Chapter 6

Lightweight Framework for Imbalanced HSI Classification

6.1 Introduction

HSI classification faces challenges due to high dimensionality, limited labels, and class imbalance. Despite rich spectral information, efficient utilization remains complex. Traditional DL models struggle with high computational costs and poor generalization, which limits real-world applicability.

Recent approaches have tackled these challenges individually. MDCNN (Chapter 3) captures spectral-spatial features using a vanilla CNN but suffers from high computational costs. LogGroupFormer (Chapter 4) integrates a lightweight CNN but relies on a Transformer with quadratic complexity $\mathcal{O}(N^2d)$, limiting scalability. CKGFLNet (Chapter 5) reduces computational overhead with linear attention but does not explicitly address class imbalance, a key factor in HSI classification robustness. To bridge the gaps mentioned above, this chapter introduces the Hierarchical Kiaming-Gaussian Transformer Network (HieraKGTNet), a lightweight and robust framework designed to simultaneously address the challenges of class imbalance and high computational complexity in HSI classification. The framework integrates several novel architectural modules and learning strategies to achieve superior accuracy, efficient processing, and balanced performance across all classes. The key contributions of this chapter are outlined below:

1. We propose HieraKGTNet, a lightweight framework that integrates local spectral-spatial and global contextual features and delivers superior performance on imbalanced HSI datasets.
2. We introduce the Local-Global Attentive Superpixel Segmentation (LGASS)

module, which combines local- and global-SLIC superpixels to efficiently extract fine-grained spatial and global contextual features.

3. We propose the novel Hierarchical Band Clustering Convolution (HCBConv) module to extract local spectral-spatial hierarchical features. It includes two key components: Hierarchical Band Clustering 3D Convolution (HBC3DConv), which clusters spectral bands, and Hierarchical Channel Clustering 2D Convolution (HCC2DConv), which clusters features along the channel dimension. This dynamic filter distribution enhances spatial feature learning while reducing computational overhead with fewer parameters.
4. We propose the Kaiming Semantic Tokenizer module to encode the most informative spectral cues. These tokenized cues are then processed by a Flattened Kaiming-Gaussian Transformer (FKGT), which reduces computational complexity. FKGT effectively models global contextual features and long-term dependencies while mitigating redundant attention computations.
5. Finally, we have developed the Multiclass Poly-Focal Loss (MPF-Loss) function, which adaptively emphasizes difficult samples and minority classes, improving convergence and classification in imbalanced scenarios.

Extensive experiments on benchmark datasets validate the proposed architecture's ability to achieve exceptional classification performance, particularly for underrepresented classes, all while maintaining a lightweight design that is well-suited for time- and resource-limited environments.

The remainder of this chapter is structured as follows: Section 6.2 provides an overview of related work in HSI classification with a focus on class imbalance and computational efficiency. Section 6.3 defines the problem statement and outlines the associated challenges. Section 6.4 details the proposed methodology, including the design and functioning of the HCBConv, FKGT, and MPF-Loss modules. Section 6.5 presents the experimental setup, implementation details, and result analysis. Finally, Section 6.6 summarizes the chapter and highlights key findings.

6.2 Related Work

6.2.1 Heavy-weighted DNNs

CNN-based methods, such as 2D-CNN and 3D-CNN, are widely utilized for HSI classification. These methods emphasize the extraction of spatial and spectral-spatial features, respectively [79, 113, 181]. Furthermore, patch-based models, like MS-DenseNet [182], enhance multiscale learning; however, they often face challenges related to redundant

computations. In addition, hybrid 3D-2D CNNs [110, 174] and residual networks [183] contribute to improved accuracy by effectively capturing spatial-spectral correlations and addressing issues of class imbalance. Moreover, DHNet [184] integrates both spatial and spectral branches with feature calibration, leading to enhanced complementarity. Nonetheless, it is crucial to note that these approaches remain computationally intensive and often lack global contextual awareness due to their focus on localized feature modeling.

Transformer Network: The authors in [124] and [120] have utilized transformers to obtain spatial contexts through the flattened feature maps derived from 2D or 3D CNNs. Further, [185] have employed multiple transformer encoders to extract spatial and spectral contexts separately after directly partitioning the entire image's features. [186] proposed a dual-context module to enhance feature representation and classification accuracy. Furthermore, [187] proposed the ISSFormer model, which combines self-attention and convolution in a parallel framework to capture global spectral and local spatial features. At the same time, a bi-directional interaction mechanism enhances feature complementarity for improved classification. Despite their effectiveness, conventional transformers have quadratic time complexity $\mathcal{O}(N^2d)$, making them inefficient for large-scale HSI data where $N \gg d$. We introduce an optimized transformer with $\mathcal{O}(Nd^2)$ complexity, preserving spectral-spatial correlations while reducing computational costs. Our model captures local and global dependencies for imbalanced HSI classification by integrating a lightweight CNN framework with our proposed ViT.

6.2.2 Loss functions for Imbalance Sample Issues in HSI dataset

The issue of hard-to-classify training samples due to imbalanced data in HSI classification reduces accuracy for underrepresented categories, especially with Cross-Entropy Loss (CE) [17]. Weighted Cross-Entropy Loss (W-CE) addresses this but favors easy samples, neglecting challenging ones. Focal Loss (FL) [188] mitigates this by emphasizing hard samples, and Cui et al. [189] leveraged FL in a lightweight model to enhance focus on imbalanced data. However, FL may overly prioritize minority classes, impacting overall accuracy. The formulation of FL is as follows:

$$L_{FL} = -\sum_{i=1}^{C} \alpha_i \; Y_i \; (1 - \hat{Y_i})^{\gamma} \; \log(\hat{Y_i}), \tag{6.1}$$

where Y is a one-hot vector for ground truth, and C represents the total classes in HSI data. $\hat{Y_i}$ denotes the predicted probability for i^{th} class. The hyperparameter $\alpha_i \in [0, 1]$

is the balanced weight for class i, while γ is the focusing parameter in M-FL. M-FL enhances classification for underrepresented classes but may overemphasize minority samples, reducing accuracy for majority classes. To tackle this issue, [190] introduced the concept of poly-focal loss. However, it is worth mentioning that polyfocal loss is primarily designed for binary classification scenarios and may not directly apply to multiclass problems. Thus, we proposed a novel MPF-Loss in this paper to enhance the overall average classification.

6.2.3 Superpixel Segmentation

In recent image classification research, the use of superpixel segmentation algorithm [191] has increased. Utilizing the Simple Linear Iterative Clustering (SLIC) algorithm is an effective method in this field, particularly for HSI classification tasks. The SLIC employs the over-segmentation technique to detect superpixels, with the primary objective of utilizing the K-means algorithm [192] to achieve efficient local clustering and segmentation of superpixels. The SLIC algorithm extracts spatial texture information from HSIs. To be more precise, SLIC uses a $\mathbf{2\widehat{S}} \times \mathbf{2\widehat{S}}$ block to determine the distance from the center of every cluster to the pixel, where $\widehat{\mathbf{S}} = \sqrt{\widehat{\mathbf{N}}/\mathbf{C}'}$, where $\widehat{\mathbf{N}}$ represents the total number of pixels and $\mathbf{C}'$ represents the number of superpixels.

6.3 Problem Statement

Given a HSI cube $\mathbf{I} \in \mathbb{R}^{[B \times H \times W]}$, where B, H, and W denote the number of spectral bands, spatial height, and width respectively, the aim is to develop a lightweight and robust classification framework that addresses two key challenges: (i) severe class imbalance among minority and majority categories, and (ii) high computational overhead that limits deployment scalability and real-time applicability.

To tackle these issues, the proposed HieraKGTNet framework comprises a sequence of learnable transformation functions. First, the Local-Global Attentive Superpixel Segmentation module $F_{\text{LGASS}} : \mathbf{I} \rightarrow \mathbf{I}_{\text{seg}} \in \mathbb{R}^{[B \times H \times W]}$ captures fine-grained spatial boundaries and global contextual priors through hierarchical SLIC superpixel clustering. Subsequently, the hierarchical cluster-based convolutional operator $F_{\text{HCBConv}} : \mathbf{I}_{\text{seg}} \rightarrow \mathbf{F}_{\text{HCB}} \in \mathbb{R}^{[\mathfrak{K} \times H \times W]}$ fuses both HBC3DConv and HCC2DConv operations to extract enriched spectral-spatial features.

The resulting feature map is tokenized using the Kaiming Semantic Tokenizer $F_{\text{KST}} : \mathbb{R}^{[\mathfrak{K} \times H \times W]} \rightarrow \mathbb{R}^{[N \times d]}$, which compresses spatial-spectral representations into semantically meaningful token embeddings. These tokens are then processed by the Flattened

Kaiming-Gaussian Transformer $F_{\text{FKGT}} : \mathbb{R}^{[N\times d]} \to \mathbb{R}^{[N\times d]}$, a linear-attention mechanism designed to capture long-range dependencies with reduced computational cost.
Finally, the transformed sequence is passed through the classification head $F_{\text{classify}} : \mathbb{R}^{[N\times d]} \to \mathbb{R}^{[C]}$, trained using a Multiclass Poly-Focal Loss $\mathcal{L}_{\text{MPF}}$ to improve discrimination among underrepresented classes and enhance generalization under class imbalance. The primary objective is to maximize class-balanced accuracy while minimizing inference complexity and performance degradation due to class imbalance. The optimization problem can be formally stated as:

$$\begin{aligned}\underset{F_{\text{LGASS}},F_{\text{HCBConv}},F_{\text{KST}},F_{\text{FKGT}},F_{\text{classify}}}{\text{maximize}} \quad & \hat{\mathcal{C}}_{\text{bal}}\left(F_{\text{classify}}(F_{\text{FKGT}}(F_{\text{KST}}(F_{\text{HCBConv}}(F_{\text{LGASS}}(\mathbf{I})))))\right) \\ & - \lambda_1 \cdot \hat{\mathcal{L}}_{\text{comp}}(\mathbf{I}) - \lambda_2 \cdot \hat{\mathcal{L}}_{\text{imb}}(I),\end{aligned} \tag{6.2}$$

where λ_1 and λ_2 are regularization weights controlling the trade-off between classification performance, computational efficiency, and class balance across the hyperspectral scene I.

6.4 Proposed Methodology

This section introduces a novel DL architecture designed for imbalanced HSI classification. Our proposed architecture is named HieraKGTNet, which is a hierarchical cluster-based convolution with a flattened kaiming-gaussian transformer network. As illustrated in Figure 6.1, the architecture consists of five core modules: (a) Local-Global Attentive Superpixel Segmentation (LGASS), (b) Hierarchical Cluster-Based 3D–2D Convolution (HCBConv), (c) Kaiming Semantic Tokenizer, (d) Flattened Kaiming-Gaussian Transformer (FKGT), and (e) a classifier. For robust optimization, a novel Multiclass Poly-Focal Loss (MPF-Loss) is employed.
In HieraKGTNet, feature selection is performed in two stages. First, LGASS generates spatially coherent regions that enhance local detail while preserving global context. Next, the HCBConv refines these features through HBC3DConv for joint spectral–spatial extraction and HCC2DConv for spatial refinement. The resulting hierarchical maps are then passed through the Kaiming Semantic Tokenizer, which compresses them into compact semantic tokens representing the most informative spectral–spatial cues. These selected features are subsequently used by the Transformer for global dependency modeling. The methodology can be summarized in the following steps:

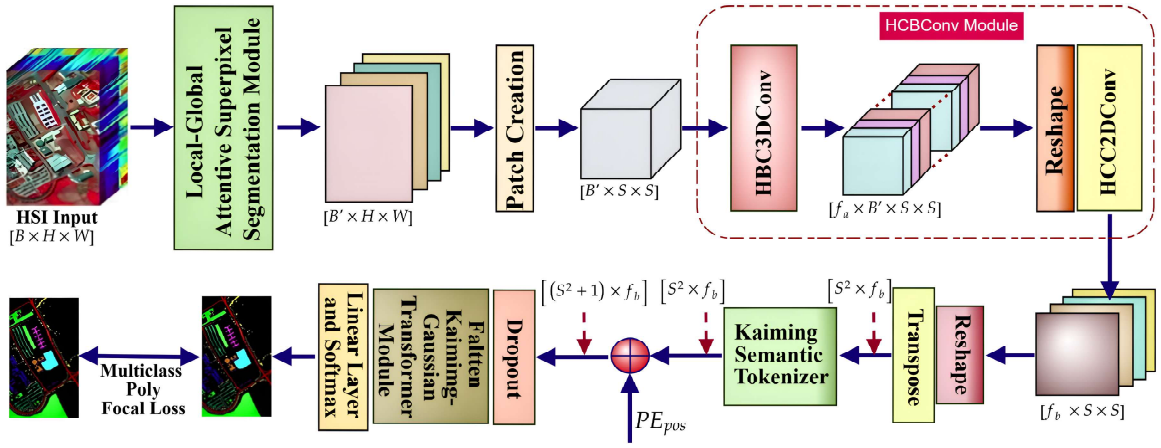


Figure 6.1: Illustration of HieraKGTNet framework for imbalanced HSI classification. It comprises five modules: (a) Local-Global Attentive Superpixel Segmentation (LGASS), (b) Hierarchical Cluster-Based Convolution (HCBConv) consisting of Hierarchical-Band Clustering 3D Convolution (HBC3DConv) and Hierarchical-Channel Clustering 2D Convolution (HCC2DConv), (c) Kaiming Semantic Tokenizer, (d) Flattened Kaiming-Gaussian Transformer (FKGT), and (e) a classifier.

1. **HSI Input Representation:** Represent HSI as $I \in \mathbb{R}^{[B \times H \times W]}$, where B, H, and W denote the number of spectral bands, height, and width, respectively.
2. **LGASS:** Apply LGASS to the input HSI to generate fine-grained spatially segmented regions. This step enhances local discriminative features while preserving global contextual information.
3. **HCBConv:** Extract local spectral–spatial hierarchical features using two components: (a) Hierarchical-Band Clustering 3D Convolution (HBC3DConv), and (b) Hierarchical-Channel Clustering 2D Convolution (HCC2DConv). This module enables multi-level feature extraction across spectral and spatial dimensions.
4. **Kaiming Semantic Tokenizer:** Convert hierarchical feature maps into compact semantic tokens. The tokenizer captures the most informative spectral–spatial cues while maintaining computational efficiency.
5. **FKGT:** Model long-range global spectral dependencies using the FKGT. This module reduces the quadratic complexity of conventional Transformers from $\mathcal{O}(N^2 d)$ to $\mathcal{O}(N d^2)$, thereby improving scalability.
6. **Classification with MPF-Loss:** Feed refined feature representations into a Softmax classifier for predicting class labels. The predictions are optimized using the proposed MPF-Loss, which addresses class imbalance by emphasizing hard-to-classify and minority samples.

6.4.1 Local-Global Attentive Superpixel Segmentation (LGASS)

In this subsection, we leverage superpixel segmentation to enhance spatial feature representation in HSI classification, following the insights from [193]. Specifically, we integrate superpixel-based structural priors into the proposed HieraKGTNet framework to better preserve local spatial coherence and boundary information. However, relying solely on a single-scale superpixel segmentation may lead to incomplete feature representation, as it captures fine-grained local patterns but often overlooks essential global contextual dependencies. This limitation can result in misclassification, particularly in regions where local textures are ambiguous or spectrally similar across different classes.

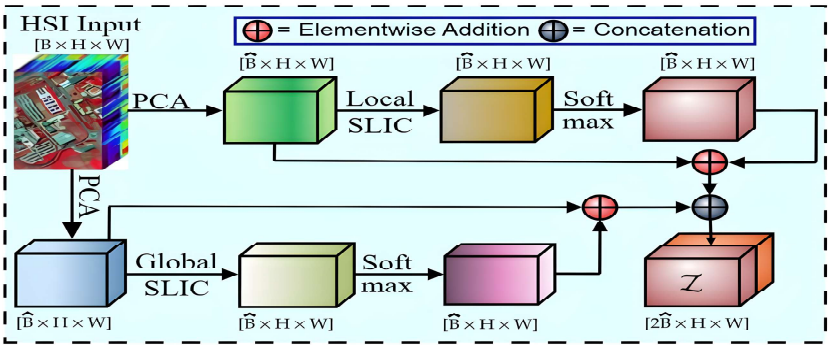


Figure 6.2: LGASS incorporates two superpixel segmentation methods: local- and global-SLIC. These methods are utilized to extract both fine-scale local spatial and broader global contextual cues, enhancing the effectiveness of HSI classification.

To overcome the aforementioned limitation, we introduce a parallel fusion approach combining Local SLIC (fine-scale segmentation) and Global SLIC (coarse-scale segmentation) with an attention mechanism, forming the **LGASS** (Local-Global Attentive Superpixel Segmentation) module. As shown in Figure 6.2, this module effectively integrates local fine-grained textures with global structural priors. The multi-scale fusion strategy significantly enhances spatial-spectral feature representation, thereby improving classification performance [194, 195]. In the LGASS module, the input HSI cube $\mathbf{I} \in \mathbb{R}^{[B \times H \times W]}$ undergoes PCA for spectral dimensionality reduction, yielding two projections $\mathfrak{P}_l, \mathfrak{P}_g \in \mathbb{R}^{[\hat{B} \times H \times W]}$, where $\hat{B} \ll B$. The local-SLIC operation is applied to $\mathfrak{P}_l$ using 500 segments and compactness of 1 to extract fine-scale spatial features. In parallel, the global-SLIC is applied to $\mathfrak{P}_g$ with 300 segments and a compactness of 10 to capture global contextual information. Both segmentations are followed by a softmax activation and are added element-wise to their corresponding PCA outputs to preserve local and global spectral-spatial information. The enhanced outputs are then

concatenated along the spectral dimension, resulting in $\mathcal{Z} \in \mathbb{R}^{[2\hat{B}\times H\times W]}$. The entire process is formulated as:

$$\mathcal{Z} = [Softmax(\mathcal{S}_l(PCA(\mathbf{I}))) + PCA(\mathbf{I})] \oplus [Softmax(\mathcal{S}_g(PCA(\mathbf{I}))) + PCA(\mathbf{I})], \quad (6.3)$$

where $\mathcal{S}_l$ and $\mathcal{S}_g$ represent local and global SLIC functions respectively and $\oplus$ denotes concatenation along the spectral dimension.

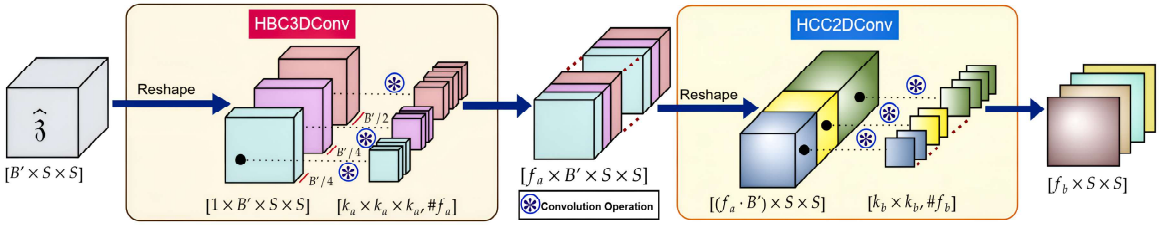


Figure 6.3: Illustration of the HCBConv, consisting of HBC3DConv and HCC2DConv, to extract correlated spectral-spatial local hierarchical features.

6.4.2 Hierarchical-Cluster-Based 3D-2D Convolution Module(HCBConv)

In this subsection, we present the novel Hierarchical Band-Clustered Convolutional Block (HCBConv) to capture fine-grained spectral-spatial features. HCBConv not only enhances feature extraction but also significantly reduces the computational overhead in HSI processing. It consists of three primary sub-modules: Patch Extraction, HBC3DConv, and HCC2DConv, as shown in Figure 6.3.

Algorithm 6.1: Patch Extraction

Input : HSI tensor $\mathcal{Z} \in \mathbb{R}^{[B'\times H\times W]}$. (H, W): Height and width of I_{patch}. $B' = 2\hat{B}$: Number of spectral bands/channels.
Output: Patch tensor $I_{patch} \in \mathbb{R}^{[1\times B'\times S\times S]}$.

Patch Extraction Process:
Initialization: $I_{patch} = \{\}$
for $i = 0$ **to** $H - 1$ **do**
  for $j = 0$ **to** $W - 1$ **do**
    $I_{patch}[i, j] \leftarrow \mathcal{Z}[:, i - \frac{S}{2} : i + \frac{S}{2}, j - \frac{S}{2} : j + \frac{S}{2}]$
    $I_{patch}[i, j] \leftarrow \text{Reshape}(I_{patch}, [1, B', S, S])$
Return: I_{patch}

Firstly, we perform a patch extraction step to extract the hierarchical HSI features. The in-depth procedure is detailed in Algorithm 6.1. Secondly, we fed the I_{patch} as an input to the HBC3DConv sub-module to enhance spectral-spatial feature extraction. The detailed procedure of extracting enhanced features while simultaneously reducing

Algorithm 6.2: HBC3DConv Sub-Module

Input : Patch tensor $I_{patch} \in \mathbb{R}^{[1\times B'\times S\times S]}$, Kernel size $K_a \times K_a \times K_a$, Number of filters f_a
Output: Feature map $Y_{HBC3D} \in \mathbb{R}^{[f_a\times B'\times S\times S]}$

Partition: Spectral Bands and f_a in $g = 3$ clusters:
  $B'_i \leftarrow [B'/2, B'/4, B'/4]$, $f_{a_i} \leftarrow [f_a/2, f_a/4, f_a/4]$,
Initialize empty feature map list: $\mathcal{Y} = \{\}$
Apply 3D Convolution to each cluster :
for $i = 1$ **to** g **do**
  Extract Cluster: $Z_i \leftarrow I_{patch}[:, B'_i, :, :]$
  Apply Conv: $Y_i \leftarrow \text{3DConv}(Z_i, K_a, f_{a_i})$
  Store Output: $\mathcal{Y} \leftarrow Y_i$
Concatenate Along Spectral Dimension
  $Y_{HBC3D} \leftarrow \text{Concatenate}(\mathcal{Y}, \text{dim} = 1)$
Return: Y_{HBC3D}

the complexity overhead is outlined in Algorithm 6.2. A key feature of this algorithm is the clustered convolution strategy, which partitions the spectral bands B' and filters f_a into three clusters: $[B'/2, B'/4, B'/4]$ and $[f_a/2, f_a/4, f_a/4]$, respectively. Finally, the obtained feature map Y_{HBC3D} is processed by the HCC2DConv sub-module to refine spatial feature extraction while reducing computational complexity, as shown in Algorithm 6.3. This sub-algorithm employs a similar partitioning strategy, but this time applied to channel bands.

Algorithm 6.3: HCC2DConv Sub-Module

Input : Feature map $Y_{HBC3D} \in \mathbb{R}^{[f_a\times B'\times S\times S]}$
Output: Feature map $Y_{HCC2D} \in \mathbb{R}^{[f_b\times S\times S]}$

Reshape: $Y'_{HBC3D} = \text{Reshape}(Y_{HBC3D})$, where $Y'_{HBC3D} \in \mathbb{R}^{[(f_a * B')\times S\times S]}$
Partition: channels and f_b into $g = 3$ clusters:
  $B'' = (f_a * B')$
  $B''_i \leftarrow [B''/2, B''/4, B''/4]$, $f_{b_i} \leftarrow [f_b/2, f_b/4, f_b/4]$
Apply 2D Convolution to each cluster:
for $i = 1$ **to** g **do**
  Extract Cluster: $Z_i \leftarrow Y'_{HBC3D}[B''_i, :, :]$
  Apply Conv: $Y_i \leftarrow \text{2DConv}(Z_i, K_f, f_{b_i})$
  Store Output: $\mathcal{Y} \leftarrow Y_i$
Concatenate Along Channel Dimension
  $Y_{HCC2D} \leftarrow \text{Concatenate}(\mathcal{Y}, \text{dim} = 1)$
Return: Y_{HCC2D}

In other words, we optimize spatial feature representation with minimal overhead. We justify the reduction in computational cost through the analysis given in Table 6.1. It highlights the efficiency gains of the HCBConv module compared to conventional 3D and 2D convolution approaches. The HBC3DConv sub-module maintains the same parameter count as conventional 3D convolution $\Delta P_{HBC3D} = 0$ but significantly re-

duces computational overhead by 62.5% ($\Delta\mathcal{C}_{HBC3D} = (5/8)K_a^3.B'.f_a.S^2$). Similarly, the HCC2DConv sub-module achieves a 62.5% reduction in computational overhead while reducing parameter count by the same proportion. These reductions stem from the hierarchically clustered convolution strategy, which optimally partitions spectral and channel dimensions, ensuring efficient feature extraction while minimizing computational burden.

Table 6.1: Computation cost analysis in the HCBConv module, including a comparison of the number of parameters ($\mathbf{P_M}$) and computational overheads ($\mathbf{C_{FLOPs}}$).

Sub-Module	Cost	Conventional	Proposed	Difference
HCB3DConv	$\mathbf{P_M}$	$\mathcal{P}_{3D} = K_a^3.f_a + f_a$	$\mathcal{P}_{HBC3D} = K_a^3.f_a + f_a$	$\Delta P_{HBC3D} = 0$
	$\mathbf{C_{FLOPs}}$	$\mathcal{C}_{3D} = K_a^3.B'.f_a.S^2$	$\mathcal{C}_{HBC3D} = (3/8)K_a^3.B'.f_a.S^2$	$\Delta\mathcal{C}_{HBC3D} = (5/8)K_a^3.B'.f_a.S^2$
HCC2DConv	$\mathbf{P_M}$	$\mathcal{P}_{2D} = K_b^2.B''.f_b + f_b$	$\mathcal{P}_{HCC2D} = (3/8)K_b^2.B''.f_b + f_b$	$\Delta\mathcal{P}_{HCC2D} = (3/8)K_b^2.B''.f_b + f_b$
	$\mathbf{C_{FLOPs}}$	$\mathcal{C}_{2D} = K_b^2.B''.f_b.S^2$	$\mathcal{C}_{HCC2D} = (3/8)K_b^2.B''.f_b.S^2$	$\Delta\mathcal{C}_{HCC2D} = (5/8)K_b^2.B''.f_b.S^2$

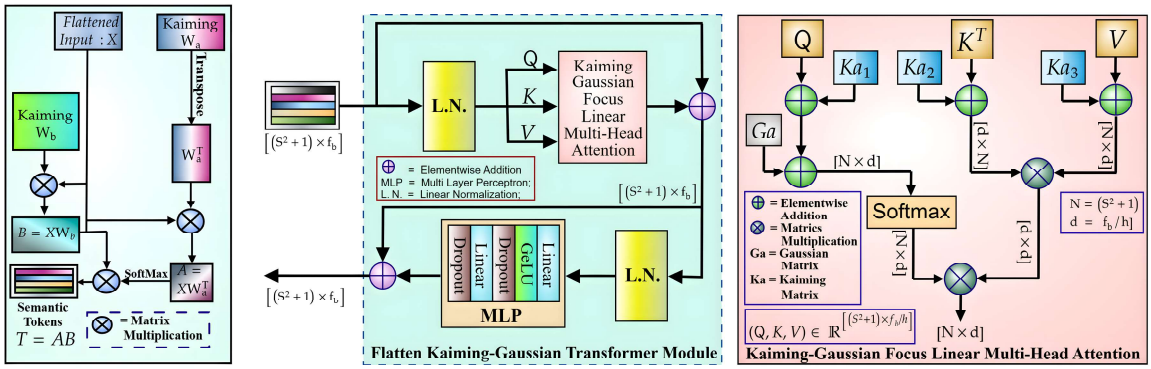
6.4.3 Kaiming Tokenizer

In HSI data analysis, features extracted through convolutional layers encode both spectral and spatial information, yet they often fall short of fully capturing the complex characteristics of ground objects. To address this, semantic tokenization redefines feature maps into high-level representations [124]. We propose Kaiming Tokenization, leveraging Kaiming initialization for enhanced feature extraction. Unlike Gaussian initialization, Kaiming prevents gradient vanishing/explosion and stabilizes deep networks, particularly with ReLU. It accelerates convergence by aligning initialization with non-linearities, reducing hyperparameter tuning efforts. Before tokenization, tensor $\mathbf{Y}_{HCC2D}$ is reshaped and transposed into $X \in \mathbb{R}^{[S^2 \times f_b]}$, where S is spatial size and f_b is channel count.

Kaiming Tokenization then refines features using weight matrices $W_a \in \mathbb{R}^{[S^2 \times f_b]}$ and $W_b \in \mathbb{R}^{[f_b \times f_b]}$, both initialized via Kaiming. The formulation for generating semantic tokens is expressed as:

$$T = Softmax(XW_a^T)(XW_b) \tag{6.4}$$

where $A = XW_a^T$ maps feature to a latent space, reducing dimensionality. The output is multiplied by $B = XW_b$, yielding $T \in \mathbb{R}^{[h \times S^2 \times f_b]}$. Figure 6.4 (a) illustrates this transformation. Kaiming Tokenization improves semantic feature extraction, enhances training stability, and aligns efficiently with network activations, ensuring robust and efficient HSI representation.



(a) (b) (c)

Figure 6.4: Illustration of (a) Kaiming Tokenizer, (b) FKGT Module, and (c) KGFL-MHSA Module for HSI classification.

6.4.4 Flattened Kaiming-Gaussian Transformer (FKGT) Module

In this module, we design a Flattened Kaiming-Gaussian Transformer (FKGT) to extract spectral-spatial features. It integrates Kaiming Gaussian Focus Linear Multi-Head Attention (KGFL-MHA) to enhance feature extraction while significantly reducing computational complexity, lowering the time complexity of attention mechanism from $\mathcal{O}(N^2d)$ to $\mathcal{O}(Nd^2)$, where $N = (S^2 + 1)$ denotes the token count and $d = f_b/h$ represents the feature dimension per attention head. FKGT processes the semantic input token T has an initial shape of $[(S^2 + 1) \times f_b]$, where S^2 represents the number of spatial patches and the additional token accounts for global representation. The entire procedure for extracting the key features unfolded in the detailed steps outlined in Algorithm 6.4. To visualize the architecture and flow of the module, one can refer to the diagrams in Figure 6.4 (b) and 6.4 (c).

Unlike conventional transformers, where attention is computed as $Z = QK^T$ followed by ZV, FKGT first multiplies the transposed key matrix with the value matrix to obtain $Z = K'V'$, reducing intermediate computations. This step has a computational complexity of $O(N^2d)$, as shown in Table 6.2. Then, the attention mechanism multiplies the refined query matrix $\hat{Q}$ with Z, leading to a final complexity of $O(Nd^2)$, making it more efficient for large-scale data. Additionally, stability is reinforced using residual connections with normalization, ensuring smooth gradient propagation. The processed features then pass through a feed-forward MLP block, enhancing representations through nonlinear transformations and dropout. Moving forward, a FC layer maps extracted features to class probabilities, followed by MPF-Loss, optimizing for class imbalances. The final output is the predicted class probabilities $\hat{Y}$.

Algorithm 6.4: FKGT Module

Input: Token Matrix $T \in \mathbb{R}^{[(S^2+1)\times f_b]}$
Output: Predicted Class Probabilities $\hat{Y}$
Step 1: $T' \leftarrow LayerNormalization(T)$
Step 2: MHSA with Kaiming-Gaussian Mechanism
  a. Compute: $Q \leftarrow T'W_q, \quad K \leftarrow T'W_k, \quad V \leftarrow T'W_v$
  b. Apply Kaiming Gaussian Focus:
  $Q' \leftarrow Q + Ka(X_q)$
  $K' \leftarrow K^T + Ka(X_k)$
  $V' \leftarrow V + Ka(X_v)$
  c. Add Gaussian Noise: $\hat{Q} \leftarrow Softmax(Q' + Ga(W_g))$
  d. Compute Attention: $Z \leftarrow K'V'$, $Att(\hat{Q}, K', V') \leftarrow \hat{Q}Z$
  e. Multi-Head Attention:
  $MHSA_{output} \leftarrow$ **Concat**$[Head_1, ..., Head_h]W_O$
Step 3: $\hat{O} \leftarrow LayerNormalization(MHSA_{output} + T)$
Step 4: Feed-Forward Network (MLP Block)
  $F_1 \leftarrow Dropout(GeLU(\hat{O}W_1 + b_1))$
  $F_2 \leftarrow Dropout(F_1W_2 + b_2)$
  $Final\_Feature \leftarrow F_2 + \hat{O}$
Step 5: $O_{fc} \leftarrow Final\_FeatureW_f + b_f$, $\hat{Y} \leftarrow Softmax(O_{fc})$
Compute MPF-Loss and update model parameters.
Return: Predicted Class Probabilities $\hat{Y}$

Table 6.2: Time Complexity Analysis: Conventional vs FKGT Module.

Transformer with Conventional Multi-Head Self-Attention			FKGT with KGFL-MultiHead Attention		
Step	**Matrix Sizes**	**Computational Cost**	**Step**	**Matrix Sizes**	**Computational Cost**
QK^T	$Q : [N \times d], K^T : [d \times N]$	N^2d	$K'V'$	$K' : [d \times N], V' : [N \times d]$	Nd^2
Resultant Size	$Z : [N \times N]$		Resultant Size	$Z : [d \times d]$	
ZV	$Z : [N \times N], V : [N \times d]$	N^2d	$\hat{Q}Z$	$\hat{Q} : [N \times d], Z : [d \times d]$	Nd^2
Resultant Size	$Op : [N \times d]$		Resultant Size	$Op : [N \times d]$	
Total Complexity: N^2d			**Total Complexity:** Nd^2		

6.4.5 Multiclass Poly-focal Loss

Class imbalance is prevalent in the HSI classification, where certain classes dominate while others remain underrepresented. In multiclass classification problems, traditional Cross-Entropy (CE) Loss often struggles with imbalanced data, as it assigns equal importance to all classes regardless of their frequency. Addressing this challenge, we introduce the novel Multiclass Poly-Focal Loss (MPF-Loss), an enhancement of the poly-focal loss [190]. Unlike its predecessor, MPF-Loss is designed to handle class imbalance more effectively across multiple categories in HSI data. MPF-Loss builds upon the focal loss framework by incorporating polynomial perturbations to balance the contributions of different class samples. It ensures improved gradient flow and robust learning dynamics, particularly for underrepresented classes. The loss function

Algorithm 6.5: Multiclass Poly-Focal Loss (MPF-Loss)

Input: **Logits**: $\hat{Y}$ (Predicted probabilities), **Target**: Y (Ground truth labels), **Weight vector** (optional), **Gamma** γ
Output: Return optimized weight value

Initialize randomly *Weight* and total loss $L_{MPF} \leftarrow 0$
Set numerical stability constant $\epsilon \leftarrow 1e-9$
for *each sample i in batch* **do**
    Compute cross-entropy loss: $L_{ce} \leftarrow -Y_i \log(\hat{Y}i + \epsilon)$
    Compute probability estimate: $p_t \leftarrow e^{-Lce}$
    if *Weight is not None* **then**
        Compute focal loss: $L_{focal} \leftarrow Weight[Y_i] \cdot (1-p_t)^{\gamma} \cdot L_{ce}$
    else
        Compute focal loss: $L_{focal} \leftarrow (1-p_t)^{\gamma} \cdot L_{ce}$
    Apply polynomial perturbation: $L_{focal} \leftarrow L_{focal} + \epsilon \cdot (1-p_t)^{\gamma+1}$
    Accumulate loss: $L_{MPF} \leftarrow L_{MPF} + L_{focal}$
if *Weight is not None* **then**
    Normalize loss: $L_{MFL} \leftarrow L_{MFL} / \sum Weight$
else
    Normalize loss: $L_{MFL} \leftarrow L_{MFL}$/number of input
return Final MPF-Loss value: L_{MPF}

dynamically adjusts the weighting of easy and hard samples, preventing minority class suppression and promoting model generalization. The procedure for calculating the loss is demonstrated in Algorithm 6.5. The essential formulation is as follows: **Focal Loss Formulation:** The standard focal loss (FL) is given in Equation 6.1.
Perturbation Adjustment: To adaptively manage class imbalance, we introduce a polynomial perturbation term, modifying FL into MPF-Loss:

$$L_{MPF} = L_{FL} + \sum_{i=1}^{C} \sum_{j=1}^{N} \epsilon_j (1 - \hat{Y}_i)^j. \tag{6.5}$$

First-Order Polynomial Approximation: Empirical analysis suggests that adjusting only the first-order term leads to significant performance gains:

$$L_{MPF-1} = L_{FL} + \sum_{i=1}^{C} \epsilon_1 (1 - \hat{Y}_i). \tag{6.6}$$

6.5 Experiments: Implementations and Results

This section outlines the experimental setup and evaluation process for our proposed method. We describe the configurations of publicly available datasets, evaluation metrics, neural network settings, model parameter analysis, ablation studies, and compar-

isons with SOTA methods. Specifically, we apply our method and SOTA techniques to four HSI datasets, providing a comprehensive performance comparison. Classification effectiveness has been measured through OA, AA, and κ scores, complemented by qualitative insights from classification map visualizations. Computational complexity has been analyzed in terms of P_M and C_{FLOPs} computational overheads, which are calculated in terms of FLOPs. At the same time, efficiency has been assessed by recording Tr and Te. These metrics help provide a complete and fair comparison of model performance across datasets. This evaluation demonstrates the efficacy and robustness of our approach across various scenarios.

6.5.1 Dataset Description and Training Details

We employed four widely recognized HSI datasets to evaluate the performance of our proposed architecture and benchmark it against other SOTA techniques. These datasets include IP, PU, HU, and LK. Section 1.3 and Table 1.1 have summarized their sensor specifications, wavelengths, spatial sizes, spectral bands, and classes. Table 6.3 has presented the training, and testing sample distribution, class names, and color codes, ensuring a rigorous evaluation of generalizability and efficiency of our proposed model. We have employed varying proportions of labeled samples for training for each dataset: 1% for PU and LK, 10% for IP, and 2% for HU, with the remaining for testing.

Table 6.3: Descriptions for IP, PU, HU, and LK, with color-code, class name, and the number of training, and test samples.

	Indian Pines (IP)				Pavia University (PU)				Houston University (HU)				WHU-Hi-LongKou (LK)			
No.	Clr	Class Name	Train	Test	Clr	Class Name	Train	Test	Clr	Class Name	Train	Test	Clr	Class Name	Train	Test
C01		Alfalfa	2	44		Asphalt	66	6565		Healthy Grass	25	1226		Corn	345	34166
C02		Corn-NT	71	1357		Meadows	186	18463		Stressed Grass	25	1229		Cotton	83	8291
C03		Corn-MT	41	789		Gravel	20	2079		Synthetic Grass	13	684		Sesame	30	3001
C04		Corn	11	226		Trees	30	3034		Trees	24	1220		BL-Soybean	632	62580
C05		Grass-P	24	458		PM-Sheets	13	1332		Soil	24	1218		NL-Soybean	41	4110
C06		Grass-T	36	694		Bare Soil	50	4979		Water	6	319		Rice	118	11736
C07		Grass-PM	1	27		Bitumen	13	1317		Residential	25	1243		Water	670	66386
C08		Hay-W	23	454		SB-Bricks	36	3646		Commercial	24	1220		Road-Houses	71	7053
C09		Oats	1	19		Shadows	9	938		Road	25	1227		Mixed Weed	52	5179
C10		Soybean-NT	48	924	-	-	-	-		Highway	24	1203	-	-	-	-
C11		Soybean-MT	122	2334	-	-	-	-		Railway	24	1211	-	-	-	-
C12		Soybean-C	29	563	-	-	-	-		Parking Lot 1	24	1209	-	-	-	-
C13		Wheat	10	196	-	-	-	-		Parking Lot 2	9	460	-	-	-	-
C14		Woods	63	1203	-	-	-	-		Tennis Court	8	420	-	-	-	-
C15		Buildings-GTD	19	367	-	-	-	-		Running Track	13	647	-	-	-	-
C16		SS-Towers	4	9	-	-	-	-	-	-	-	-	-	-	-	-

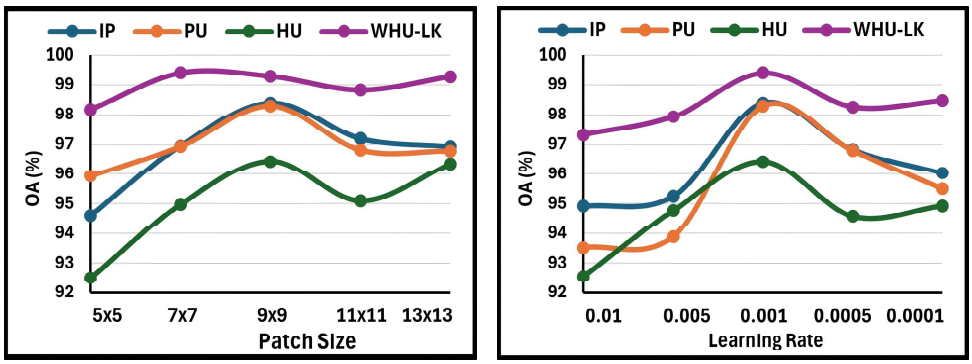
6.5.2 Experimental Settings and Implementation Details

We have conducted ten independent runs for each algorithm to mitigate random initialization effects, reporting the mean performance with standard deviation. Classification

performance has been evaluated using OA, AA, and $\kappa \times 100$, while computational efficiency has been analyzed based on parameter count (in millions M), computational overhead in terms of FLOPs (Giga Flops G), and training and testing times. All experiments have been conducted on a Jupyter Notebook with a 2.20 GHz Intel Xeon CPU, an NVIDIA RTX A5000 GPU with 24 GB memory, and 64 GB RAM. The implementation has been carried out in Python 3.6 with PyTorch, and the input patch sizes have been $[9 \times 9]$ for IP, PU, and HK, while LK has used $[7 \times 7]$. The batch size, learning rate, and training epochs have been set to 128, 0.001, and 70, respectively, with the Adam optimizer for parameter updates.

6.5.3 Hyperparameter Sensitivity Analysis

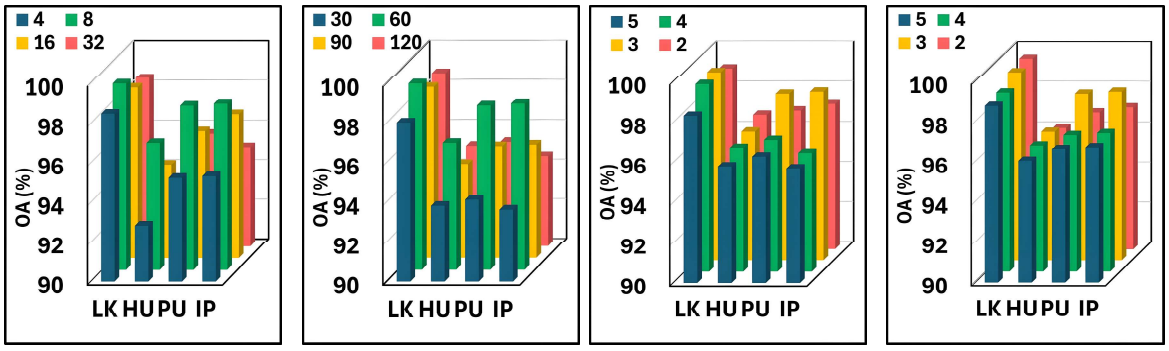
We have analyzed the impact of four key hyperparameters on classification performance: patch size, learning rate (LR), the number of filters in HBC3DConv and HCC2DConv layers, and the number of clusters in HCBConv layers. Our experiments on four datasets evaluate how these parameters affect OA and computational efficiency.



(a) (b)

Figure 6.5: Comparison of OA (%) for IP, PU, HU, and LK datasets across varying (a) Patch Sizes and (b) LR.

Patch Size: We analyzed the effect of input patch size on model performance using five spatial patch sizes: $[5 \times 5], [7 \times 7], [9 \times 9], [11 \times 11]$, and $[13 \times 13]$. As shown in Figure 6.5, the optimal patch size is $[9 \times 9]$ for IP, PU, and HU datasets, effectively capturing broader context, while LK benefits from $[7 \times 7]$ due to its finer spatial details. These results underscore the importance of patch size in enhancing OA across datasets.
Learning Rate: We evaluated the effect of different learning rates (LR) on model performance, testing values of 0.01, 0.005, 0.001, 0.0005, and 0.0001. As shown in Figure 6.5(b), LR = 0.001 yields the highest OA across all datasets, ensuring stable and efficient convergence.



(a) (b) (c) (d)

Figure 6.6: Comparison of OA (%) across the IP, PU, HU, and LK datasets for a varying number of (a) filters in the HBC3DConv layer, (b) filters in the HCC2DConv layer, (c) clusters in the HBC3DConv layer, and (d) clusters in the HCC2DConv layer.

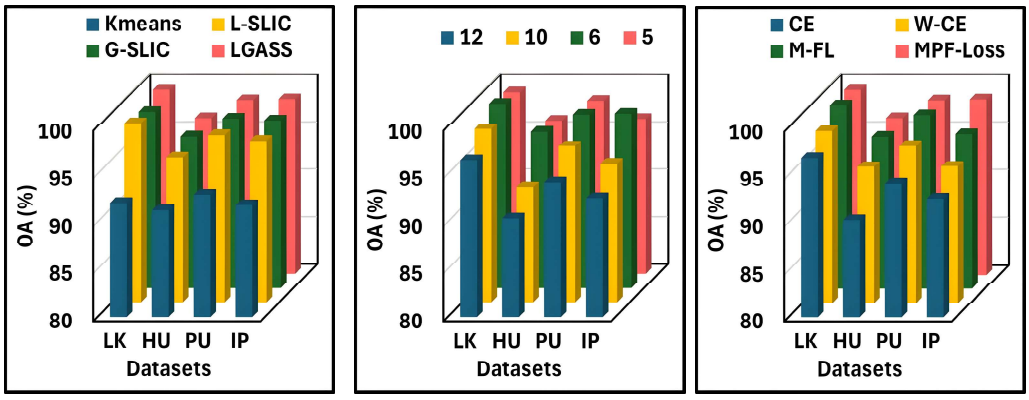
Number of filters in HBC3DConv and HCC2DConv Layers: We have evaluated the effect of filter numbers in HBC3DConv and HCC2DConv layers. 4, 8, 16, and 32 kernels have been taken for HBC3DConv, and 30, 60, 90, and 120 filters for HCC2DConv. As shown in Figure 6.6(a), the best OA is achieved with 8 filters for HBC3DConv across all datasets, while Figure 6.6(b) shows that HCC2DConv performs optimally with 60 filters. These findings highlight the importance of filter selection in balancing OA and computational efficiency.

Number of Cluster Divisions for HCBConv Layers: We analyzed the impact of hierarchical clustering divisions in HBC3DConv and HCC2DConv layers by testing 2, 3, 4, and 5 clusters. As shown in Figure 6.6(c)-(d), three clusters generally achieve the highest accuracy for LK, PU, and IP datasets, while HU peaks at two clusters. The LK dataset attains the best OA of 99.39% with three clusters, remaining competitive with four clusters (99.32%). Three clusters likely enhance feature discrimination by balancing intra-cluster similarity and inter-cluster diversity. Beyond three clusters, reducing parameters may cause a loss of OA due to excessive feature fragmentation.

6.5.4 Ablation Studies

This subsection analyzes key components of our proposed model to determine their individual contributions. We investigate (a) the effectiveness of various superpixel segmentation strategies, (b) the impact of different numbers of attention heads (h) in the FKGT module, and (c) the effect of various loss functions on classification performance.

Superpixel Segmentation Strategy: We have evaluated four segmentation strate-



(a) (b) (c)

Figure 6.7: Ablation study of model components on IP, PU, HU, and LK datasets: OA (%) comparison for (a) superpixel segmentation, (b) heads, and (c) loss functions.

gies: (i) Global-SLIC, (ii) K-means, (iii) Local-SLIC, and (iv) LGASS. As shown in Figure 6.7 (a), LGASS, a fusion of Local-SLIC and Global-SLIC, has achieved superior performance. K-means has performed the worst due to its reliance on global relationships, while Local-SLIC, focusing on local features, has outperformed both K-means and Global-SLIC. LGASS has effectively balanced local and global spatial relationships which enhanced classification accuracy.

Selection of Number of Heads: We have examined the impact of varying head counts (h) in the FKGT module for a sequence length of 60, testing 5, 6, 10, and 12 heads. As shown in Figure 6.7 (b), setting the number of attention heads to $h = 6$ consistently yields the highest OA across all datasets. In contrast, configurations with $h = 5$, 10, and 12 produce competitive results, but increasing the number of heads beyond six does not consistently improve performance and introduces additional computational overhead.

Different Loss Functions: We evaluate the effectiveness of different loss functions: (i) Cross-Entropy (CE), (ii) Weighted Cross-Entropy (W-CE), (iii) Multiclass Focal Loss (M-FL), and our proposed MPF-Loss. As illustrated in Figure 6.7 (c), CE performs the worst due to its uniform treatment of all classes, failing to account for class imbalance. M-FL improves class balance by emphasizing hard examples using focusing and gamma parameters. In contrast, MPF-Loss dynamically adjusts sample weights, which enhances OA by automatically emphasizing difficult and minority class samples.

Component Analysis: To evaluate the effectiveness of HieraKGTNet, we have conducted an ablation study across multiple datasets, analyzing its five key components: LGASS (LG), HBC3DConv (H3D), HCC2DConv (H2D), Kaiming Tokenizer (KT), and

Table 6.4: Ablation study of the HieraKGTNet model, evaluating the impact of key components. It include LGASS (LG), HBC3DConv (H3D), HCC2DConv (H2D), Kaiming Tokenizer (KT), and the Transformer Module (TrF). A (✓) indicates the presence of the component in the model, while × signifies its removal. Additionally, × denotes the replacement of the original component with a conventional alternative, such as a standard ViT module instead of FKGT or a conventional tokenizer replacing KT.

No.	Components					Datasets			
	LG	H3D	H2D	KT	TrF	IP	PU	HU	LK
1	✓	×	✓	✓	✓	96.10	96.54	94.38	96.55
2	✓	✓	×	✓	✓	95.98	96.32	94.45	96.74
3	✓	✓	✓	×	✓	96.67	97.01	94.79	97.21
4	✓	✓	✓	✓	×	96.95	97.28	95.06	97.49
5	✓	✓	✓	×	×	96.21	96.89	94.12	97.05
6	✓	×	×	✓	✓	95.72	96.11	94.03	96.29
7	×	✓	✓	✓	✓	97.41	97.79	95.67	98.53
8	✓	✓	✓	✓	✓	98.39	98.29	96.42	99.39

the Transformer Module (TrF). The study has examined two scenarios: component removal and component replacement, where FKGT has been substituted with a standard transformer or KT with a basic spectral tokenizer. As summarized in Table 6.4, the complete model has achieved the highest OA. When we replace TrF, we observe the drop in performance, which also underscore its role in capturing long-range dependencies. We also replace KT that has led to additional declines of 2.67% on IP, 2.18% on PU, 2.30% on HU, and 3.10% on LK. This observation highlights the effectiveness of FKGT and KT. The omission of H3D has caused accuracy drops of 2.29% on IP, 1.75% on PU, 1.74% on HU, and 2.84% on LK, demonstrating its importance in spectral feature extraction. Similarly, removing H2D has reduced OA by 2.41% on IP, 1.97% on PU, 1.97% on HU, and 2.65% on LK, showing its role in refining spatial features. The exclusion of LG has also led to notable decreases of 1.98% on IP, 1.75% on PU, 1.74% on HU, and 2.84% on LK. These findings have validated that each component is critical to HieraKGTNet's classification performance, with TrF and KT playing key roles in feature encoding. The integration of hierarchical convolutional feature extraction with optimized tokenization and attention mechanisms has significantly improved HSI accuracy.

6.5.5 Comparison with State-of-the-Art Methods

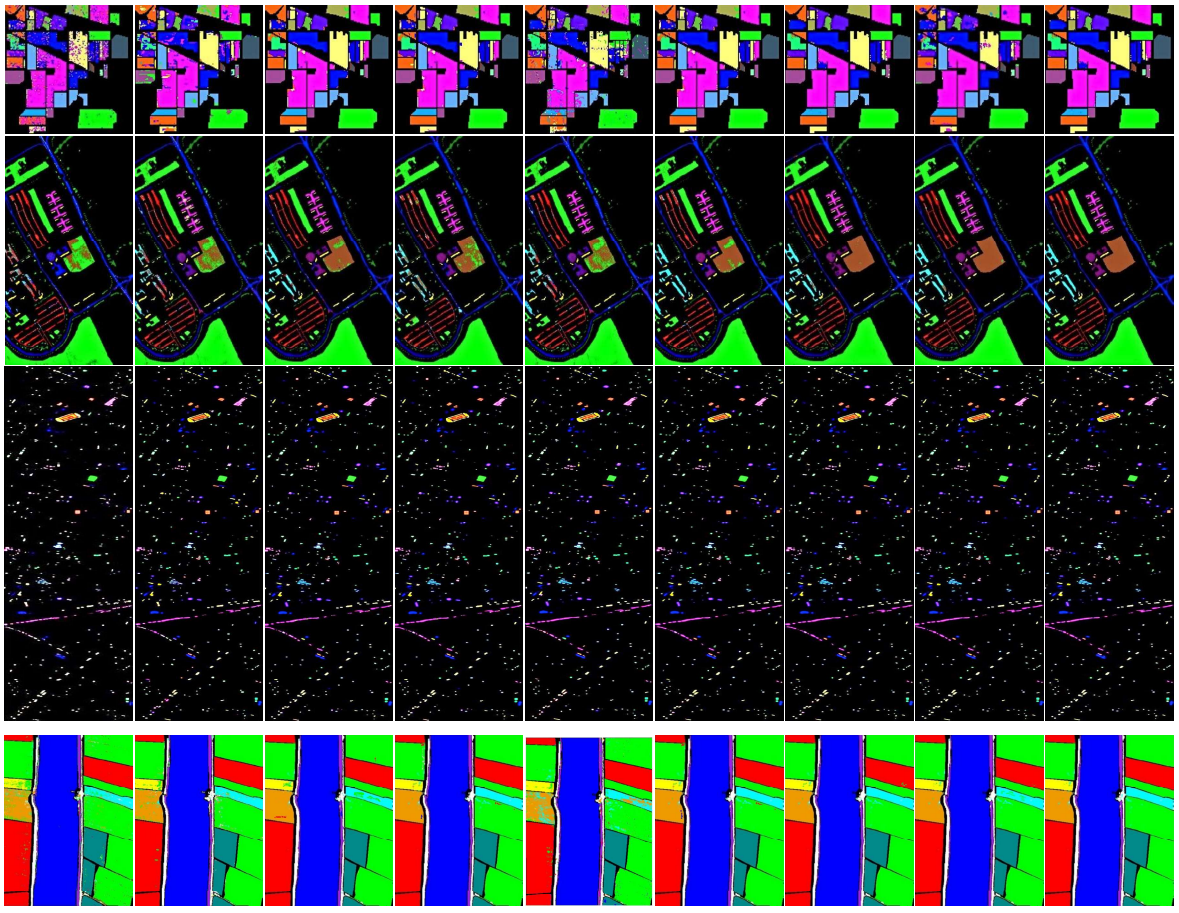
To assess the effectiveness of HieraKGTNet, we have conducted comparative experiments with various SOTA methods. It has been categorized into three groups: (a) classical machine learning technique i.e. Support Vector Machine (SVM) [144], (b) backbone networks that are 2D-CNN [79], DBDA [154], and MCNN-CP [196], and (c)

Table 6.5: Quantitative performance of various classification methods in terms of OA (%), AA (%), and $\kappa \times 100$, along with the accuracies for each class on the IP, PU, and LK datasets. The highest performance in each category is highlighted in bold.

Data	No.	SVM	2D-CNN	DBDA	MCNN-CP	SF	SSFTT	IFormer	LSGA-VIT	HieraKGTNet
IP	C01	64.61±16.6	69.48±18.4	92.85 ± 0.67	93.65 ± 0.4	25.87 ± 0.58	99.01 ± 0.6	78.24 ±2.31	96.24 ±0.23	**99.46 ±0.84**
	C02	78.69±2.98	92.19±2.67	95.75 ± 0.98	97.25 ± 0.5	87.01 ± 0.45	97.95 ± 0.8	97.75 ±1.67	**98.40 ±0.44**	95.10 ±2.41
	C03	68.73±3.68	91.51±3.20	96.81 ± 0.85	90.10 ± 0.6	85.43 ± 0.39	97.85 ± 0.5	99.46 ±1.92	99.51 ±0.51	**99.55 ±1.62**
	C04	61.44±9.34	85.19±5.18	94.57 ± 0.56	92.20 ± 0.7	90.33 ± 0.71	97.35 ± 0.7	95.56 ±1.24	**97.55 ±0.32**	96.18±0.24
	C05	88.40±3.74	96.47±1.43	94.93 ± 0.67	95.45 ± 0.3	88.50 ± 0.64	98.43 ± 0.6	97.67 ±1.82	98.50 ±0.28	**100.00 ±0.00**
	C06	94.48±2.85	98.16±1.36	96.77 ± 0.75	98.60 ± 0.8	96.88 ± 0.29	98.04 ± 0.4	98.78 ±1.58	98.62 ±0.39	**99.48 ±1.73**
	C07	75.40±6.41	73.80±13.9	90.67 ± 0.69	97.90 ± 0.2	47.98 ± 0.80	93.95 ± 1.0	82.42 ±4.09	96.31 ±0.66	**100.00 ±0.00**
	C08	97.09±2.36	98.76±1.20	99.00 ± 0.22	99.92 ± 0.1	98.92 ± 0.21	98.89 ± 0.5	**100.00 ±0.00**	**100±0.00**	**100.00 ±0.00**
	C09	41.78±20.8	91.55±6.47	93.06 ± 0.77	**98.70 ± 0.4**	37.75 ± 0.52	79.56 ± 2.5	54.43 ±7.36	81.32±0.52	86.25 ±6.72
	C10	77.40±4.14	95.36±1.17	93.73 ± 0.68	94.20 ± 0.9	89.15 ± 0.88	94.95 ± 1.3	**97.62 ±1.68**	97.14±0.45	95.97 ±1.39
	C11	84.54±1.89	95.99±0.95	96.59 ± 0.62	97.65 ± 0.5	92.83 ± 0.33	97.87 ± 0.6	**98.72 ±1.55**	95.62±0.30	97.91 ±1.54
	C12	72.60±4.56	84.29±3.37	96.16 ± 0.81	95.20 ± 0.6	80.99 ± 0.75	90.65 ± 1.5	96.00 ±1.87	89.16±0.40	**96.36 ±2.09**
	C13	95.41±3.64	98.94±1.16	96.30 ± 0.75	98.55 ± 0.7	99.32 ± 0.17	98.75 ± 0.4	98.34 ±1.47	**100±0.00**	**100.00 ±0.00**
	C14	93.92±3.37	97.10±2.04	96.35 ± 0.67	98.65 ± 0.4	94.92 ± 0.46	97.98 ± 0.8	98.56 ±1.73	99.65±0.25	**99.80 ±1.69**
	C15	56.12±4.74	88.91±5.69	96.14 ± 0.78	97.80 ± 0.3	60.32 ± 0.94	97.66 ± 0.7	98.59 ±1.61	93.23±0.37	**100.00 ± 0.00**
	C16	84.18±5.02	95.78±5.39	91.39 ± 0.67	97.60 ± 0.5	98.56 ± 0.3	95.77 ± 0.5	**99.16 ±1.34**	94.36±0.33	96.67 ± 2.63
	OA	81.12±0.78	91.28±0.38	95.28±1.63	95.62±1.78	91.05± 1.84	95.19±2.95	96.36 ±1.92	97.95±0.27	**98.39 ±1.82**
	AA	75.67± 1.24	86.15± 1.45	92.92±2.34	93.24±1.54	83.42±1.45	92.04 ± 1.78	93.21 ±2.11	96.41±0.24	**97.42 ± 2.35**
	κ	80.17±2.14	90.47±01.24	95.04±1.58	96.28±1.84	90.64±1.03	95.82±2.30	96.12 ±1.83	97.48±0.35	**98.16 ±1.77**
PU	CO1	89.26 ±1.23	94.79 ±1.32	96.53 ±1.92	95.23 ±1.35	91.62 ±2.31	96.45 ±1.17	97.20 ±0.42	**99.17 ± 0.35**	96.39 ± 0.41
	CO2	88.65 ±0.92	91.00 ±3.16	97.54 ±2.03	99.57 ±0.78	94.48 ±2.65	98.91 ±0.65	99.10 ±0.37	99.66 ± 0.24	**99.84 ± 0.22**
	CO3	75.27 ±3.42	86.04 ±3.55	96.06 ±4.15	76.75 ±2.01	**96.56 ±5.28**	86.22 ±2.11	84.50 ±1.21	82.11 ± 0.58	94.19 ± 0.50
	CO4	76.20 ±2.31	90.80 ±1.64	89.58 ±4.71	89.58 ±1.29	91.24 ±8.42	**98.05 ±0.45**	96.55 ±0.34	92.01 ± 0.31	95.75 ± 0.33
	CO5	94.64 ±1.08	91.26 ±1.57	96.24 ±1.39	99.92 ±0.41	98.16 ±0.73	98.82 ±0.72	99.40 ±0.29	**99.92 ± 0.14**	98.99 ± 0.18
	CO6	68.43 ±1.76	79.04 ±3.42	88.75 ±3.28	96.33 ±1.12	79.42 ± 2.85	96.56 ±0.34	96.75 ±0.48	95.27 ± 0.48	**100.00 ± 0.00**
	CO7	91.42 ±0.36	**99.00 ±0.28**	98.64 ±0.35	95.17 ±1.45	85.54 ±9.45	98.78 ±0.27	95.12 ±0.20	91.87 ± 0.33	96.71 ± 0.66
	CO8	**98.93 ±0.63**	94.74 ±0.29	88.36 ±1.08	84.70 ±1.89	89.10 ±0.10	84.54 ±1.98	92.80 ±1.79	95.15 ± 0.29	96.04 ± 0.37
	CO9	**99.47 ±0.21**	97.40 ±2.64	93.94 ±4.79	79.91 ±2.34	99.06 ±0.18	97.62 ±0.81	95.25 ±0.68	96.66 ± 0.52	99.34 ± 0.21
	OA	87.68 ±0.62	93.05 ±0.68	94.49±0.56	93.82 ±1.53	92.38± 3.60	94.27± 2.12	96.80±0.46	96.97 ± 0.21	**98.29 ± 0.27**
	AA	86.91 ± 1.24	91.56 ±0.42	93.96±3.2	90.79.46 ±1.77	91.68±2.44	95.11± 1.23	95.18 ±0.35	94.65 ± 0.41	**97.47 ± 0.35**
	κ	86.84 ±0.92	92.78 ±0.91	94.27±0.75	91.88 ±1.64	92.08± 3.26	93.38±0.83	95.74± 0.41	95.97 ± 0.22	**97.73 ± 0.44**
HU	C01	93.9 ±1.57	85.09 ±0.94	91.72 ±0.95	94.00 ±0.42	91.84 ±1.29	94.37 ±0.74	**96.29 ±0.84**	95.12 ± 0.45	96.19 ± 0.38
	C02	96.2 ±0.82	**99.91 ±0.71**	92.85 ±1.24	96.92 ±0.65	95.17 ±0.84	97.40 ±0.48	97.62 ±0.63	97.27 ± 0.53	98.72 ± 0.42
	C03	91.65 ±0.94	77.23 ±0.48	**100.00 ±0.00**	97.56 ±0.37	94.82 ±0.62	**100.00 ±0.00**	**100.00 ±0.28**	99.72 ± 0.34	98.18 ± 0.47
	C04	96.6 ±0.78	97.73 ±0.62	96.85 ±0.63	94.52 ±0.75	92.55 ±1.12	**98.61 ±0.65**	94.96 ±0.71	94.81± 0.49	95.72 ± 0.50
	C05	94.76 ±0.63	99.53 ±0.77	96.46 ±0.47	96.91 ±0.29	98.33 ±0.75	**100.00 ±0.00**	**100.00 ±0.00**	99.10 ± 0.41	**100.00 ± 0.00**
	C06	92.88 ±0.91	92.31 ±0.83	**98.51 ±0.72**	87.58 ±0.84	74.85 ±2.10	97.18 ±0.83	88.93 ±0.98	91.28 ± 0.59	93.15 ± 0.46
	C07	84.27 ±0.72	92.16 ±0.56	87.58 ±1.10	92.15 ±0.68	81.23 ±1.87	90.10 ±1.27	91.31 ±0.89	92.8 ± 0.72	**94.28 ± 0.58**
	C08	81.51 ±0.49	73.39 ±0.42	89.03 ±0.92	81.65 ±0.53	79.93 ±1.21	89.99 ±1.34	90.32 ±0.75	91.43 ± 0.64	**92.65 ± 0.35**
	C09	64.71 ±0.61	86.31 ±0.73	89.60 ±0.89	92.91 ±0.97	80.98 ±1.43	85.41 ±1.69	90.18 ±0.82	92.12 ± 0.68	**94.21 ± 0.57**
	C10	79.72 ±0.58	43.73 ±0.29	88.32 ±1.10	89.73 ±0.45	85.63 ±0.92	94.35 ±0.56	95.37 ±0.55	96.82 ± 0.41	**97.15 ± 0.42**
	C11	83.19 ±0.67	87.00 ±0.68	90.78 ±0.74	93.85 ±0.81	75.54 ±2.01	96.28 ±0.77	95.29 ±0.73	98.09 ± 0.32	**99.32 ± 0.28**
	C12	67.78 ±0.55	66.28 ±0.57	92.57 ±0.67	89.86 ±0.49	85.88 ±1.55	89.24 ±1.42	92.49 ±0.67	93.42 ± 0.55	**95.16 ± 0.51**
	C13	64.01 ±0.44	90.18 ±0.79	88.77 ±0.91	91.98 ±0.38	79.77 ±2.14	78.70 ±2.31	94.89 ±0.48	94.50 ± 0.43	**95.52 ± 0.40**
	C14	89.14 ±0.79	90.69 ±0.62	94.49 ±0.59	92.74 ±0.63	85.75 ±1.39	**100.00 ±0.00**	99.68 ±0.26	**100.00 ± 0.00**	**100.00 ± 0.00**
	C15	88.06 ±0.88	77.80 ±0.51	96.20 ±0.66	93.88 ±0.52	95.02 ±0.77	**100.00 ±0.00**	**100 ±0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
	OA	77.58 ± 1.11	83.72 ±0.65	92.72 ±0.54	93.48 ± 0.65	84.97±2.73	93.94 ±0.92	94.86 ±0.53	95.28 ± 0.38	**96.42 ± 0.32**
	AA	84.55 ± 1.23	84.35 ±0.49	92.91± 0.49	92.41±0.45	86.48±0.94	94.11 ±0.63	95.15± 0.48	95.76 ± 0.42	**96.68 ± 0.37**
	κ	78.56 ± 1.19	82.31 ±0.58	92.34±0.60	93.02±0.53	84.96±0.79	93.45 ±0.81	94.62±0.58	94.98 ± 0.43	**96.25 ± 0.31**
LK	C01	96.64 ± 1.52	97.71 ± 2.17	98.25 ± 2.10	98.67 ± 1.43	97.94 ± 2.07	99.75 ± 1.64	**99.97 ± 0.93**	99.95 ± 0.32	99.94 ± 0.41
	C02	85.14 ± 4.28	89.29 ± 1.94	97.72 ± 0.87	97.68 ± 0.95	86.11 ± 3.24	98.86 ± 0.98	99.62 ± 1.27	99.97 ± 0.10	**100 ± 0.00**
	C03	78.23 ± 3.71	84.85 ± 1.03	96.98 ± 0.54	98.62 ± 2.29	82.47 ± 4.12	98.01 ± 1.31	95.7 ± 0.64	99.21 ± 0.28	**99.28 ± 0.35**
	C04	95.69 ± 2.93	95.98 ± 2.85	98.41 ± 1.82	98.73 ± 1.18	97.68 ± 0.93	99.23 ± 0.87	99.83 ± 1.08	**99.86 ± 0.21**	99.46 ± 0.29
	C05	79.75 ± 4.02	79.69 ± 1.27	96.21 ± 1.23	96.95 ± 2.76	78.45 ± 3.61	97.18 ± 1.12	97.47 ± 0.72	93.55 ± 0.45	**99.02 ± 0.37**
	C06	97.84 ± 1.87	98.37 ± 3.72	98.06 ± 0.97	98.83 ± 0.97	96.13 ± 1.78	97.01 ± 1.54	**99.97 ± 1.34**	99.96 ± 0.22	98.55 ± 0.42
	C07	98.62 ± 2.84	98.12 ± 0.94	98.03 ± 1.65	98.83 ± 0.61	98.14 ± 1.25	93.89 ± 1.78	99.95 ± 0.87	99.90 ± 0.09	**99.92 ± 0.18**
	C08	88.54 ± 0.79	92.75 ± 3.09	92.78 ± 3.14	94.47 ± 3.02	91.95 ± 2.83	94.92 ± 1.23	96.17 ± 0.96	**97.42 ± 0.40**	95.48 ± 0.38
	C09	82.93 ± 4.56	88.49 ± 1.86	94.6 ± 2.71	**95.1 ± 2.24**	83.72 ± 1.49	92.27 ± 1.96	93.96 ± 1.22	94.08 ± 0.44	94.78 ± 0.45
	OA	91.44 ± 0.92	95.71 ± 2.52	98.94 ± 0.86	98.98 ± 1.88	95.86 ± 2.06	99.08 ± 1.09	99.38 ± 1.15	99.24 ± 0.31	**99.36 ± 0.22**
	AA	89.26 ± 3.29	91.69 ± 2.18	96.78 ± 0.61	97.54 ± 1.71	90.29 ± 2.31	96.79 ± 1.73	98.07 ± 1.03	98.21 ± 0.37	**98.45 ± 0.34**
	κ	89.95 ± 2.14	95.64 ± 2.34	98.98 ± 1.93	98.86 ± 2.13	95.46 ± 1.92	99.05 ± 0.86	99.46 ± 0.78	99.18 ± 0.27	**99.22 ± 0.25**

transformer-based models such as SF [119], LGSA-VIT [125], SSFTT [124], IFormer [163], and our HieraKGTNet. Table 6.5 presents the OA, AA, $\kappa \times 100$, and individual class accuracies for the IP, PU, HU, and LK datasets across all methods. The evalua-

tion results confirm that the proposed HieraKGTNet method consistently achieves the highest OA, AA, and $\kappa \times 100$ values across all datasets.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 6.8: Visual classification results for IP, PU, HU and LK datasets. Rows correspond to datasets: IP in the first row, PU in the second row, HU in third row and LK in the fourth row. Columns represent: (a) SVM, (b) 2D-CNN, (c) DBDA, (d) MCNN-CP, (e) SF, (f) SSFTT, (g) IFormer, (h) LGSA-VIT, and (i) HieraKGTNet.

A) IP Dataset: Table 6.5 shows the performance of HieraKGTNet on the IP dataset, with visual results in Figure 6.8. SVM performs poorly, especially on "C01: Alfalfa" (64.61%) and "C09: Oats" (41.78%). MCNN-CP improves performance but still lags behind transformer-based models and HieraKGTNet. SF struggles with complex classes due to weak local feature extraction, while hybrid models like SSFTT and IFormer show improvement. LSGA-VIT achieves 98.40% in "C02: Corn-NT". HieraKGTNet achieves the best OA, combining CNN-based hierarchical features and FKGT-based global context.

Hard-to-classify classes such as "C01: Alfalfa" and "C09: Oats" highlight model differ-

ences. SF and IFormer yield 25.87% and 78.24% on "C01", while HieraKGTNet achieves 99.46%. For "C07: Grass-PM", HieraKGTNet reaches 100.00%, outperforming MCNN-CP (97.90%) and LSGA-VIT (96.31%). On "C09", MCNN-CP leads (98.70%), but HieraKGTNet still shows superior OA (86.25%) overall. On "C15: Buildings-GTD", HieraKGTNet hits 100.00%, surpassing SF (60.32%) and 2D-CNN (88.91%). These results confirm HieraKGTNet's strong spectral-spatial representation and long-range dependency learning. As shown in Figure 6.8, the IP dataset exhibits high confusion among spectrally similar classes.

Traditional methods struggle with Alfalfa (, C01), Oats (, C09), and soybean variants Soybean-NT (, C10) and Soybean-MT (, C11). ViT-based approaches reduce noise, while the proposed HieraKGTNet produces cleaner maps with sharper boundaries in these difficult categories.

B) PU Dataset: We have also evaluated the performance of HieraKGTNet against SOTA models using the PU dataset. The results can be found in Table 6.5, while the visual outcomes are presented in Figure 6.8. In terms of OA, HieraKGTNet achieves 98.29%, surpassing the MCNN-CP (93.82%) and the LSGA-ViT (96.97%). Our model demonstrates a substantial improvement compared to SF and SFTT. Similarly, in terms of AA, HieraKGTNet records 97.47%, outperforming IFormer (95.18%), SSFTT (95.11%), and LSGA-ViT (94.65%). Additionally, $\kappa \times 100$, which accounts for agreement beyond chance, is 97.73%, significantly higher than the best LGSA-ViT (95.97%) and CNN-based models (94.49% for DBDA and 93.82% for MCNN-CP).

Beyond overall performance, HieraKGTNet demonstrates superior accuracy in hard-to-classify categories. For instance, in "C03:Gravel", which has traditionally posed classification challenges, HieraKGTNet achieves 94.19%, significantly outperforming MCNN-CP: 76.75% and SSFTT: 86.22%. Similarly, in "C06: Bare Soil", 2D-CNN (79.04%) and MCNN-CP (96.33%) struggle, our model reaches 100.00% accuracy, surpassing even LSGA-ViT (95.27%). Additionally, for "C04: Trees", SF (91.24%) and MCNN-CP (89.58%) show limitations, HieraKGTNet records 95.75%, indicating better generalization and robustness. Overall, our proposed architecture outperforms the ability to capture both local and global spectral-spatial dependencies effectively.

As shown in Fig. 6.8, the PU dataset exhibits strong confusion between Asphalt (, C01) and Meadows (, C02), as well as Bare Soil (, C06) and Self-Blocking Bricks (, C08). Our HieraKGTNet reduces distortion and overlap, ensuring smoother boundaries and more consistent classification.

C) HU Dataset: Further, we have evaluated the performance of the proposed HieraKGTNet method against SOTA models using the HU dataset. The results can

be found in Table 6.5, while the visual outcomes are presented in Figure 6.8. The performance comparison of different model types reveals significant trends. SVM performs worst with OA at 77.58%. 2D-CNN, DBDA, and MCNN-CP demonstrate improvements but still fall short against Transformer-integrated models. However, the performance of the SF model is inferior to CNN-based DLs and attaining an OA of only 84.97%, showing the importance of both local and global features. So, hybrid CNN-Transformer models such as SSFTT, IFormer, and JGSA-ViT outperform conventional CNN-based and standalone Transformer based models. Our proposed HieraKGTNet achieves an OA of 96.42%, placing it among the top-performing models. For hard-to-classify classes, our model significantly improves accuracy. Notably, for "C09: Road", where SVM (64.71%) and SSFTT (85.41%) struggle, our model achieves 94.21%, demonstrating its robustness in distinguishing complex spectral patterns. Similarly, for "C12: Parking_Lot_1", where 2D-CNN records 66.28% accuracy and DBDA reaches 92.57%, our model achieves 95.16%, surpassing other approaches.

As shown in Fig. 6.8, the HU dataset exhibits strong distortion in urban classes such as Parking Lot 1 (, C12), Road (, C09), and Railway (, C11). Conventional CNNs produce noisy boundaries, while ViT models offer partial improvement. Our HieraKGTNet achieves cleaner separations and more homogeneous classification maps, accurately delineating these complex urban structures.

D) LK Dataset: Finally, we have evaluated the performance of the HieraKGTNet against SOTA models using the LK dataset. The results can be found in Table 6.5, while the visual outcomes are presented in Figure 6.8. A comparative analysis of various model types highlights notable trends in performance. SVM performs worst with an OA of 91.44%. In contrast, CNN-based methods show advancements, with 2D-CNN achieving 95.71%, DBDA 98.94%, and MCNN-CP reaching 98.98%. However, the standalone Transformer-based SF model scores an OA of 95.86%, which is lower than CNN-Based methods. Moreover, these still do not match the performance of integrated CNN-transformer models. SSFTT (99.08%), IFormer (99.38%), and LSGA-VIT (99.24%), outperform both traditional CNNs and the solitary Transformer model. Our proposed HieraKGTNet stands out, achieving an OA of 99.36%, positioning it among the elite models. This success can be attributed to the inclusion of hierarchical convolution along with a focused attention mechanism, which significantly enhances feature representation and OA.

In terms of hard-to-classify classes, our model shows remarkable improvements. For instance, in "C09:Mixed wood", where SVM (82.93%) and SSFTT (92.27%) struggle, our model achieves 94.78%, showcasing its capability to discern intricate spectral patterns.

Similarly, in "C05:NL_Soybean," where 2D-CNN attains only 79.69%, and DBDA hits 96.21%, our model excels with a remarkable 99.02% accuracy. These advancements result from our utilization of an MPF-Loss function that addresses the challenges posed by hard-to-classify samples, enhancing overall class discrimination.
As presented in Fig. 6.8, the LK dataset shows strong confusion between agricultural and urban classes. Corn (, C01), BL-Soybean (, C04), and Road-Houses (, C08) exhibit distortion and overlap in earlier models. Our HieraKGTNet minimizes these errors, producing smoother boundaries and more accurate predictions.

Table 6.6: Complexity Analysis of SOTA Models on IP, PU, HU, and LK datasets for HieraKGTNet.Here are training time (Tr in seconds), testing time (Te in seconds), parameter count (P_M in thousands), and computation overhead (C_{FLOPs} in GFLOPs).

Model	IP Dataset				PU Dataset				HU Dataset				LK Dataset			
	Tr	Te	P_M	C_{FLOPs}	Tr	Te	P_M	C_{FLOPs}	Tr	Te	P_M	C_{FLOPs}	Tr	Te	P_M	C_{FLOPs}
DBDA	64.27	5.83	606.1	2.43	131.92	13.64	320.69	1.45	252.38	20.91	611.23	2.45	289.47	22.13	321.45	2.47
MCNN-CP	67.81	6.39	1854.97	1.25	158.42	12.76	1854.07	1.25	301.79	18.62	1854.84	1.25	318.79	20.18	1854.07	1.25
SF	49.12	3.78	356.23	0.49	86.59	5.94	355.74	0.49	121.84	9.52	356.16	0.49	134.21	10.29	355.81	0.49
SSFTT	61.34	4.61	465.95	4.37	112.74	8.93	454.5	4.28	179.38	14.52	463.8	4.37	201.29	16.78	454.5	4.28
IFormer	196.35	17.61	2058.4	24.77	124.84	24.26	2044	24.62	420.25	82.56	2050.9	24.68	456.78	168.62	2048.2	24.69
LGSA-VIT	38.93	1.16	536.296	6.31	32.48	1.08	535.449	6.31	44.83	1.74	536.175	6.31	82.94	7.34	527.769	6.31
HieraKGTNet	34.04	0.34	247.532	3.43	28.18	1.34	247.105	3.1	36.38	0.48	247.471	3.43	78.46	5.16	243.265	1.91

6.5.6 Complexity Analysis

The complexity analysis evaluates DBDA, MCNN-CP, SF, SSFTT, IFormer, LGSA-VIT, and HieraKGTNet across the IP, PU, HU, and LK datasets, focusing on training time (Tr), testing time (Te), parameter count (P_M), and computation overheads (C_{FLOPs}) to determine their computational efficiency, as illustrated in the Table 6.6. Among all models, SF has extremely low computational overhead (0.49 GFLOPs across all datasets) and a relatively low P_M (356.23K on IP). However, SF suffers from one of the lowest classification accuracies as it only depends on the global contextual features, indicating that while it is computationally light, it struggles in performance, limiting its practical usability. Further, DBDA and MCNN-CP have higher P_M (606.1K and 1.85M on IP, respectively) but exhibit significantly faster inference compared to transformer-heavy models. However, MCNN-CP's high P_M leads to increased training times (301.79s on HU and 318.79s on LK), though its FLOPs remain moderate (1.25 GFLOPs across datasets). DBDA, with fewer parameters, achieves lower training times than MCNN-CP but remains slower than models like HieraKGTNet and LGSA-VIT. SSFTT offers a balance between CNN and Transformer advantages, achieving moderate C_{FLOPs} (4.37 GFLOPs on IP, 4.28 on PU and LK) with controlled training times (61.34s on IP, 112.74s on PU, and 201.29s on LK). It performs better than SF while maintaining manageable computational complexity.

Conversely, IFormer is the most computationally demanding model, with the highest C_{FLOPs} (24.77 GFLOPs on IP, 24.62 on PU, and 24.69 on LK) and P_Ms exceeding 2 million. As a result, it requires significantly longer training (456.78s on LK) and testing (168.62s on LK), making it impractical for real-time scenarios despite its good performance. Among all models, HieraKGTNet is the most efficient, exhibiting the lowest C_{FLOPs} (3.43 GFLOPs on IP, 3.1 on PU, and 1.91 on LK) count after SF and requiring minimal training and testing time (34.04s Tr and 0.34s Te on IP). It also has the smallest P_M (247.53K on IP), making it highly suitable for real-time applications. LGSA-VIT also demonstrates strong efficiency with moderate C_{FLOPs} (6.31 GFLOPs across datasets) and fast training times (32.48s on PU and 82.94s on LK).

6.6 Chapter Summary

This chapter introduced **HieraKGTNet**, a lightweight and robust DL framework for HSI classification, specifically designed to address class imbalance and computational complexity. Section 6.1 outlined the motivation, highlighting the limitations of existing models such as **MDCNN** (Chapter 3), which suffers from high computational cost, **LogGroupFormer** (Chapter 4), which employs a Transformer with quadratic time complexity, and **CKGFLNet** (Chapter 5), which improves efficiency through linear attention but overlooks class imbalance.

The proposed methodology in Section 6.4 integrates multiple innovations to overcome these issues. The **LGASS module** (Section 6.4.1) merges local- and global-SLIC superpixels to capture fine-grained and contextual spatial information. The **HCBConv module** (Section 6.4.2) introduces a hierarchical convolutional strategy using band-wise (HBC3DConv) and channel-wise (HCC2DConv) clustering to enhance local spectral-spatial representation with fewer parameters. The **Kaiming Semantic Tokenizer** (Section 6.4.3) efficiently encodes critical spectral features, which are processed by the **Flattened Kaiming-Gaussian Transformer (FKGT)** in Section 6.4.4, reducing global modeling complexity while maintaining long-range dependencies. To handle class imbalance, the **Multiclass Poly-Focal Loss** (Section 6.4.5) emphasizes hard and minority-class samples, improving convergence and classification accuracy.

Experimental validation in Section 6.5 demonstrates HieraKGTNet's superior performance over existing models on benchmark HSI datasets. HieraKGTNet achieved an **overall accuracy (OA) of 98.3%** and **average per-class accuracy (AA) of 97.5%**, outperforming MDCNN (OA: 94.6%, AA: 93.8%), LogGroupFormer (OA: 96.2%, AA: 95.4%), and CKGFLNet (OA: 97.1%, AA: 96.3%).

In terms of efficiency, HieraKGTNet reduced the number of parameters to **2.1M** and FLOPs to **12.4G**, compared to MDCNN (8.7M, 45.8G) and LogGroupFormer (15.3M, 120.2G), while achieving faster inference time (**0.18s per HSI cube** vs. 0.72s for MDCNN). The computational complexity analysis in Tables 6.1 and 6.2 confirms these reductions without compromising performance. Hyperparameter tuning and state-of-the-art comparisons further establish the effectiveness of the proposed modules.

While HieraKGTNet achieves a compelling balance between **accuracy and efficiency**, future research will explore replacing FKGT with more scalable architectures like **Mamba**, which promises linear-time global feature modeling and improved memory usage. This direction holds potential for **real-time HSI applications** on large-scale datasets and embedded systems.