNet's ability to generate classification maps closely aligned with the ground truth, particularly in urban regions with intricate land cover patterns and mixed-class boundaries. For instance, in the XZ dataset, HGSTNet demonstrates remarkable boundary continuity in narrow urban roads (e.g., the yellow region), where other methods often fail, producing blurred or discontinuous boundaries. Similarly, on the HS dataset, HGSTNet precisely distinguishes fragmented land cover regions, delivering sharp boundaries and minimal misclassification. While some comparative methods, such as DBDA and MSSGU, achieve competitive performance in specific categories (e.g., DBDA attains 100% accuracy in category 3 on the XZ dataset), their overall performance is limited. In contrast, HGSTNet excels in both spatial feature learning and regional consistency, making it significantly more effective in handling complex land cover scenarios.

Table 13
The hierarchical graph merge block placement in encoder–decoder layers.

Methods	Encoder-Decoder					OA(%)				
	1	2	3	4	5	IP	PU	SA	XZ	HS
1	×	×	×	×	×	80.67	84.19	87.97	86.72	84.33
2	✓	×	×	×	×	81.41	85.56	87.12	86.07	85.28
3	×	✓	×	×	×	83.89	86.89	88.04	89.23	87.72
4	×	×	✓	×	×	82.43	86.57	88.16	89.17	87.10
5	×	×	×	✓	×	82.05	86.41	88.56	89.12	87.07
6	×	×	×	×	✓	84.19	86.94	89.05	89.66	87.67
7	✓	✓	×	×	×	87.67	88.26	88.11	90.11	89.70
8	×	✓	✓	×	×	87.74	88.79	90.09	91.58	90.01
9	×	×	✓	✓	×	87.70	88.71	90.14	91.44	90.07
10	×	×	×	✓	✓	88.14	90.01	91.65	92.21	91.70
11	✓	✓	✓	×	×	93.05	94.14	94.65	95.17	94.54
12	×	✓	✓	✓	×	94.37	95.79	95.04	95.50	94.98
13	×	×	✓	✓	✓	95.25	96.61	96.40	96.00	95.33
14	✓	✓	✓	✓	✓	98.05	99.01	99.56	98.84	98.00
15	✓	✓	✓	✓	×	97.91	98.53	98.00	98.17	97.82
16	×	✓	✓	✓	✓	**98.47**	**99.39**	**99.59**	**99.24**	**98.73**

Table 14
Comparison of OA, run time, and parameters across datasets using different transformer.

Module	Dataset	OA (%)	Run time (s)	Parameter (K)
Without	IP	97.93	65.75	169.86
	PU	98.84	400.82	278.58
	SA	99.01	171.54	226.10
	XZ	98.43	100.78	717.03
	HS	98.34	587.09	787.77
Self + MLP (Transformer)	IP	98.15	50.07	157.11
	PU	99.01	327.77	221.14
	SA	99.23	143.48	226.10
	XZ	98.89	98.61	557.38
	HS	98.66	478.88	647.87
SPA + FFN (Sparse transformer)	IP	**98.44**	**43.97**	**143.24**
	PU	**99.46**	**178.84**	**188.34**
	SA	**99.59**	**100.88**	**178.19**
	XZ	**99.24**	**70.91**	**488.65**
	HS	**98.73**	**412.92**	**587.54**

In summary, HGSTNet employs a multilevel graph learning mechanism to extract multi-scale spatial and spectral features and integrate them into a unified representation, enabling effective classification of complex land covers. HGSTNet effectively captures irregular areas and their surrounding environments, ensuring consistent classification results, demonstrating its effectiveness in HSI classification.

4.3. Ablation study

In this section, we will validate the impact of two key modules, the hierarchical graph merge block and the sparse transformer block, on the HGSTNet's performance through detailed ablation experiments.

4.3.1. Impact of the hierarchical graph merge block

As shown in Table 13, "✓" indicates the insertion of the hierarchical graph merge block, while "×" represents its absence. When all hierarchical graph merge blocks are removed, the OA across all datasets drops to the lowest levels, underscoring the critical role of hierarchical graph merge block in enabling effective multi-scale feature fusion and improving classification performance. When hierarchical graph merge block is inserted only into the first layer of the encoder–decoder, the model's performance experiences a noticeable decline. This can be attributed to the fact that the first layer is primarily tasked with extracting low-level local features, where the spatial relationships and geometric structures of features remain unstable and the information is sparse. Conducting graph-based feature fusion at this stage risks losing essential fine-grained details, ultimately hindering the model's ability to effectively capture global semantic relationships.

Table 15
Influence of the ReLU in the SPA (On the IP, PU, SA dataset).

Activate function	Parameters (K)			Run times (s)			OA (%)		
	IP	PU	SA	IP	PU	SA	IP	PU	SA
Softmax	157.11	221.14	226.10	50.07	327.77	143.48	98.18	99.01	99.23
PReLU	149.77	197.40	191.71	47.30	243.87	130.81	98.22	99.37	99.40
Mish	147.54	192.52	180.31	45.10	200.67	124.48	98.30	99.39	99.44
Lush	158.45	196.47	188.62	48.58	222.29	128.10	98.23	99.39	99.37
ReLU	**143.24**	**188.34**	**178.19**	**43.97**	**178.84**	**100.88**	**98.44**	**99.46**	**99.59**

As hierarchical graph merge block is progressively inserted into deeper layers, the OA consistently improves. Specifically, inserting hierarchical graph merge blocks from the second layer to the fifth layer results in a substantial enhancement in performance. The highest OA is achieved when hierarchical graph merge blocks are applied to all layers except the first. This result demonstrates that hierarchical graph merge block operates effectively in the feature domain, leveraging hierarchical graph structures to facilitate multi-layer integration of spatial and semantic information. Furthermore, the gradual improvement in OA across deeper layers confirms that hierarchical graph merge blocks are most effective when applied to more semantically meaningful feature maps.

4.3.2. Impact of the sparse transformer block

As shown in Table 14, when HGSTNet does not employ a transformer, its OA, runtime, and parameter count exhibit the poorest performance.

In contrast, the model incorporating the sparse transformer module (SPA + FFN) achieves the best results in terms of OA, runtime, and parameter count across all datasets. For instance, on the IP dataset, the sparse transformer achieves an OA of 98.44%, whereas the traditional transformer and the model without a transformer achieve OA values of 98.15% and 97.93%, respectively, clearly lagging behind the sparse transformer. This demonstrates that the sparse transformer effectively captures critical features, thereby enhancing classification performance. In terms of runtime, the efficiency advantage of the sparse transformer is particularly significant. On the IP dataset, the runtime of SPA + FFN is 43.97 s, which is noticeably lower than the 50.07 s of the traditional transformer and the 65.75 s of the model without a transformer. On the PU dataset, the runtime of SPA + FFN is 178.84 s, which is nearly half of the 327.77 s of the traditional transformer and significantly lower than the 400.82 s of the model without a transformer. These results demonstrate that the sparse transformer, effectively reduces redundant computations and improves computational efficiency. In terms of parameter count, the sparse transformer also exhibits remarkable optimization. For example, on the IP dataset, the parameter count of SPA + FFN is 143.24K, which is reduced by 8.8K and 15.7K compared to the traditional transformer (157.11K) and the model without a transformer (169.86K), respectively. On larger datasets, such as the XZ and HS datasets, the sparse transformer continues to demonstrate parameter optimization, further validating its computational efficiency.

In conclusion, the sparse transformer leverages its sparse self-attention mechanism to suppress low-relevance or redundant information, reducing computational complexity. The experimental results confirm that the sparse transformer outperforms traditional transformers and non-transformer-based models in terms of classification performance, computational efficiency, and parameter optimization.

Influence of the relu in the SPA. To better evaluate the role of the ReLU activation function in the SPA, we designed a series of ablation experiments focusing on assessing the necessity and performance of ReLU within the SPA module. These experiments systematically analyze the contributions of ReLU to parameter efficiency, computational speed, and OA by comparing it with other activation functions, such as Softmax, PReLU, Mish, and Lush.

Table 16
Influence of the ReLU in the SPA (On the XZ, HS dataset).

Activate function	Parameters (K)		Run times (s)		OA (%)	
	XZ	HS	XZ	HS	XZ	HS
Softmax	557.38	647.87	98.61	478.88	98.89	98.66
PReLU	529.31	589.31	84.41	500.10	98.98	98.49
Mish	517.43	600.86	76.52	468.42	99.14	98.60
Lush	531.47	596.58	83.79	493.63	99.07	98.54
ReLU	**488.65**	**587.54**	**70.91**	**412.92**	**99.24**	**98.73**

Tables 15 and 16 provide detailed comparisons of training time, testing time, and parameter counts across different datasets (IP, PU, SA, XZ, and HS) when ReLU, PReLU, Swish, Mish, and Lush are used as activation functions within HGSTNet's SPA module. The experimental results reveal that the choice of activation function has a significant impact on runtime and parameter counts. For example, among all activation functions, Softmax exhibits the longest runtime and the highest parameter count. On the IP dataset, although the model using Lush achieves a shorter runtime compared to Softmax, it is still slightly higher than ReLU, PReLU, and Mish. This indicates that more complex activation functions increase computational overhead, making the model less efficient. In contrast, ReLU demonstrates the lowest runtime and the smallest computational cost among all activation functions. This highlights that the SPA module in sparse transformers, leveraging ReLU activation, can not only effectively suppress low-relevance or redundant information but also significantly reduce computational complexity.

Overall, the experiments validate the necessity and advantages of using ReLU activation in the SPA module of sparse transformers. Compared to other activation functions, ReLU ensures high classification performance while reducing runtime and computational cost, further confirming its effectiveness and efficiency.

4.4. Visualization of HGSTNet's hierarchical learning

4.4.1. Hierarchical graph learning

To further verify the effectiveness and accuracy of the encoders and decoders used for graph learning in HGSTNet, as well as to demonstrate the feature learning capabilities of the hierarchical graph merge blocks in U-Net, we visualized the learned graph representations on the larger-scale XZ and HS datasets, as shown in Fig. 14. The edge weights in the figure are mapped to a range of $[0, 1]$, with a color palette transitioning from red to deep blue to represent the weight intensity. As illustrated in Fig. 14, HGSTNet successfully learns and captures efficient graph structures on the XZ and HS datasets. Notably, in the final level of the encoder visualizations (XZ-4 and HS-4), the model effectively distinguishes urban areas with distinct land cover types. While these larger datasets showcase the exceptional performance of HGSTNet, the method is equally applicable to smaller datasets such as IN, PU, and SA.

Specifically, boundaries within the same urban land cover regions typically exhibit higher edge weights, whereas boundaries between different categories tend to have lower weights. This observation reflects HGSTNet's precision in handling local area features. Additionally, as the graph level increases, the overall intensity of edge weights decreases, indicating that higher-level graphs prioritize smoothing feature

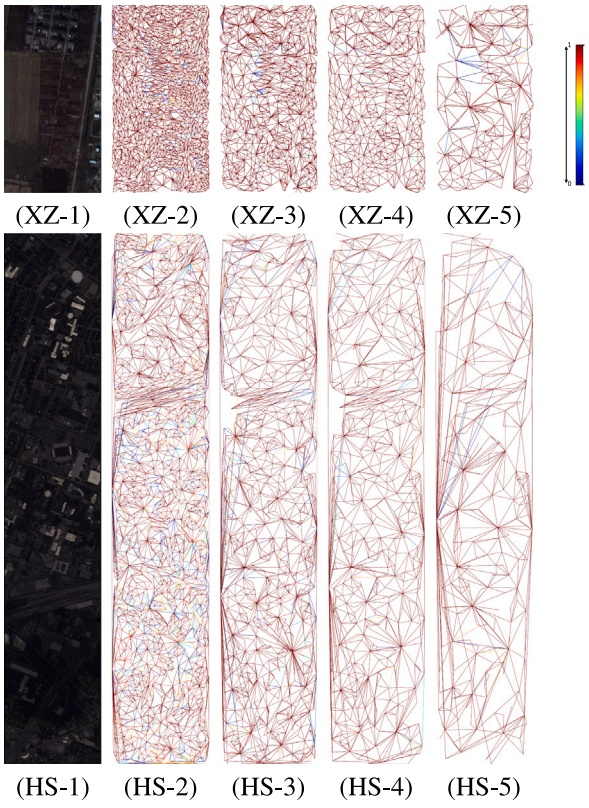


Fig. 14. Visualizing graph learning relationships: (a) XZ: visualized graph edges learned in the second to fifth encoders for the XZ dataset, with line colors representing edge weights. (b) HS: visualized graph edges learned in the second to fifth encoders for the HS dataset.

maps and emphasizing global consistency while becoming less sensitive to subtle differences between land cover categories. To accommodate this feature distinction, we gradually reduce the output feature channels in the encoder design, ensuring that feature representation and discrimination capabilities align with the requirements of different graph levels.

4.4.2. Hierarchical feature representation

This section elaborates on the feature extraction and fusion process of the hierarchical graph merging blocks in the encoder and decoder of the unified U-Net framework, guided by the hierarchical graph structure of HGSTNet. To provide a more intuitive demonstration of this process, we visualized the feature representations learned by the hierarchical graph merging blocks at each stage in Figs. 15 and 16. Furthermore, to validate the effectiveness and generalization capability of HGSTNet across diverse application scenarios, we selected the XZ and HS datasets, which exhibit significant distribution differences and larger scales, for analysis.

The encoder of HGSTNet learns multi-scale features at different stages, which demonstrate remarkable complementarity. On the XZ dataset, the lower-level features extracted by the encoder (e.g., Encoder-1 and Encoder-2) primarily capture clear boundary information and fine-grained textures, which are crucial for identifying building structures and subtle variations in land cover. As the network depth increases (e.g., Encoder-4 and Encoder-5), the model gradually extracts larger-scale features that emphasize the global spatial structure within regions, such as the overall layout of urban areas and the classification of blurred boundary regions. Similarly, on the HS dataset, HGSTNet also exhibits efficient multi-scale feature extraction capabilities. The lower-level features (e.g., Encoder-1 and Encoder-2) capture prominent edge information and local textures, while higher-level features (e.g., Encoder-5) focus on abstract semantic information, such as the global distribution characteristics between different land cover types. During the decoding phase, these multi-scale features are progressively integrated, achieving a seamless transition from coarse to fine representations and ultimately generating more refined feature maps that accurately describe actual ground conditions.

4.5. Limited-sample learning efficiency

We explore the learning capabilities of the HGSTNet under conditions of constrained sample sizes, as well as its advantages compared to other model methodologies. For each category in the three datasets, the number of training samples per category is incrementally increased in groups of five, from 5 to 25. The specific comparison method, OA is illustrated in Fig. 17.

When dealing with limited training samples, methods like DBDA, LMFFNet, UFMS-LN and DATN exhibit significantly lower OA compared to CEGCN and MSSGU. This disparity is primarily attributed to the reliance of those models on substantial sample quantities for optimal performance, where network variability and training stability can be significantly impacted by insufficient samples. This observation underscores the benefit of employing superpixels, which consistently yield higher accuracy with limited training data. Notably, our proposed HGSTNet, merging the advantages of U-Net and transformer architectures, demonstrates superior efficiency due to its ability to integrate multi-level feature extraction and spectral–spatial information learning simultaneously. The trends observed in Fig. 17 effectively illustrate HGSTNet's effectiveness and adaptability in handling scenarios with small sample sizes.

4.6. Performance comparison

To assess the efficiency of the HGSTNet, we have selected various other classification methods for comparison. The comparison includes the training time, testing time, average accuracy, and the number of network parameters for each method. We conducted each experiment ten times, reporting the average values across all datasets. It allows for a robust evaluation of the HGSTNet performance relative to its peers, highlighting both its computational efficiency and its effectiveness in handling classification tasks.

As shown in Table 17, HGSTNet demonstrates excellent overall operational efficiency while achieving high classification accuracy. Consistent with the comparisons presented in Section 4.2.1, the best results are highlighted in bold, and the second-best results are underlined. In practical applications, the depth, scale, and other parameters of HGSTNet can be adjusted according to specific requirements, enabling an optimal balance between classification accuracy, computational cost, and model size.

5. Conclusion and future work

HSI presents challenges due to the varied morphologies and scales of different land covers, resulting in complex spatial structures. To effectively address these challenges, we propose a novel joint U-Nets with hierarchical graph structure and sparse transformer (HGSTNet), whcih employs a hierarchical graph incremental merging method to construct a hierarchical graph merge block, enabling end-to-end training with high precision and effectiveness. This hierarchical graph merge blcoks captures the relationships between land covers at different scales, enhancing the model's ability to learn complex spatial patterns. The model progressively fuses features from coarse to fine scales, facilitating message passing between different graph levels, enabling collaborative learning across scales. This cross-scale feature fusion mechanism leverages information from different scales, enhancing the model ability to handle complex land covers. Additionally, we introduce sparse

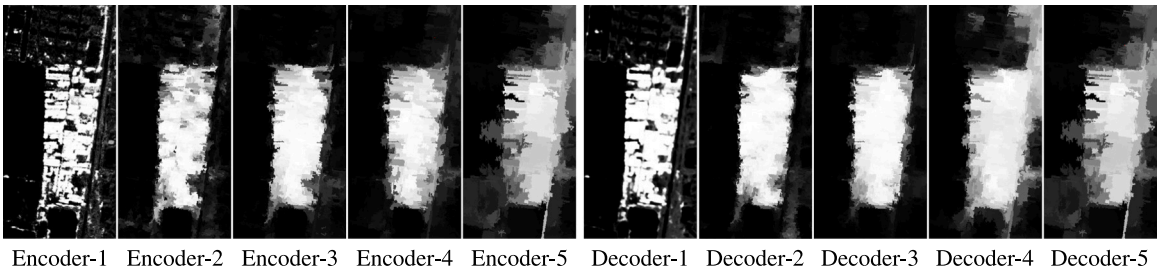


Fig. 15. Map visualization: encoder and decoder of U-Net feature representations on the XZ dataset.

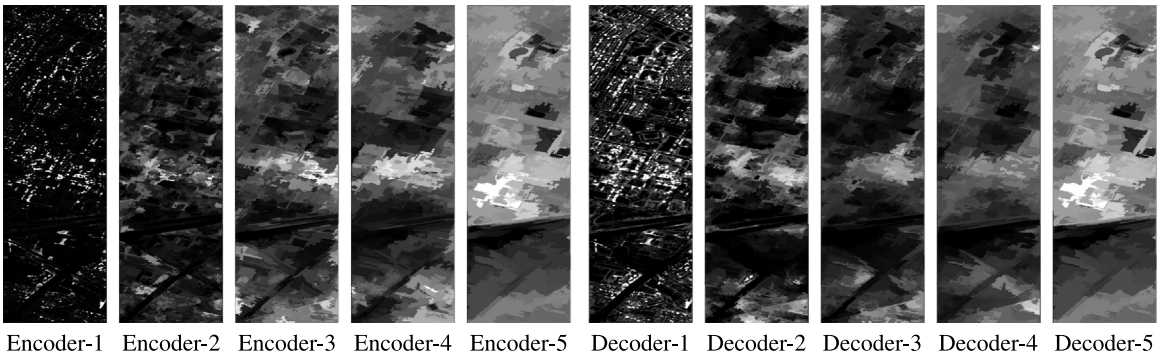


Fig. 16. Map visualization: encoder and decoder of U-Net feature representations on the HS dataset.

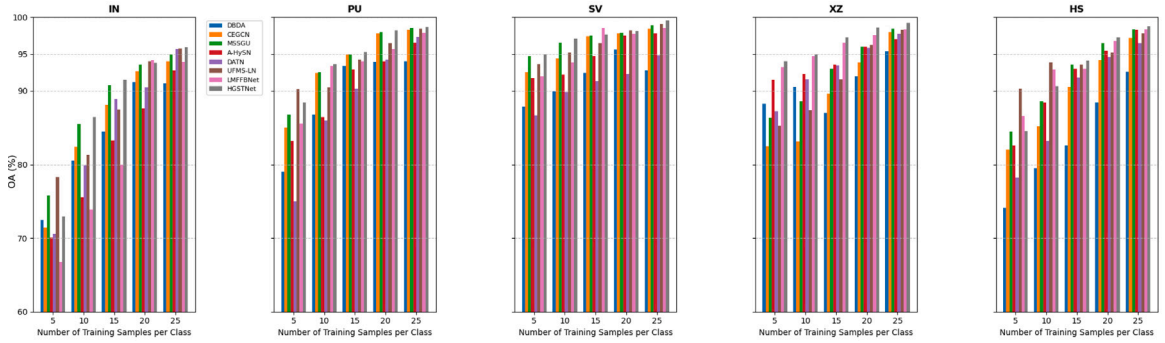


Fig. 17. Comparative analysis of classification accuracy with varying training sample datasets.

Table 17
Performance comparison between different methods.

Dataset	Item	DBDA TGRS 2020	CEGCN TGRS 2021	MSSGU TGRS 2022	A-HySN IJRS 2023	DATN TGRS 2024	UFMS-LN EAAI 2024	LMFFBNet ESWA 2024	HGSTNet
	Reference	Li et al. (2020)	Liu et al. (2021)	Liu et al. (2022)	Fan et al. (2023)	Shu et al. (2024)	Li, Xu et al. (2024)	Shi et al. (2024)	
IP	Training(s).	158.3	46.3	36.5	68.3	41.4	55.7	38.1	**35.8**
	Testing(s).	19.9	**1.6**	2.1	5.8	3.0	6.6	10.4	1.8
	OA(%).	94.33	97.47	98.25	89.03	97.07	95.69	97.40	**98.44**
PU	Training.	216.5	183.7	177.3	191.5	193.4	187.8	179.3	**161.6**
	Testing.	90.7	25.2	19.3	48.8	8.2	11.2	43.9	**16.0**
	OA.	91.77	98.43	98.70	93.01	98.77	97.07	89.04	**99.46**
SA	Training.	237.4	179.6	151.0	209.5	176.6	166.7	104.3	**91.3**
	Testing.	72.7	33.6	20.5	27.4	24.5	23.1	13.8	**9.4**
	OA.	94.70	99.25	99.45	94.31	99.51	98.07	89.30	**99.59**
XZ	Training.	213.7	99.1	88.4	99.5	106.6	76.7	70.5	**65.7**
	Testing.	65.8	20.2	12.3	27.4	34.6	23.5	13.2	**5.1**
	OA.	95.38	97.94	98.43	97.04	97.77	98.30	98.34	**99.24**
HS	Training.	645.3	506.1	442.6	469.5	536.6	467.6	433.2	**377.9**
	Testing.	103.8	66.6	58.5	48.1	84.5	73.1	60.4	**35.9**
	OA.	92.62	97.22	98.34	98.27	96.51	97.80	98.35	**98.73**

transformer blocks during the first four encoder–decoder feature fusion stages, employing sparse self-attention mechanism to accurately extract and learn features across different scales and layers and to achieve spatially adaptive updates, thereby better accommodating pixel classification tasks.

Although our current HGSTNet employs a hierarchical graph incremental merging method to generate hierarchical graph merge block, further exploration of finer scales and more intricate graph structures is necessary to comprehensively characterize the relationships between different ground objects in HSI. In the future, we will explore the integration of local region guidance and bipartite graph-based projection learning's method (Zhang et al., 2024) as alternatives to hierarchical graph incremental merging method and multimodal remote sensing data fusion (Cai et al., 2024) for improved HSI classification.

CRediT authorship contribution statement

Pengfei Zhu: Writing – original draft, Data curation, Software, Validation, Supervision, Formal analysis. **Jinglei Liu:** Conceptualization, Methodology, Project administration, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We are grateful to the reviewers for their detailed and helpful comments, which allowed us to greatly improve this paper. This work was supported by National Natural Science Foundation of China (62273290, 61572419, 62072391).

Data availability

Data will be made available on request.

References

Adão, T., Hruška, J., Pádua, L., Bessa, J., Peres, E., Morais, R., et al. (2017). Hyperspectral imaging: A review on UAV-based sensors, data processing and applications for agriculture and forestry. *Remote Sensing*, *9*(11), 1110.

Asif, N. A., Sarker, Y., Chakrabortty, R. K., Ryan, M. J., Ahamed, M. H., Saha, D. K., et al. (2021). Graph neural network: A comprehensive review on non-euclidean space. *IEEE Access*, *9*, 60588–60606.

Cai, Y., Zhang, Z., Liu, X., Ding, Y., Li, F., & Tan, J. (2024). Learning unified anchor graph for joint clustering of hyperspectral and LiDAR data. *IEEE Transactions on Neural Networks and Learning Systems*.

Fan, J., Zhang, X., Chen, Y., & Sun, C. (2023). Classification of hyperspectral image by preprocessing method based relation network. *International Journal of Remote Sensing*, *44*(22), 6929–6953. http://dx.doi.org/10.1080/01431161.2023.2275325, URL https://doi.org/10.1080/01431161.2023.2275325.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, *15*, 315–323.

Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., et al. (2022). A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*, 87–110.

He, X., Chen, Y., & Lin, Z. (2021). Spatial-spectral transformer for hyperspectral image classification. *Remote Sensing*, *13*(3), http://dx.doi.org/10.3390/rs13030498, URL https://www.mdpi.com/2072-4292/13/3/498.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition* (pp. 770–778). http://dx.doi.org/10.1109/CVPR.2016.90.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., et al. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning* (pp. 1989–1998). Pmlr.

Hong, D., Gao, L., Yao, J., Zhang, B., Plaza, A., & Chanussot, J. (2021). Graph convolutional networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *59*(7), 5966–5978. http://dx.doi.org/10.1109/TGRS.2020.3015157.

Hong, L., & Zhang, M. (2020). Object-oriented multiscale deep features for hyperspectral image classification. *International Journal of Remote Sensing*, *41*, 5549–5572.

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., et al. (2019). Searching for MobileNetV3. In *2019 IEEE/CVF international conference on computer vision* (pp. 1314–1324). http://dx.doi.org/10.1109/ICCV.2019.00140.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition* (pp. 2261–2269). http://dx.doi.org/10.1109/CVPR.2017.243.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. CoRR arXiv:1412.6980. URL https://api.semanticscholar.org/CorpusID:6628106.

Li, W., Liu, Q., Fan, S., Bai, H., & Xin, M. (2023). Multistage superpixel-guided hyperspectral image classification with sparse graph attention networks. *IEEE Transactions on Geoscience and Remote Sensing*, *61*, 1–18. http://dx.doi.org/10.1109/TGRS.2023.3304716.

Li, X., Xu, M., Liu, S., Sheng, H., & Wan, J. (2024). Ultra-lightweight feature-compressed multi-head self-attention learning networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*.

Li, Y., Yang, X., Tang, D., & Zhou, Z. (2024). RDTN: Residual densely transformer network for hyperspectral image classification. *Expert Systems with Applications*, *250*, Article 123939.

Li, R., Zheng, S., Duan, C., Yang, Y., & Wang, X. (2020). Classification of hyperspectral image based on double-branch dual-attention mechanism network. *Remote Sensing*, *12*(3), http://dx.doi.org/10.3390/rs12030582, URL https://www.mdpi.com/2072-4292/12/3/582.

Lin, M., Jing, W., Di, D., Chen, G., & Song, H. (2022). Context-aware attentional graph U-net for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, *19*, 1–5. http://dx.doi.org/10.1109/LGRS.2021.3069987.

Liu, Q., Xiao, L., Yang, J., & Wei, Z. (2021). CNN-enhanced graph convolutional network with pixel- and superpixel-level feature fusion for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *59*, 8657–8671. http://dx.doi.org/10.1109/TGRS.2020.3037361.

Liu, Q., Xiao, L., Yang, J., & Wei, Z. (2022). Multilevel superpixel structured graph U-nets for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–15. http://dx.doi.org/10.1109/TGRS.2021.3112586.

Liu, B., Yu, A., Gao, K., Wang, Y., Yu, X., & Zhang, P. (2022). Multiscale nested U-Net for small sample classification of hyperspectral images. *Journal of Applied Remote Sensing*, *16*(1), Article 016506. http://dx.doi.org/10.1117/1.JRS.16.016506, URL https://doi.org/10.1117/1.JRS.16.016506.

Loshchilov, I. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Mou, L., Lu, X., Li, X., & Zhu, X. X. (2020). Nonlocal graph convolutional networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *58*, 8246–8257.

Munipalle, V. K., Nelakuditi, U. R., CVSS, M. K., & Nidamanuri, R. R. (2024). Ultra-high-resolution hyperspectral imagery datasets for precision agriculture applications. *Data in Brief*, *55*, Article 110649.

Oswald, W., Browning, C., Yasmin, R., Deal, J., Rich, T. C., Leavesley, S. J., et al. (2024). Fluorescence excitation-scanning hyperspectral imaging with scalable 2D–3D deep learning framework for colorectal cancer detection. *Scientific Reports*, *14*(1), 14790.

Qin, A., Shang, Z., Tian, J., Wang, Y., Zhang, T., & Tang, Y. Y. (2019). Spectral–spatial graph convolutional networks for semisupervised hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, *16*(2), 241–245. http://dx.doi.org/10.1109/LGRS.2018.2869563.

Ramakrishnan, D., & Bharti, R. (2015). Hyperspectral remote sensing and geological applications. *Current Science*, 879–891.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention – MICCAI 2015* (pp. 234–241).

Shi, C., Chen, J., & Wang, L. (2024). Hyperspectral image classification based on a novel lush multi-layer feature fusion bias network. *Expert Systems with Applications*, *247*, Article 123155.

Shu, Z., Wang, Y., & Yu, Z. (2024). Dual attention transformer network for hyperspectral image classification. *Engineering Applications of Artificial Intelligence*, *127*, Article 107351.

Sun, L., Zhao, G., Zheng, Y., & Wu, Z. (2022). Spectral–spatial feature tokenization transformer for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–14. http://dx.doi.org/10.1109/TGRS.2022.3144158.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-ResNet and the impact of residual connections on learning. In *Proceedings of the thirty-first AAAI conference on artificial intelligence* (pp. 4278–4284). AAAI Press.

Varma, M. K. S., Rao, N., Raju, K., & Varma, G. (2016). Pixel-based classification using support vector machine classifier. In *2016 IEEE 6th international conference on advanced computing* (pp. 51–55).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Wan, S., Gong, C., Zhong, P., Du, B., Zhang, L., & Yang, J. (2020). Multiscale dynamic graph convolutional network for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *58*(5), 3162–3177. http://dx.doi.org/10.1109/TGRS.2019.2949180.

Wang, W., Dou, S., Jiang, Z., & Sun, L. (2018). A fast dense spectral–spatial convolution network framework for hyperspectral images classification. *Remote Sensing*, *10*(7), http://dx.doi.org/10.3390/rs10071068, URL https://www.mdpi.com/2072-4292/10/7/1068.

Wei, X., Yang, Q., Gong, Y., Ahuja, N., & Yang, M.-H. (2018). Superpixel hierarchy. *IEEE Transactions on Image Processing*, *27*(10), 4838–4849. http://dx.doi.org/10.1109/TIP.2018.2836300.

Xu, L., Ren, J. S., Liu, C., & Jia, J. (2014). Deep convolutional neural network for image deconvolution. *Advances in Neural Information Processing Systems*, *27*.

Yang, Y., Wang, Y., Zhao, E., Song, M., & Zhang, Q. (2023). A swin transformer-based fusion approach for hyperspectral image super-resolution. In *IGARSS 2023-2023 IEEE international geoscience and remote sensing symposium* (pp. 7372–7375). IEEE.

Yang, H., Yu, H., Hong, D., Xu, Z., Wang, Y., & Song, M. (2022). Hyperspectral image classification based on multi-level spectral-spatial transformer network. In *2022 12th workshop on hyperspectral imaging and signal processing: evolution in remote sensing* (pp. 1–4). http://dx.doi.org/10.1109/WHISPERS56178.2022.9955116.

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., & Huang, T. (2019). Free-form image inpainting with gated convolution. In *2019 IEEE/CVF international conference on computer vision* (pp. 4470–4479). http://dx.doi.org/10.1109/ICCV.2019.00457.

Zhang, D., Huang, G., Zhang, Q., Han, J., Han, J., & Yu, Y. (2021). Cross-modality deep feature learning for brain tumor segmentation. *Pattern Recognition*, *110*, Article 107562. http://dx.doi.org/10.1016/j.patcog.2020.107562, URL https://www.sciencedirect.com/science/article/pii/S0031320320303654.

Zhang, Y., Jiang, G., Cai, Z., & Zhou, Y. (2024). Bipartite graph-based projected clustering with local region guidance for hyperspectral imagery. *IEEE Transactions on Multimedia*.

Zhang, J., Liu, J., Yang, J., & Wu, Z. (2023). Crossed dual-branch U-net for hyperspectral image super-resolution. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.

Zhong, Z., Li, J., Luo, Z., & Chapman, M. (2018). Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework. *IEEE Transactions on Geoscience and Remote Sensing*, *56*(2), 847–858. http://dx.doi.org/10.1109/TGRS.2017.2755542.