Chapter 4

Lightweight Deep Neural Networks Leveraging Logarithmic Feature Grouping for HSI Classification

4.1 Introduction

HSI classification is essential for remote sensing and geospatial analysis, enabling precise land-cover identification through rich spectral cues. Its high spectral resolution helps differentiate materials and surface types, which is vital for environmental monitoring, precision agriculture, and urban planning. However, the high-dimensional nature of HSI data poses challenges such as high computational complexity, the curse of dimensionality, and the need for effective feature extraction techniques. DL approaches, particularly CNNs, have remarkably succeeded in HSI classification by extracting hierarchical spectral-spatial features. Chapter 3 introduced the MDCNN to integrate spectral, spatial, and morphological features, leveraging mathematical morphological operations and hybrid 3D-2D convolutional layers. Although MDCNN improved classification accuracy compared to traditional methods, it was limited in capturing long-range dependencies and global contextual information. Additionally, its reliance on extensive computational resources posed scalability challenges.

However, CNNs focus primarily on local spectral-spatial relationships, which makes them less effective in capturing long-range dependencies. These dependencies are crucial for comprehensive scene understanding from an image. Local contextual information enhances feature discrimination by preserving fine-grained spectral-spatial details, while global contextual information captures broader spatial correlations and semantic structures across the image. A robust HSI classification framework should integrate

both local and global features to balance fine-detail accuracy with overall scene coherence, ensuring improved classification performance and generalization across datasets.

In order to overcome the challenges mentioned above, this chapter presents a novel lightweight DL framework, **LogGroupFormer**. It enhances local-global feature extraction by combining CNNs with a vision transformer. The model introduces a logarithmic-based 3D and 2D group convolution mechanism for efficient spectral-spatial feature extraction. It also incorporates sine-cosine positional embeddings to improve spatial encoding. By leveraging transformer architectures, LogGroupFormer captures long-range dependencies. This improves classification accuracy while reducing computational overhead. The key contributions of this chapter are as follows:

1. **Efficient Spectral-Spatial Feature Extraction using Logarithmic Convolution:** A logarithmic-based 3D and 2D group convolution mechanism is introduced to extract correlated spectral-spatial features while minimizing computational costs. By mapping HSI input channels and filters into a logarithmic domain before convolution, our model achieves a more compact representation. Moreover, it also reduces parameter requirements and enhances computational efficiency.
2. **Enhanced Spatial Contextual Positional Encoding:** We introduce a spatial contextual sine-cosine positional embedding to enhance the ability of our proposed model and capture spatial nuances. Our approach improves classification accuracy and representations. The improvement is done by incorporating spatial positions into feature extraction using sine and cosine functions.
3. **Hybrid Local-Global Contextual Feature Extraction Framework:** The proposed model integrates CNNs for fine-grained spectral-spatial feature extraction with ViT architecture for modeling long-range dependencies and global relationships, ensuring a comprehensive feature representation that enhances HSI classification performance.

This chapter is structured as follows: Section 4.2 reviews SOTA HSI classification architectures, highlighting their methodologies and challenges. Next, Section 4.3 defines the problem statement, while Section 4.4 presents the proposed methodology, detailing its framework and innovations. Section 4.5 describes the experimental setup and compares results against existing models. Finally, Section 4.6 summarizes the findings and contributions of this work.

4.2 Related Work

4.2.1 CNN-Based Local Contextual Feature Extraction

CNNs are widely used in HSI classification due to their ability to learn intricate spectral patterns [147, 148]. However, capturing both spectral and spatial features remains challenging. Early 3D convolution-based models preserved spectral-spatial correlations but had high computational costs and limited receptive fields [149]. Hybrid models emerged to address these issues. These models integrate 2D and 3D convolution layers within a unified framework. The 2D convolution captures fine spatial details, while the 3D convolution extracts spectral-spatial relationships [150]. The concept of residual connections was first introduced in ResNet [151, 152]. It helps mitigate performance degradation in deeper spectral-spatial CNNs [153]. Embedding attention mechanisms further refine CNN-based models by emphasizing crucial spectral-spatial patterns. The attention mechanism helps in improving classification accuracy [154–156]. However, conventional attention mechanisms are limited by the local operations of CNNs. This constraint reduces their ability to model long-range dependencies. Future research should address these limitations to enhance spectral-spatial feature extraction.

4.2.2 Vision Transformer (ViT) for Long-Range Dependencies

The transformer paradigm has originally been developed for natural language processing and has significantly advanced computer vision. It introduced a novel approach to image analysis by leveraging self-attention mechanisms. ViT [67] marked a major breakthrough in image classification, achieving competitive performance compared to CNNs. This innovation paved the way for more efficient transformer-based architectures in visual recognition tasks. In HSI classification, ViT has excelled at capturing intricate long-term spectral-spatial dependencies. The Deep Spatial-Spectral Transformer [157] and the Spectral-Swin Transformer [158] have significantly enhanced spectral-spatial feature extraction, enabling more detailed HSI analysis. Similarly, the Nested Transformer [159] and the SSFTT [124] have further improved global feature representation and high-level spectral-spatial modeling, leading to superior classification accuracy. Moreover, [125] extended SSFTT with LSGA-ViT, integrating a lightweight self-attention mechanism for efficient global feature extraction. This model reduces C_{FLOPs} by employing a hybrid spatial-spectral tokenizer and a Gaussian absolute position bias while enhancing classification performance. These advancements highlight the transformative role of ViTs in HSI analysis, enhancing classification accuracy, robustness, and efficiency, and affirming their potential as key tools in the field.

4.2.3 Group Convolution

CNNs excel at feature extraction in HSI classification applications due to their diverse filter banks, which capture complex patterns. However, their high parameter count often leads to overfitting and poor generalization, particularly when labeled training data is scarce. Additionally, their substantial computational and memory demands hinder deployment in resource-limited environments.

To address these challenges, researchers have explored alternative convolutional strategies. Li et al. [160] introduced group CNNs, partitioning feature maps into smaller groups for separate processing, reducing computational complexity while maintaining representational power. Wang et al. [161] further optimized this approach by employing spatially separable convolutional layers to minimize parameter redundancy without compromising spatial feature integrity. Moreover, Gao et al. [162] proposed using two independent 2D kernels along the X-L and Y-L spatial dimensions instead of a conventional 3D kernel, significantly lowering computational costs while enhancing efficiency. These advancements in group convolution enhance feature extraction, improving the scalability and practicality of CNN-based HSI models in real-world applications.

4.3 Problem Statement

Given an HSI cube $\mathbf{I} \in \mathbb{R}^{[H\times W\times B]}$, where H, W, and B represent the spatial height, width, and the number of spectral bands, respectively, our approach ensures an efficient and accurate classification framework. To efficiently capture spectral correlations while maintaining computational efficiency, we introduce a logarithmic spectral grouping function $\mathcal{L} : \mathbb{R}^{[1\times 1\times B]} \to \{\mathbf{g}_1, \ldots, \mathbf{g}_m\}$, where each group $\mathbf{g}_i \in \mathbb{R}^{[1\times 1\times b_i]}$ represents a subset of correlated spectral bands. The grouping is designed such that $\sum_{i=1}^{m} b_i = B$, with group sizes determined using a logarithmic distribution. If the number of groups m is even, then the sizes are assigned using the normalized rule: $b_i = \frac{B}{2^i} \cdot \left(\frac{1}{\sum_{j=1}^{m} \frac{1}{2^j}} \right)$. Otherwise, the first group is assigned half the bands as $b_1 = B/2$, while the remaining $m-1$ groups are equally divided as: $b_i = \frac{B}{2(m-1)}$, for $i = 2, \ldots, m$. To further enhance spectral-spatial feature extraction, we introduce a Logarithmic Group Convolutional Operator $F_{\text{LGConv}} : \mathbf{I} \to \mathbb{R}^{[H\times W\times \mathfrak{K}_{\text{LG}}]}$. The operator consists of 3D and 2D convolutions applied within the log-mapped groups to extract fine-grained local spectral-spatial features efficiently. The extracted features are then transformed by a spatial-contextual patch embedding function F_{SCPE}. The obtained features are restructured into tokens $\mathbf{T} \in \mathbb{R}^{[N\times \mathfrak{K}_{\text{token}}]}$, where N is the number of patches. To en-

rich spatial understanding, a novel spatial-contextual sine-cosine positional embedding $F_{\text{Pos}} : \mathbb{R}^{[N\times\mathfrak{K}_{\text{token}}]} \rightarrow \mathbb{R}^{[N\times\mathfrak{K}_{\text{token}}]}$ is applied, where the positional embeddings are added to the tokens to encode spatial relationships. The resulting sequence is passed to a Transformer encoder $F_{\text{ViT}} : \mathbb{R}^{[N\times\mathfrak{K}_{\text{token}}]} \rightarrow \mathbb{R}^{[N\times\mathfrak{K}_{\text{ViT}}]}$, which captures global dependencies. Finally, we integrate local and global features that maximize classification accuracy $\hat{\mathcal{C}}$, while minimizing the computational burden $\hat{\mathcal{L}}_{\text{comp}}$. The optimization problem can be mathematically formulated as:

$$\underset{F_{\text{LGConv}},F_{\text{SCPE}},F_{\text{Pos}},F_{\text{ViT}},F_{\text{classify}}}{\text{maximize}} \quad \hat{\mathcal{C}}(F_{\text{classify}}(F_{\text{ViT}}(F_{\text{Pos}}(F_{\text{SCPE}}(F_{\text{LGConv}}(\mathbf{I})))))) - \lambda \cdot \hat{\mathcal{L}}_{\text{comp}}(\mathbf{I}), \tag{4.1}$$

where F_{classify} denotes the classification head with Softmax output, and λ is a regularization factor to balance accuracy and efficiency.

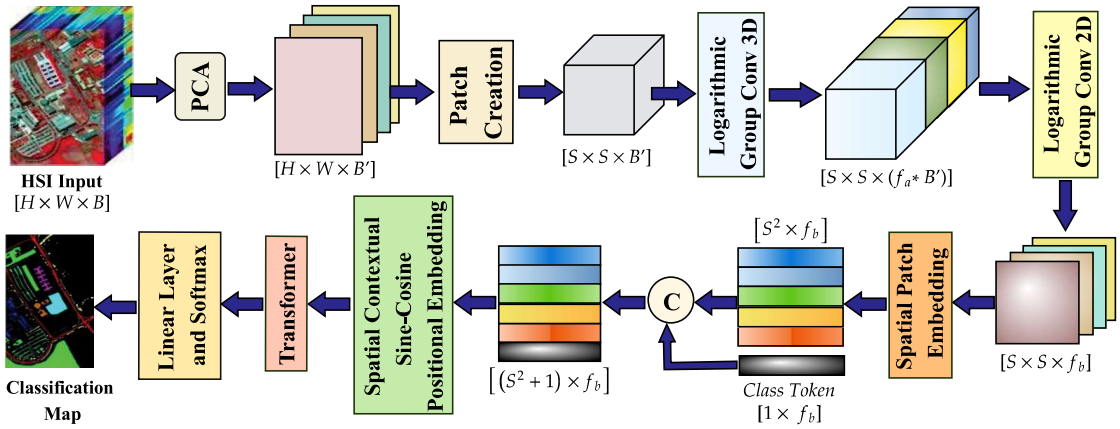


Figure 4.1: **LogGroupformer:** Illustration of proposed lightweight deep network architecture that consists of novel Logarithmic Group 3D and 2D Convolution (**LGConv**) followed by Spatial-Context Patch Embedding **(SCPE)**, novel Spatial Contextual Sine-Cosine Postional Embedding (**(SC)^{2}PosEmbed**) and ViT for HSI classification.

4.4 Proposed Methodology

The proposed LogGroupFormer framework is designed to enhance HSI classification by combining the strengths of CNNs and ViTs in a lightweight and efficient manner. Figure 4.1 illustrates the overall workflow of the framework, which consists of four core modules: (a) Logarithmic Group 3D and 2D Convolution (LGConv), (b) Spatial-Context Patch Embedding (SCPE), (c) Spatial Contextual Sine-Cosine Positional Embedding ($(SC)^2$PosEmbed), and (d) Vision Transformer (ViT).

In LogGroupFormer, feature selection begins with PCA-based band reduction to remove spectral redundancy. The reduced cube is divided into 3D spectral–spatial patches,

which are processed by LGConv. Logarithmic 3D convolutions extract local spectral–spatial features, and Logarithmic 2D convolutions refine spatial cues, yielding compact and discriminative local representations. These features are then tokenized for the ViT, which handles global contextual learning. The methodology can be summarized in the following steps:

1. **HSI Representation:** Represent HSI as a hypercube $\mathbf{I} \in \mathbb{R}^{[H \times W \times B]}$, where H, W, and B denote height, width, and number of spectral bands.
2. **Dimensionality Reduction:** Apply PCA to reduce spectral redundancy while retaining essential information. The reduced cube is denoted as $\mathbf{I_p} \in \mathbb{R}^{[H \times W \times B']}$.
3. **Patch Creation:** Segment the reduced HSI cube into structured 3D spectral–spatial patches to prepare for convolutional processing.
4. **LGConv:** Apply LGConv, consisting of group-wise 3D and 2D convolutions with logarithmic kernel scaling. This reduces the parameter count and FLOPs while preserving spectral–spatial integrity.
5. **SCPE:** Transform convolutional feature maps into structured tokens that maintain local spatial–spectral relationships.
6. $(SC)^2 PosEmbed$**:** Enhance spatial awareness by embedding sine–cosine based positional and contextual information into the tokens.
7. **Global Feature Extraction:** Pass the enriched tokens into the ViT to capture long-range dependencies and global spectral–spatial interactions.
8. **Classification:** Use a FC layer followed by softmax activation for the final classification of objects/pixels. The model is trained with an optimized loss function.

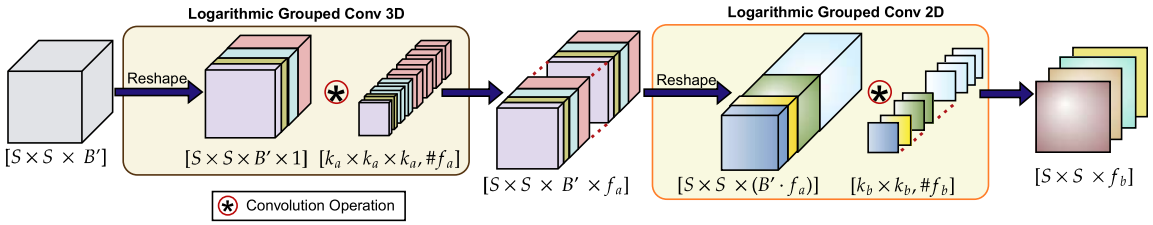


Figure 4.2: Illustration of The Logarithmic Group 3D and 2D Convolution (LGConv).

4.4.1 Logarithmic 3D and 2D Convolution Module (LGConv)

In this subsection, we provide details of the Logarithmic 3D and 2D Convolution Module (LGConv). It is a key component of LogGroupFormer for efficient fine-grained spectral-spatial feature extraction. LGConv consists of three primary sub-modules: Input Patch Extraction, Logarithmic 3D convolution (LGConv3D) to capture correlated

Algorithm 4.1: Input Patch Extraction

Input : HSI cube $\mathbf{I} \in \mathbb{R}^{[H \times W \times B]}$. (H, W), K_f: Height and Width of I_{patch} and Filter. B: Number of spectral bands.
Output: Patch tensor $I_{patch} \in \mathbb{R}^{[S \times S \times B^{'}]}$.

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

spectral-spatial features in 3D data, and Logarithmic 2D convolution (LGConv2D) for important spatial patterns in 2D images. Our proposed LGConv extends group convolution [160] by incorporating a logarithmic filter grouping strategy for efficient spectral-spatial feature extraction. Unlike conventional CNNs, which apply uniform convolutions across spectral and spatial channels, LGConv employs logarithmic spectral partitioning, as illustrated in Figure 4.2.

Algorithm 4.2: LGConv3D Sub-Module

Input : Patch tensor $I_{patch} \in \mathbb{R}^{[S \times S \times B^{'}]}$, Kernel size $K_a \times K_a \times K_a$, Number of filters f_a
Output: Feature map $Y_{LGConv3D} \in \mathbb{R}^{[S \times S \times B^{'} \times f_a]}$

Partition: Spectral Bands: $B_i' \leftarrow [B'/2, B'/4, B'/8, B'/8]$, for $g = 4$ Logarithmic groups
Partition: Number of Filters: $f_{a_i} \leftarrow [f_a/2, f_a/4, f_a/8, f_a/8]$, for $g = 4$ Logarithmic groups
Reshape Operation: $I^{'}_{patch} \leftarrow Reshape(I_{patch}), \quad I^{'}_{patch} \in \mathbb{R}^{[S \times S \times B^{'} \times 1]}$
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: $Y_i \leftarrow$ **3DConv**(Z_i, K_a, f_{a_i})
  Store Output: $\mathcal{Y} \leftarrow Y_i$
Concatenate Along Spectral Dimension
  $Y_{LGConv3D} \leftarrow$ **Concat**$[\mathcal{Y}, \dim = 1]$
Return: $Y_{LGConv3D}$

The HSI input is represented as $\mathbf{I} \in \mathbb{R}^{[H \times W \times B]}$, where H and W denote spatial dimensions, and B represent spectral bands. Initially, $\mathbf{I}$ undergoes a dimension reduction process using PCA while preserving critical features. A patch extraction module then segments $\mathbf{I}$ into $I_{\text{patch}} \in \mathbb{R}^{[S \times S \times B^{'}]}$, where $S \times S$ is the spatial size and $B^{'}$ the reduced spectral bands. The in-depth procedure is detailed in Algorithm 4.1. Secondly, we fed the I_{patch} as an input to the LGConv3D sub-module to enhance spectral-spatial feature extraction. The detailed procedure of extracting enhanced features while si-

multaneously reducing the complexity overhead is outlined in Algorithm 4.2. A key feature of this algorithm is the group convolution strategy, which partitions the spectral bands B' and filters f_a into four groups of logarithmic scale: $[B'/2, B'/4, B'/8, B'/8]$ and $[f_a/2, f_a/4, f_a/8, f_a/8]$, respectively. Finally, the obtained feature map $Y_{LGCconv3D}$ from the LGConv3D sub-module is processed by the LGConv2D sub-module to refine spatial feature extraction while reducing computational complexity, as shown in Algorithm 4.3. The given sub-algorithm employs a similar partitioning strategy but is applied to channel bands. In addition, it improves the optimization of spatial feature representation with minimal overhead.

Algorithm 4.3: LGConv2D Sub-Module

Input : Feature map $Y_{LGConv3D} \in \mathbb{R}^{[S\times S\times B^{'}\times f_a]}$
Output: Feature map $Y_{LGConv2D} \in \mathbb{R}^{[S\times S\times f_b]}$

Reshape: $Y^{'}_{LGConv3D} = \text{Reshape}(Y_{LGConv3D})$, where $Y^{'}_{LGConv3D} \in \mathbb{R}^{[S\times S\times (B^{'}\cdot f_a)]}$
Partition: Channels for $g = 4$ logarithmic groups :
  $B'' = (B^{'} \cdot f_a)$
  $B^{''}_i \leftarrow [B^{''}/2, BB^{''}/4, B^{''}/8, B^{''}/8]$,
Partition: Number of filters for $g = 4$ logarithmic groups:
  $f_{b_i} \leftarrow [f_b/2, f_b/4, f_b/8, f_b/8]$
Apply 2D Convolution to each cluster:
for $i = 1$ **to** g **do**
  Extract Cluster: $Z_i \leftarrow Y^{'}_{LGConv3D}[:, :, B^{''}_i]$
  Apply Conv: $Y_i \leftarrow$ **2DConv**(Z_i, K_f, f_{b_i})
  Store Output: $\mathcal{Y} \leftarrow Y_i$
Concatenate Along Channel Dimension
  $Y_{LGConv2D} \leftarrow$ **Concat**$[\mathcal{Y}, \text{dim} = 1]$
Return: $Y_{LGConv2D}$

Table 4.1: Computation cost analysis of LGConv module.

LGConv3D Sub-Module			
Cost Parameters	**Conventional**	**Proposed**	**Difference**
Parameter Count	$\mathcal{P}_{3D} = K_a^3.f_a + f_a$	$\mathcal{P}_{LGConv3D} = K_a^3.f_a + f_a$	$\Delta P_{LGConv3D} = 0$
Computational Overhead	$\mathcal{C}_{3D} = K_a^3.B'.f_a.S^2$	$\mathcal{C}_{LGConv3D} = (11/32)K_a^3.B'.f_a.S^2$	$\Delta\mathcal{C}_{LGConv3D} = (21/32)K_a^3.B'.f_a.S^2$
LGConv2D Sub-Module			
Cost Parameters	**Conventional**	**Proposed**	**Difference**
Parameter Count	$\mathcal{P}_{2D} = K_b^2.B''.f_b + f_b$	$\mathcal{P}_{LGConv2D} = (11/32)K_b^2.B''.f_b + f_b$	$\Delta\mathcal{P}_{HCC2D} = (21/32)K_b^2.B''.f_b + f_b$
Computational Overhead	$\mathcal{C}_{2D} = K_b^2.B''.f_b.S^2$	$\mathcal{C}_{LGConv2D} = (11/32)K_b^2.B''.f_b.S^2$	$\Delta\mathcal{C}_{LGConv2D} = (21/32)K_b^2.B''.f_b.S^2$

The parameter count and computational complexity in CNNs are determined by Equation 4.2, where P_{3D} and C_{3D} denote the total number of parameters and computational overhead, respectively:

$$P_{3D} = K_v \cdot C_{in} \cdot C_{out} + C_{out}, \qquad C_{3D} = K_v \cdot C_{in} \cdot C_{out} \cdot H' \cdot W' \cdot B' \tag{4.2}$$

Here, K_v represents the kernel size, C_{in} and C_{out} denote the number of input and output channels, respectively. The LGConv3D sub-module employs a kernel size of $[K_a \times K_a \times K_a]$, whereas LGConv2D uses $[K_b \times K_b]$. Table 4.1 presents a comparative computational cost analysis of the LGConv module against conventional 3D and 2D convolution approaches. The LGConv3D sub-module retains the same parameter count as standard 3D convolution ($\Delta P_{LGConv3D} = 0$) while achieving a substantial 65.63% reduction in computational overhead ($\Delta \mathcal{C}_{LGConv3D} = (21/32)K_a^3.B'.f_a.S^2$). Similarly, LGConv2D reduces computational overhead and parameters by 65.63%.

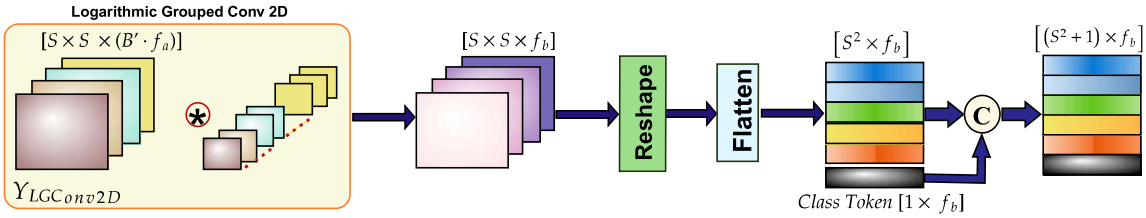


Figure 4.3: Illustration of the spatial-contextual patch Embedding (SCPE) module.

4.4.2 Spatial-Context Patch Embedding (SCPE) module

The SCPE module overcomes the limitations of conventional patch embedding, which prioritizes spectral features while overlooking spatial relationships. Traditional methods divide the input $[S \times S \times f_b]$ into S^2 patches of $[1 \times 1 \times f_b]$, leading to high computational complexity due to $S^2 > f_b$ and ineffective spatial dependency modeling. In contrast, SCPE forms f_b patches of $[S \times S]$, ensuring a balanced spectral-spatial representation with reduced computational overhead. As illustrated in Figure 4.3, SCPE processes $\mathbf{Y}_{\text{LGConv2D}}$ by reshaping, transposing, and flattening it into spectral-spatial patch tokens $[S^2 \times f_b]$. A class token $[1 \times f_b]$ is appended, forming a final token representation of $[(S^2 + 1) \times f_b]$. The transformation is mathematically formulated as:

$$\hat{\mathbf{Y}}_{sp} = \textbf{Flatten}(\textbf{Transpose}(\textbf{Reshape}(Y_{LGConv2D}))), \quad \hat{\mathbf{Y}}_{spc} = \textbf{Concat}[\hat{\mathbf{Y}}_{sp}, C_t], \qquad (4.3)$$

where $\hat{\mathbf{Y}}_{sp}$ is the output correlated spectral-spatial patch tokens $\mathbb{R}^{[S^2 \times f_b]}$. **Concat**$[\cdot, \cdot]$ is the concatenation function of $\hat{\mathbf{Y}}_{spc}$ and class token C_t. $\hat{\mathbf{Y}}_{spc}$ is the class-wise spectral-spatial correlated tokens.

4.4.3 Spatial Contextual-based Sine-Cosine Position Embedding Module

In this section, we introduce a novel S$(SC)^2$*PosEmbed* module comprising two key components: (a) Global Sine-Cosine Positional Embedding (*GSCPosEmbed*) and (b)

Algorithm 4.4: Spatial Contextual-based Sine-Cosine Position Embedding ($(SC)^2PosEmbed$) Module

Input : Feature map P of size $S \times S$, Small constant $\epsilon = 10^{-6}$, Temperature constant $\tau = 10^4$
Output: Spatial positional embedding $I_{SPE} \in \mathbb{R}^{[(S^2+1)\times f_b]}$

Step 1: Global Sine-Cosine Positional Embedding (GSCPosEmbed)
Initialize tensor P of ones: $P \leftarrow \textbf{ones}(S, S)$
Compute cumulative summation along spatial axes:
$X_{embed}(i,j) = P_{ij} \leftarrow \sum_{m=0}^{i} P(m,j),\ Y_{embed}(i,j) = P_{ji} \leftarrow \sum_{n=0}^{j} P(i,n)$
Normalize positional values:
$X'_{embed_{ij}} \leftarrow \frac{2\pi \cdot P_{ij}}{Max(P_i)+\epsilon},\ Y'_{embed_{ji}} \leftarrow \frac{2\pi \cdot P_{ji}}{Max(P_j)+\epsilon}$
Channel-wise division of embeddings:
for $k = 0$ **to** $f_b/2 - 1$ **do**
    $\hat{X}_{em}(i,j,k) \leftarrow \frac{X'_{embed_{ij}}}{\tau^{(2k/f_b)}},\quad \hat{Y}_{em}(i,j,k) \leftarrow \frac{Y'_{embed_{ij}}}{\tau^{(2k/f_b)}}$
Apply sine and cosine transformations:
for $k = 0$ **to** $f_b/2 - 1$ **do**
    if $k \mod 2 == 0$ **then**
        $\hat{X}_{sin} \leftarrow \sin(\hat{X}_{em}),\quad \hat{Y}_{sin} \leftarrow \sin(\hat{Y}_{em})$
    else
        $\hat{X}_{cos} \leftarrow \cos(\hat{X}_{em}),\ \hat{Y}_{cos} \leftarrow \cos(\hat{Y}_{em})$
Concatenate sine and cosine embeddings:
$Pos_X \leftarrow \textbf{Concat}[\hat{X}_{sin}, \hat{X}_{cos}],\quad Pos_Y \leftarrow \textbf{Concat}[\hat{Y}_{sin}, \hat{Y}_{cos}]$
Step 2: Spatial Contextual Positional Embedding (SCPosEmbed)
Concatenate Pos_X and Pos_Y:
$Pos_{XY} \leftarrow \textbf{Concat}[Pos_X, Pos_Y]$
Apply permutation and 1×1 convolution:
$\hat{I}_{XY} \leftarrow \textbf{Transpose}[\textbf{Reshape}[\textbf{2DConv}(\textbf{Permute}[Pos_{XY}])]]$
Concatenate class token C_{T_2}:
$I_{SPE} \leftarrow \textbf{Add}[\textbf{Concat}[\hat{I}_{XY}, C_{T_2}], \hat{Y}_{spc}]$
Return: I_{SPE}

Spatial Contextual Positional Embedding ($SCPosEmbed$), as illustrated in Figure 4.4. The GSCPosEmbed module initializes a positional tensor **P** as a reference matrix for storing spatial position information. As outlined in Algorithm 4.4 (Steps 1–2), pixel positions in X_{embed} and Y_{embed} are computed and normalized between 00 and 2π to maintain consistency across image sizes. A small constant $\epsilon = 10^{-6}$ prevents numerical instability. A temperature constant $\tau = 10^4$ regulates encoding frequency for smooth positional transitions for high-resolution HSIs. After normalization, sine and cosine transformations generate high-dimensional embeddings $\hat{X}_{sin}$, $\hat{X}_{cos}$, $\hat{Y}_{sin}$, and $\hat{Y}_{cos}$. The sine function captures periodic patterns to distinguish spatial positions, whereas the cosine function introduces a phase shift for improved differentiation. These transformed values are concatenated into Pos_{XY}, a multi-channel representation integrating horizontal and vertical encodings, as illustrated in Figure 4.4. The SCPosEmbed module

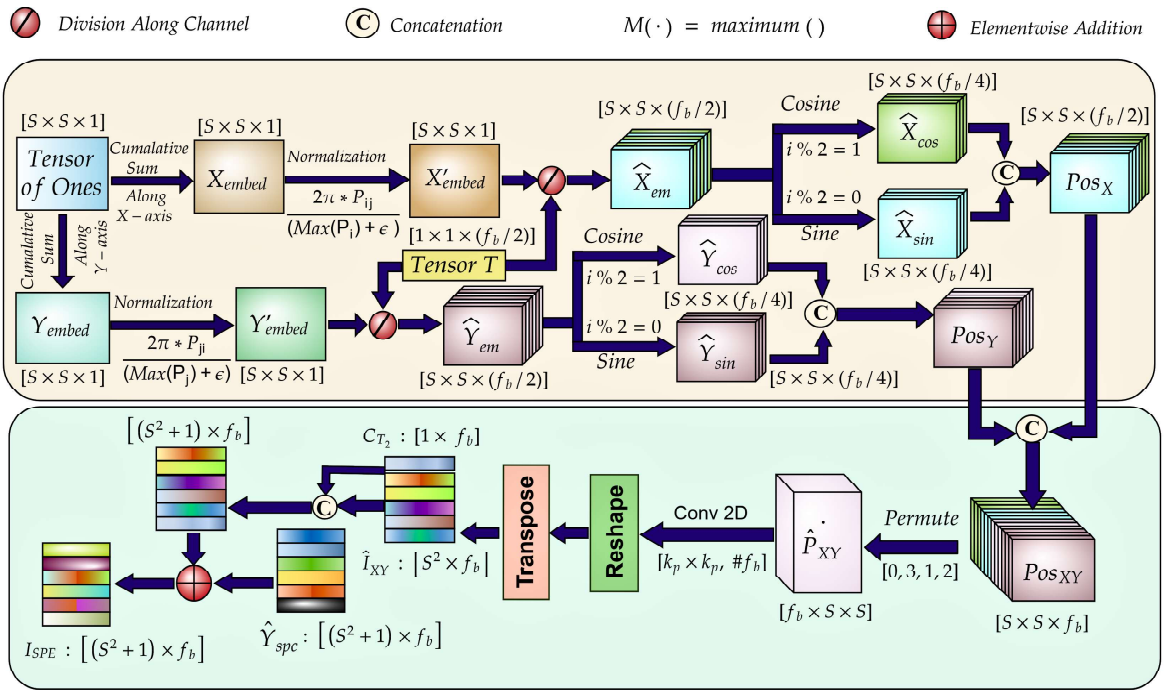


Figure 4.4: Illustration of Spectral-Spatial Contextual-based Sine-Cosine Position Embedding Module ($(SC)^2PosEmbed$) for extracting the positional embedding with respect to spatial and its contextual information.

enhances positional encoding by integrating local spatial dependencies. The concatenated embedding Pos_{XY} is permuted and processed via a convolutional layer to extract regional spatial relationships, yielding $\hat{I}_{XY}$. This output is reshaped, transposed, and combined with a class token C_{T_2} to introduce a global reference point, improving model learning. The final position-encoded tokens I_{SPE} encapsulate absolute positional encoding and spatial context.

4.4.4 Transformer Encoder Module

In this subsection, we present the transformer encoder, which consists of MHSA and an MLP. MHSA captures complex patterns while preserving key features by projecting input data into multiple subspaces, enabling parallel processing across attention heads. As illustrated in Figures 4.5 (a) and (b), the concatenated outputs are mapped to enhance feature representation with minimal computational overhead.

As illustrated in Figure 4.5(c) and detailed in Algorithm 4.5, the MHSA process begins with the input I_{SPE}, which undergoes linear transformations to generate query (Q), key (K), and value (V) matrices. These matrices are then split into nn attention heads

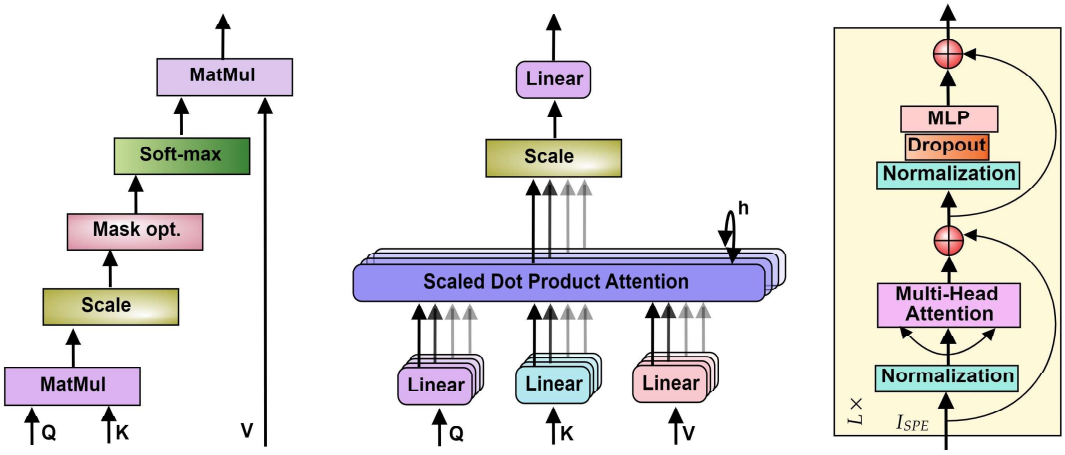


Figure 4.5: Illustration of the attention mechanism in transformers and the transformer encoder (a) Self-attention module. (b) Multi-head attention and (c) Transformer Encoder.

Algorithm 4.5: Transformer Encoder with Multi-Head Self-Attention (MHSA)

Input: Input Token Matrix $I_{SPE} \in \mathbb{R}^{[(S^2+1)\times f_b]}$
Output: Predicted Class Probabilities $\hat{Y}$

Step 1: Compute Query, Key, and Value Matrices
  $Q \leftarrow W_q I_{SPE}, \quad K \leftarrow W_k I_{SPE}, \quad V \leftarrow W_v I_{SPE}: \quad (Q, K, V) \in \mathbb{R}^{[S\times S\times f_b]}$
Step 2: Multi-Head Segmentation
  $Q = [Q_1, Q_2, \ldots, Q_h], \quad K = [K_1, K_2, \ldots, K_h], \quad V = [V_1, V_2, \ldots, V_h]$
  $(Q_i, K_i, V_i) \in \mathbb{R}^{[S\times S\times f_b/h]}$
Step 3: Compute Attention for Each Head
for $i = 1$ *to* h **do**
  $\mathcal{Z}_i \leftarrow Softmax(Q_i K_i^T/\sqrt{f_K})V_i$
end
Step 4: Multi-Head Attention Aggregation
  $MHSA_{output}(Q_i, K_i, V_i) \leftarrow$ **Concat**$[\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_h]W$
Step 5: Feed-Forward Network (MLP Block)
  $O^{'} \leftarrow MHSA_{output} + I_{SPE}$
  $F_1 \leftarrow GeLU(Dropout(LayerNormalization(O^{'}))W_1 + b_1)$
  $Final\_Feature: F_2 \leftarrow F_1 + O'$
Step 6: Classification Layer
  $O_{fc} \leftarrow F_2 W_f + b_f$
  $\hat{Y} \leftarrow Softmax(O_{fc})$
Return: Predicted Class Probabilities $\hat{Y}$

as $Q = [Q_1, Q_2, \ldots, Q_n]$, $K = [K_1, K_2, \ldots, K_n]$, and $V = [V_1, V_2, \ldots, V_n]$, facilitating parallel computation. Each head independently computes scaled dot-product attention as $\mathcal{Z}_i = Softmax(Q_i K_i^T/\sqrt{f_K})V_i$, refining spatial-spectral dependencies. The resulting attention outputs are concatenated and mapped through a learnable parameter matrix.

The aggregated output then passes through normalization, dropout, and an MLP layer to enhance feature representation. Finally, the processed features are fed into a classifier comprising a linear layer and a softmax function, ensuring accurate class probability estimation for HSI data.

4.5 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 comparisons with SOTA methods. Specifically, we apply our method and SOTA techniques to four HSI datasets, providing a comprehensive performance comparison. This evaluation demonstrates the efficacy and robustness of our approach across various scenarios.

4.5.1 Dataset Description and Training Details

We have 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 4.2 has presented the training, validation, and testing sample distribution, class names, and color codes, ensuring a rigorous evaluation of LogGroupFormer's generalizability and efficiency in HSI classification. We have allocated different proportions of labeled samples per dataset for a balanced evaluation. For PU and LK, 1% per class was assigned to training, 1% to validation, and 98% to testing. For IP, 5% per class was used for training and validation, with 90% for testing. Similarly, HU had 2% per class for training, 2% for validation, and 96% for testing.

4.5.2 Experimental settings and Implementation Details

We have evaluated our proposed LogGroupTransformer model for HSI classification using accuracy metrics and computational cost analysis to ensure a comprehensive assessment of performance and efficiency. 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}, which are calculated in terms of FLOPs. At the same time, efficiency has been assessed by recording Tr and Te.

All experiments have been conducted in a PyTorch environment on Google Colab. The computational setup included an Intel Xeon CPU, 26 GB of RAM, and an NVIDIA

Table 4.2: IP, PU HU, and LK Dataset description with number of training, validation, and test samples per class and overall used within the model.

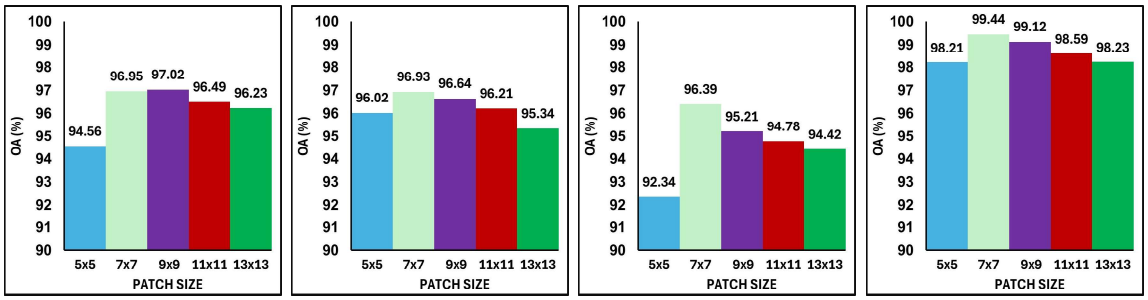
IP Dataset						HU Dataset					
No.	Color	Class	Train	Valid	Test	No.	Color	Class	Train	Valid	Test
C01		Alfalfa	2	2	42	C01		Healthy_grass	25	25	1201
C02		Corn-NT	71	71	1286	C02		Stressed_grass	25	25	1204
C03		Corn-MT	41	41	748	C03		Synthetic_grass	13	13	671
C04		Corn	11	11	215	C04		Trees	24	24	1196
C05		Grass-P	24	24	434	C05		Soil	24	24	1194
C06		Grass-T	36	36	658	C06		Water	6	6	313
C07		Grass-PM	1	1	26	C07		Residential	25	25	1218
C08		Hay-W	23	23	431	C08		Commercial	24	24	1196
C09		Oats	1	1	18	C09		Road	25	25	1202
C10		Soybean-NT	48	48	876	C10		Highway	24	24	1179
C11		Soybean-MT	122	122	2212	C11		Railway	24	24	1187
C12		Soybean-C	29	29	534	C12		Parking_Lot1	24	24	1185
C13		Wheat	10	10	186	C13		Parking_Lot2	9	9	451
C14		Woods	63	63	1140	C14		Tennis_court	8	8	412
C15		Buildings-GTD	19	19	348	C15		Running_track	13	13	634
C16		SS-Towers	4	4	85	–	–	–	–	–	–
PU Dataset						LK Dataset					
No.	Color	Class	Train	Valid	Test	No.	Color	Class	Train	Valid	Test
C01		Asphalt	66	66	6499	C1		Corn	345	345	33821
C02		Meadows	186	186	18277	C2		Cotton	83	83	8208
C30		Gravel	20	20	2059	C3		Sesame	30	30	2971
C04		Trees	30	30	3004	C4		Broad-Leaf soybean	632	632	61948
C05		Painted_metal_sheets	13	13	1319	C5		Narrow-Leaf soybean	41	41	4069
C06		Bare_soil	50	50	4929	C6		RIce	118	118	11618
C07		Bitumen	13	13	1304	C7		Water	670	670	65716
C08		Self_blocking_bricks	36	36	3610	C8		Road and Houses	71	71	6982
C09		Shadows	9	9	929	C9		Mixed weed	52	52	5127

A100 Tensor Core GPU with 40 GB of high-bandwidth memory. The Adam optimizer was used for optimization, with an initial learning rate of 1×10^{-6}. The training was performed in mini-batches of size 64, with each dataset undergoing 100 training epochs.

4.5.3 Hyperparameter Sensitivity Analysis

A comprehensive analysis of model hyperparameters has been conducted to assess their impact on classification performance and training efficiency. The key factors examined have included the patch size $[S \times S]$ for spatial feature extraction, filter count in LGConv's convolutional layers for feature representation, the number of groups in logarithmic convolution, which has been responsible for channel-wise information aggregation, and learning rate for optimization convergence and model stability.

Impact of Patch Size on Classification Performance: Figure 4.6 demonstrates the effect of varying patch sizes on the OA of the proposed model. The analysis identifies optimal PSs for different datasets:$[7 \times 7]$ for PU, HU, and LK, while $[9 \times 9]$ for IP. PS plays a crucial role in classification performance, as smaller patches enhance fine-grained spatial details, benefiting PU, HU, and LK datasets. In contrast, the IP dataset,

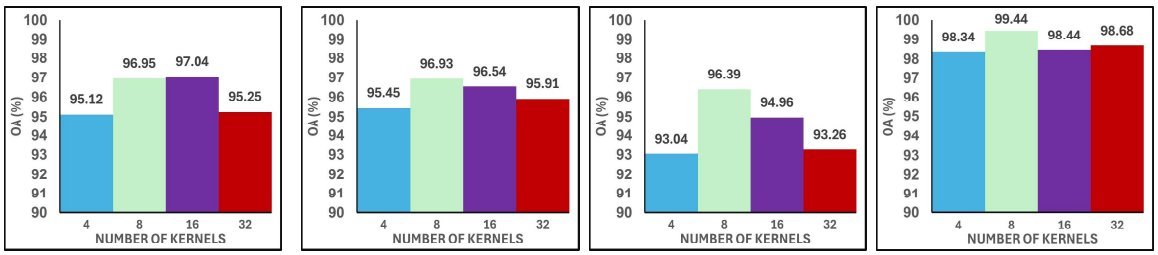


(a) IP (b) PU (c) HU (d) LK

Figure 4.6: Assessment of OA(%) for LogGroupFormer with Varying Patch Sizes.

characterized by larger spatial structures, achieves better accuracy with a $[9 \times 9]$ patch, capturing broader contextual information. Selecting an appropriate PS is crucial for balancing local feature extraction and contextual awareness, ultimately improving HSI classification accuracy.

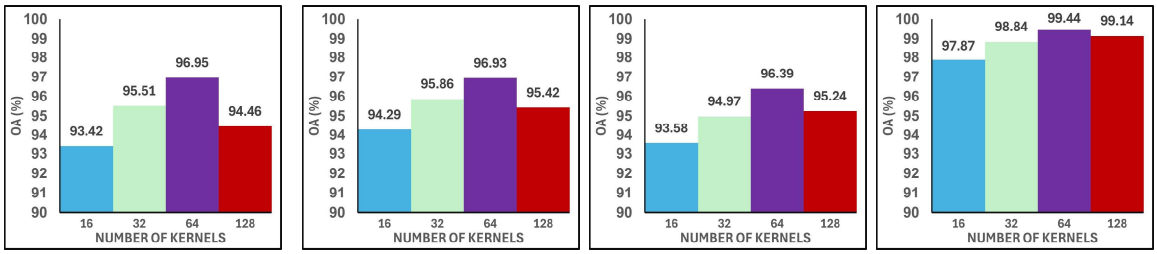
Impact of Filter Count in LGConv3D and LGConv2D Layers An ablation study has been conducted to evaluate the impact of kernel count on the performance of LGConv3D and LGConv2D layers. For LGConv3D, kernel sizes of 4, 8, 16, and 32 have been tested, while LGConv2D has been evaluated with 16, 32, 64, and 128 kernels. As shown in Figures 4.7 and 4.8, LGConv3D has performed optimally with 8 kernels for PU, HU, and LK datasets, whereas the IP dataset has benefited from 16 kernels, likely due to its complex spatial structures. In contrast, LGConv2D has consistently performed best with 64 kernels across all datasets, balancing feature extraction and computational efficiency.



(a) IP (b) PU (c) HU (d) LK

Figure 4.7: Assessment of OA(%) with Varying Filter Counts in LGConv3D Layer.

These results highlight the importance of kernel selection in convolutional layers. While LGConv3D requires dataset-specific tuning, LGConv2D exhibits a more stable optimal configuration, reinforcing the need for careful parameter tuning to enhance HSI classi-

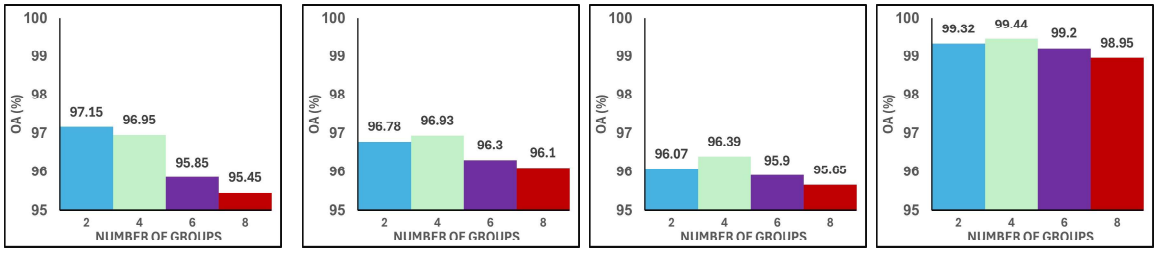


(a) IP (b) PU (c) HU (d) LK

Figure 4.8: Assessment of OA(%) with Varying Filter Counts in LGConv2D Layer.

fication performance for all users.

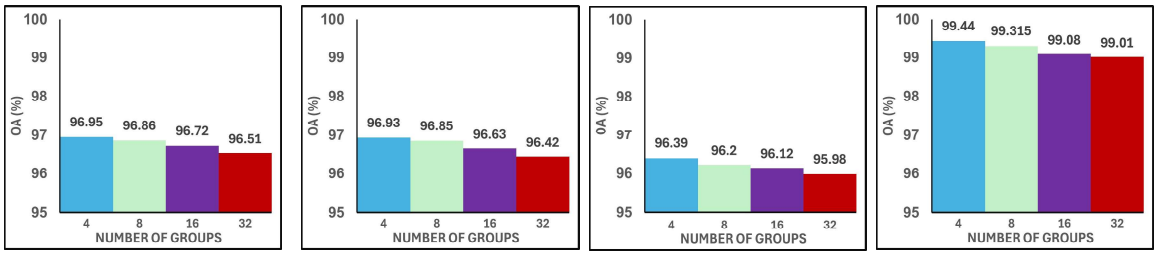
Optimizing Logarithmic Group Divisions for LGConv Layers: We have conducted an in-depth analysis to determine the optimal number of logarithmic group divisions for LGConv3D and LGConv2D layers. We have evaluated LGConv3D with 2, 4, 6, and 8 groups, while LGConv2D has been tested with 4, 8, 16, and 32 groups. As illustrated in Figures 4.9 and 4.10, we have found that the optimal configuration for both LGConv3D and LGConv2D is 4 groups across all datasets. For LGConv3D, configurations with 2, 6, and 8 groups have resulted in lower accuracy due to an imbalance in trainable and non-trainable parameters. Similarly, we have observed that increasing the number of groups in LGConv2D has led to a suboptimal parameter ratio, hindering learning and generalization. These findings have highlighted the importance of selecting an appropriate number of logarithmic group divisions to balance computational complexity and classification accuracy.



(a) IP (b) PU (c) HU (d) LK

Figure 4.9: Assessment of OA(%) for LogGroupFormer with Varying Number of Logarithmic Groups in the LGConv3D Layer.

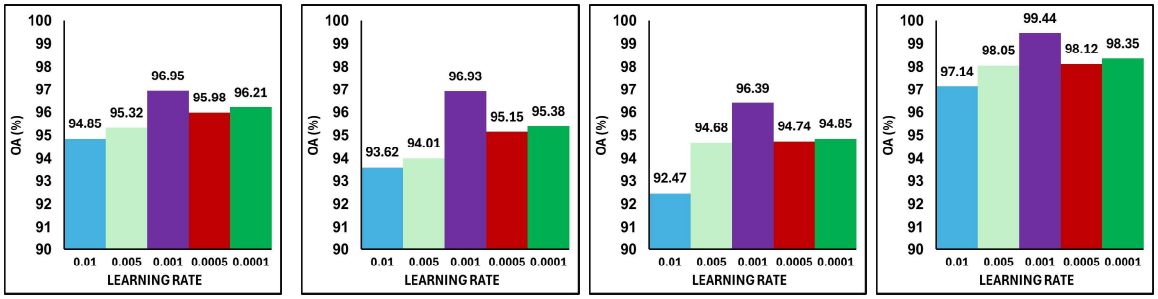
Impact of learning rates (LR) We conducted ablation studies to analyze the impact of different LR on model performance, evaluating values of 0.01, 0.005, 0.001, 0.0005, and 0.0001. As illustrated in Figure 4.11, an LR of 0.001 consistently achieves the



(a) IP (b) PU (c) HU (d) LK

Figure 4.10: Assessment of OA(%) for LogGroupFormer with Varying Number of Logarithmic Groups in the LGConv2D Layer.

highest OA across all datasets. This LR provides an optimal balance, enabling efficient model convergence while preventing overshooting, thereby improving classification precision and stability. The results highlight the critical role of hyperparameter tuning in HSI classification to achieve optimal performance.



(a) IP (b) PU (c) HU (d) LK

Figure 4.11: Assessment of OA(%) for LogGroupFormer with Varying Learning Rates.

4.5.4 Ablation Studies

This section presents ablation studies to investigate the significance of each architectural component in the proposed model. We perform controlled experiments by modifying or removing specific modules to understand their individual contributions. The impact of replacing advanced operations with conventional alternatives is also explored. These analyses help validate the effectiveness and necessity of each component in enhancing classification performance.

Component Evaluation: We have conducted ablation studies by systematically modifying model components across all datasets to assess effectiveness. Six configurations have been evaluated, and their impact on OA is analyzed in detail in Table 4.3. The

model has been deconstructed into five components: LGConv3D, LGConv2D, SCPE, $(SC)^2PosEmbed$, and the Transformer encoder. The absence of an element is marked in ×, while replacements are in ×. $(SC)^2PosEmbed$ has been replaced with simple position embedding, and SCPE has been replaced with basic patch embedding, omitting either LGConv2D or LGConv3D. Removing the Transformer encoder has consistently resulted in the lowest OA, highlighting its crucial role in feature learning.

Table 4.3: Ablation Study of the Components in the LogFormer Model

No. of	Components					Datasets			
Cases	LGConv3D	LGConv2D	$(SC)^2PosEmbed$	$ScPE$	Transformer	IP	PU	HU	LK
1	×	✓	✓	✓	✓	94.18	95.05	93.91	98.04
2	✓	×	✓	✓	✓	95.32	94.23	94.10	97.22
3	✓	✓	×	✓	✓	96.1	94.22	92.87	97.95
4	✓	✓	✓	×	✓	94.5	94.37	93.05	98.16
5	✓	✓	✓	✓	×	87.35	89.31	91.23	93.05
6	✓	✓	✓	✓	✓	96.95	96.93	96.39	99.44

Among configurations retaining the Transformer encoder, the impact of removing components varies. In the IP dataset, omitting LGConv3D has yielded the lowest OA of 94.18%, while in PU and LK, excluding LGConv2D has resulted in the lowest OA of 94.23% and 97.22%. At HU, removing $(SC)^2PosEmbed$ caused the most significant drop, leading to an OA of 92.87%. These results underscore the importance of hierarchical convolutions, positional embeddings, and Transformer representations in achieving optimal classification accuracy.

Model Analysis with Conventional Convolution Instead of LGConv: We have conducted an ablation study comparing LogGroupFormer with its variant, $(SC)^2Former$, which utilizes conventional convolutional layers. The results, summarized in Table 4.4, highlight key differences in computational efficiency and performance. LogGroupFormer consistently demonstrates reduced training and testing times due to its logarithmic group convolution operations, which enhance computational efficiency. For instance, on the IP dataset, LogGroupFormer required only 13.6 seconds for training and 0.93 seconds for testing, compared to 16.6 seconds and 1.08 seconds for $(SC)^2Former$. Additionally, LogGroupFormer maintains significantly lower parameter counts and computational overheads across all datasets, such as 192.34K parameters and 0.84G FLOPs on IP, compared to 298.79K and 1.51G FLOPs for $(SC)^2Former$. This reduction in computational complexity makes LogGroupFormer more suitable for resource-constrained environments. Despite its efficiency, LogGroupFormer achieves accuracy comparable to $(SC)^2Former$ across all datasets. While $(SC)^2Former$ shows

Table 4.4: Ablation study of the proposed LogGroupFormer with its variant, which uses only conventional convolutional layers, named as $(SC)^2Former$.

Data	Model	Tr(s)	Te(s)	Prm(K)	FS(G)	OA (%)	AA (%)	$\kappa \times 100$
IP	$(SC)^2Former$	16.6	1.08	298.79	1.510	97.44	93.82	97.08
	LogGroupFormer	13.6	0.93	192.34	0.843	96.95	94.49	96.51
PU	$(SC)^2Former$	34.57	18.23	299.12	1.510	97.26	95.60	96.36
	LogGroupFormer	12.32	10.77	191.06	0.843	96.93	95.52	95.92
HU	$(SC)^2Former$	8.94	1.57	299.76	1.510	96.84	97.08	96.51
	LogGroupFormer	7.06	1.75	192.14	0.843	96.39	96.45	96.02
LK	$(SC)^2Former$	67.63	26.02	298.73	1.510	99.55	98.40	99.41
	LogGroupFormer	51.02	21.58	191.26	0.843	99.44	98.16	99.26

slightly better accuracy in certain cases, it comes at the cost of increased computation and longer processing times. LogGroupFormer balances computational efficiency and model performance, making it a strong candidate for HSI classification tasks that require both accuracy and resource optimization.

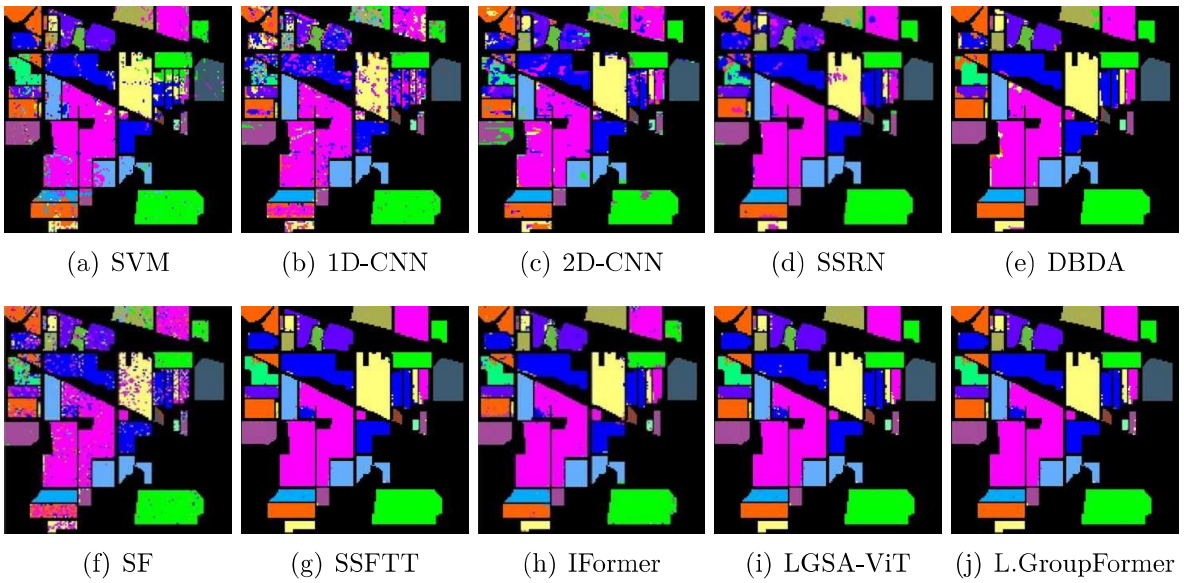
4.5.5 Comparison with State-of-the-Art (SOTA) Methods

To evaluate our methodology, we have conducted comparative experiments against established techniques, categorizing them into three groups: (a) classical ML model SVM [144]; (b) DL architectures such as 1D-CNN [72], 2D-CNN [79], SSRN [113], and DBDA [154]; and (c) Transformer-based models, including SpectralFormer (SF) [119], SSFTT [124], IFormer [163], LGSA-VIT [125], and our proposed LogGroupFormer. The results of these experimental comparisons are detailed in Table 4.5 to 4.8, which present OA, AA, κ, and class-wise accuracies for IP, PU, HU, and LK datasets, with the best results highlighted in bold.

IP Dataset: We have evaluated LogGroupFormer on the IP dataset under limited training data conditions, utilizing only 5% of samples per class for training, 5% for validation, and 90% for testing. As shown in Table 4.5, LogGroupFormer has outperformed SOTA models in OA, AA, and κ, surpassing SVM by 15.83%, which has reinforced the dominance of DL in HSI classification. Compared to convolutional architectures like SSRN and DBDA, LogGroupFormer has demonstrated superior performance, particularly in classifying 'Grass-Pasture,' 'Grass-Trees,' 'Grass-Pasture-Mowed,' and 'Hay-Windrowed,' as illustrated in Figure 4.12. However, it has underperformed on 'Alfalfa' by 25.20% relative to SSFTT and on 'Oats' by 15.28% compared to DBDA, likely due to sample imbalance—'Oats' has only 20 samples. LogGroupFormer also outperforms SF by 5.90% in OA and κ, confirming its ability to integrate local and global spectral-

Table 4.5: 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 dataset

No.	SVM	1D-CNN	2D-CNN	SSRN	DBDA	SF	SSFTT	IFormer	LGSA-VIT	LogGroupFormer
C01	64.61±16.6	34.41±6.32	69.48±18.4	96.85± 7.42	92.85 ± 0.67	25.87 ± 0.58	**99.01 ± 0.6**	78.24 ±2.31	90.48±2.31	73.81 ±1.92
C02	78.69±2.98	80.30±1.18	92.19±2.67	96.14± 3.27	95.75 ± 0.98	87.01 ± 0.45	**97.95 ± 0.8**	**97.75 ±1.67**	90.77 ± 0.44	92.77 ±2.63
C03	68.73±3.68	63.32±4.41	91.51±3.20	97.01± 2.34	96.81 ± 0.85	85.43 ± 0.39	97.85 ± 0.5	**99.46 ±1.92**	97.47 ±0.51	98.56 ± 1.37
C04	61.44±9.34	53.39±5.97	85.19±5.18	95.71± 4.74	94.57 ± 0.56	90.33 ± 0.71	**97.35 ± 0.7**	95.56 ±1.24	91.59 ±0.32	95.41 ±1.98
C05	88.40±3.74	87.34±5.91	96.47±1.43	97.63± 2.04	94.93 ± 0.67	88.50 ± 0.64	97.55 ± 0.6	97.67 ±1.82	91.74 ±0.28	**97.75 ±1.62**
C06	94.48±2.85	98.51±0.71	98.16±1.36	98.04± 1.97	96.77 ± 0.75	96.88 ± 0.29	98.04 ± 0.4	97.86 ±1.58	97.72 ±0.39	**98.36 ±1.47**
C07	75.40±6.41	35.40±25.5	73.80±13.9	49.10± 2.34	90.67 ± 0.69	47.98 ± 0.80	94.83 ± 1.0	83.34 ±4.09	96.15 ±0.66	**100.00 ±0.77**
C08	97.09±2.36	96.65±3.18	98.76±1.20	97.63± 2.41	99.00 ± 0.22	98.92 ± 0.21	98.89 ± 0.5	**100.00 ±0.65**	**100.00±0.10**	**100.00 ±0.92**
C09	41.78±20.8	27.33±21.3	91.55±6.47	29.0±4.1	**93.06 ± 0.77**	37.75 ± 0.52	79.56 ± 2.5	54.43 ±7.36	50.00±0.52	77.78 ±3.28
C10	77.40±4.14	70.70±5.72	95.36±1.17	91.30± 8.24	93.73 ± 0.68	89.15 ± 0.88	94.95 ± 1.3	**97.62 ±1.68**	88.14±0.45	96.20 ±1.63
C11	84.54±1.89	81.65±1.99	95.99±0.95	96.51± 5.13	96.59 ± 0.62	92.83 ± 0.33	97.87 ± 0.6	**98.72 ±1.55**	95.62±0.30	98.10 ±1.51
C12	72.60±4.56	76.75±3.45	84.29±3.37	94.58± 4.06	**96.16 ± 0.81**	80.99 ± 0.75	90.65 ± 1.5	96.00 ±1.87	89.16±0.40	93.41 ±2.09
C13	95.41±3.64	97.43±2.14	98.94±1.16	98.56± 1.72	96.30 ± 0.75	99.32 ± 0.17	98.75 ± 0.4	98.34 ±1.47	**100.00±0.00**	**100.00 ±0.81**
C14	93.92±3.37	94.01±2.04	97.10±2.04	98.27± 1.37	96.35 ± 0.67	94.92 ± 0.46	97.98 ± 0.8	**98.56 ±1.73**	99.65±0.25	99.74 ±1.68
C15	56.12±4.74	62.48±7.36	88.91±5.69	97.05± 2.75	96.14 ± 0.78	60.32 ± 0.94	97.66 ± 0.7	**98.59 ±1.61**	82.23±0.37	95.77 ±2.14
C16	84.18±5.02	83.16±4.06	95.78±5.39	95.21± 6.32	91.39 ± 0.67	98.56 ± 0.3	95.77 ± 0.5	**99.16 ±1.34**	83.33±0.33	94.12 ±2.56
OA	81.12±0.78	80.81±0.75	91.28±0.38	93.11±1.32	95.28±1.63	91.05± 1.84	95.19±2.95	96.36 ±1.92	93.95±0.27	**96.95 ±1.87**
AA	75.67± 1.24	78.43±0.45	86.15± 1.45	89.03±2.42	92.92±2.34	83.42±1.45	92.04 ± 1.78	93.21 ±2.11	90.25±0.41	**94.49 ±2.43**
κ	80.17±2.14	78.16±0.82	90.47±01.24	93.61±2.37	95.04±1.58	90.64±1.03	95.82±2.30	96.12 ±1.83	93.08±0.35	**96.51 ±1.78**



(a) SVM (b) 1D-CNN (c) 2D-CNN (d) SSRN (e) DBDA

(f) SF (g) SSFTT (h) IFormer (i) LGSA-ViT (j) L.GroupFormer

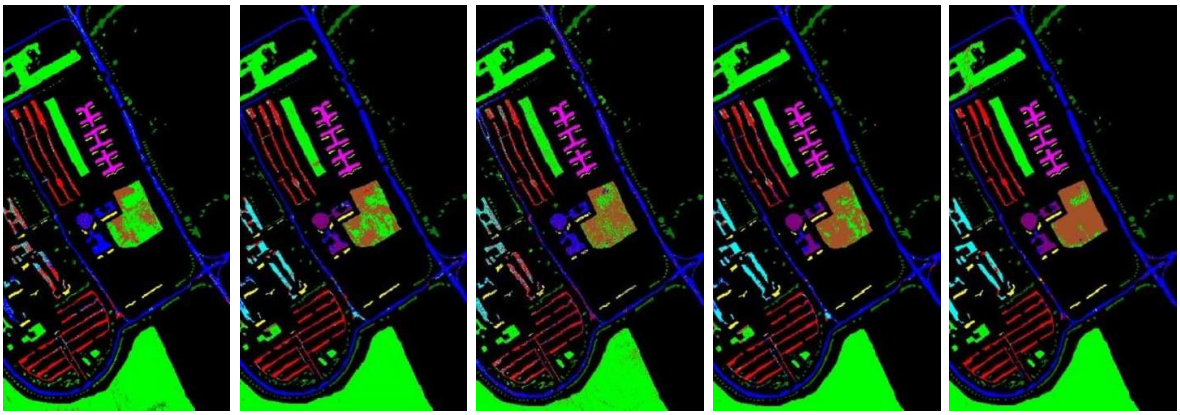
Figure 4.12: Comparison of Visual classification maps of models for IP dataset.

spatial dependencies effectively. It further achieves a 1.76% gain over SSFTT, 0.59% over IFormer, and an overall advantage over LGSA, highlighting its robust spatial-contextual feature extraction. As shown in Fig. 3.7, Vineyard-untrained (, C15) exhibits strong distortion in SVM and 3D-CNN, while Fallow (, C04; , C05) suffers boundary noise in earlier models. Our MDCNN minimizes these distortions, achieving cleaner separation and producing the most reliable classification map.

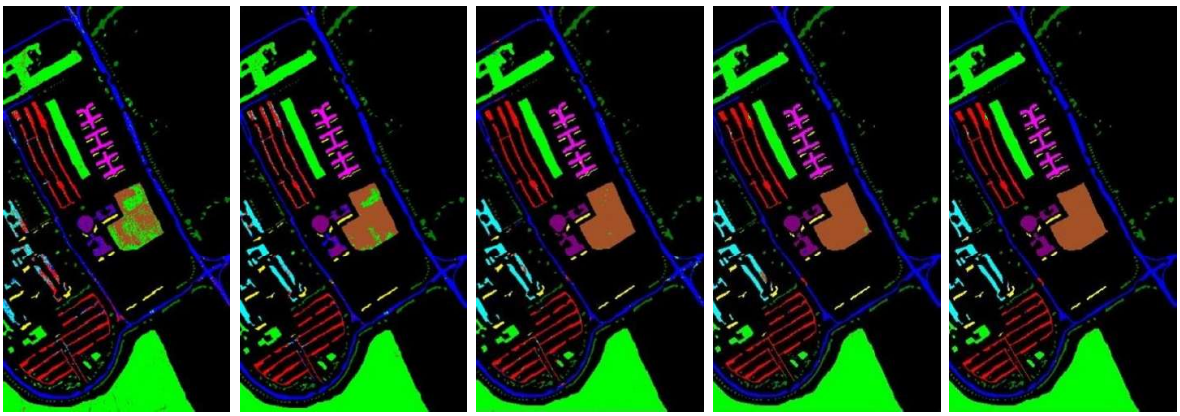
PU Dataset: The LogGroupFormer model has been evaluated on the PU dataset

Table 4.6: Quantitative performance of various classification methods in terms of OA (%), AA (%), and $\kappa \times 100$, along with the accuracies for each class on the PU dataset

No.	SVM	1D-CNN	2D-CNN	SSRN	DBDA	SF	SSFTT	IFormer	LGSA-VIT	LogGroupFormer
C01	89.26 ±1.23	90.32 ±1.62	94.79 ±1.32	92.23 ±2.41	96.53 ±1.92	91.62 ±2.31	96.45 ±1.17	97.20 ±0.42	99.17± 0.35	**97.73 ±1.32**
C02	88.65 ±0.92	82.54 ±2.59	91.00 ±3.16	94.40 ±5.24	97.54 ±2.03	94.48 ±2.65	98.91 ±0.65	99.10 ±0.37	99.66± 0.24	**99.76 ±0.91**
C03	75.27 ±3.42	76.21 ±4.47	86.04 ±3.55	94.55 ±3.60	96.06 ±4.15	**96.56 ±5.28**	86.22 ±2.11	84.50 ±1.21	82.11 ± 0.58	84.60 ±2.71
C04	76.20 ±2.31	84.56 ±1.94	90.80 ±1.64	90.92 ±3.67	89.58 ±4.71	91.24 ±8.42	**98.05 ±0.45**	96.55 ±0.34	92.01 ± 0.31	93.60 ±1.58
C05	94.64 ±1.08	88.24 ±1.42	91.26 ±1.57	99.04 ±0.51	96.24 ±1.39	98.16 ±0.73	98.82 ±0.72	99.40 ±0.29	99.92 ± 0.14	**100.00 ±0.47**
C06	68.43 ±1.76	69.12 ±3.34	79.04 ±3.42	85.20 ±16.7	88.75 ±3.28	79.42 ± 2.85	96.56 ±0.34	96.75 ±0.48	95.27 ± 0.48	**97.35 ±1.19**
C07	91.42 ±0.36	96.83 ±0.53	99.03 ±0.28	91.18 ±0.46	98.64 ±0.35	85.54 ±9.45	98.78 ±0.27	95.12 ±0.20	91.87± 0.33	**99.24 ±0.99**
C08	**98.93** ±0.63	98.25 ±0.77	94.74 ±0.29	91.91 ±0.09	88.36 ±1.08	89.10 ±0.10	84.54 ±1.98	92.80 ±1.79	95.15 ± 0.29	87.74 ±2.24
C09	99.47 ±0.21	99.13 ±0.36	97.40 ±2.64	**99.60 ±1.43**	93.94 ±4.79	99.06 ±0.18	97.62 ±0.81	95.25 ±0.68	99.47 ±0.88	96.66 ± 0.52
OA	87.68 ±0.68	89.34±0.4	93.05 ±0.68	93.62±1.79	94.49±0.56	92.38± 3.60	94.27± 2.12	96.80±0.46	96.57 ± 0.21	**96.93** ±1.51
AA	86.91±1.24	87.24±0.82	91.56 ±0.42	93.22±2.34	93.96±3.2	91.68±2.44	95.11± 1.23	95.18±0.35	94.65 ± 0.41	**95.52 ±1.65**
κ	86.84 ±0.92	88.12±0.57	92.78 ±0.91	93.54±2.35	94.27±0.75	92.08± 3.26	93.38±0.83	95.74± 0.41	95.68 ± 0.22	**95.92 ±1.37**



(a) SVM (b) 1D-CNN (c) 2D-CNN (d) SSRN (e) DBDA



(f) SF (g) SSFTT (h) IFormer (i) LGSA-ViT (j) L.GroupFormer

Figure 4.13: Comparison of Visual classification maps of models for PU dataset.

under limited training samples, with only 1% per class used for training, 1% for validation, and 98% for testing. As shown in Table 4.6, LogGroupFormer has surpassed SVM by 9.25% in OA and has outperformed traditional CNN-based models. The model

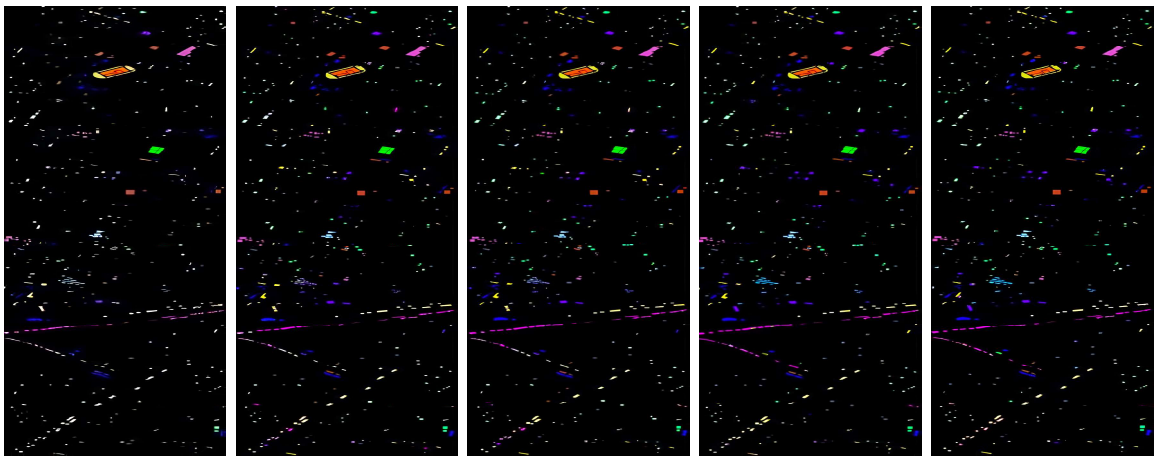
has achieved high accuracy, particularly excelling in "Asphalt," "Meadows," "Bare-soil," "Bitumen," and "Painted-metal-sheets," reaching 100% for the latte, illustrated in Figure 4.13. However, it has underperformed for "Trees" (4.45% below SSFTT) and "Gravel" (15.28% below SF), likely due to limited training samples. Performance has varied with sample availability, excelling in cases of extreme data scarcity or abundance but struggling with moderate training sizes. Additionally, LogGroupFormer has also surpassed LGSA-ViT by 1.16% in OA and 0.24 in κ, demonstrating its ability to integrate local and global features effectively. It has outperformed SSFTT by 2.66% and IFormer by 0.13%, showcasing its superior spatial-contextual representation. As shown in Fig. 4.13, Meadows (, C02) and Gravel (, C03) exhibit strong distortion in SVM, 1D-CNN, and 2D-CNN, with heavy overlap across boundaries. Painted-metal sheets (, C05) and Bitumen (, C07) improve under SSRN and IFormer but still retain residual noise. Our LogGroupFormer achieves the cleanest separation, minimizing distortion in spectrally similar classes and producing the most reliable map.

Table 4.7: Quantitative performance of various classification methods in terms of OA, AA, and κ, along with the accuracies for each class on the HU dataset.

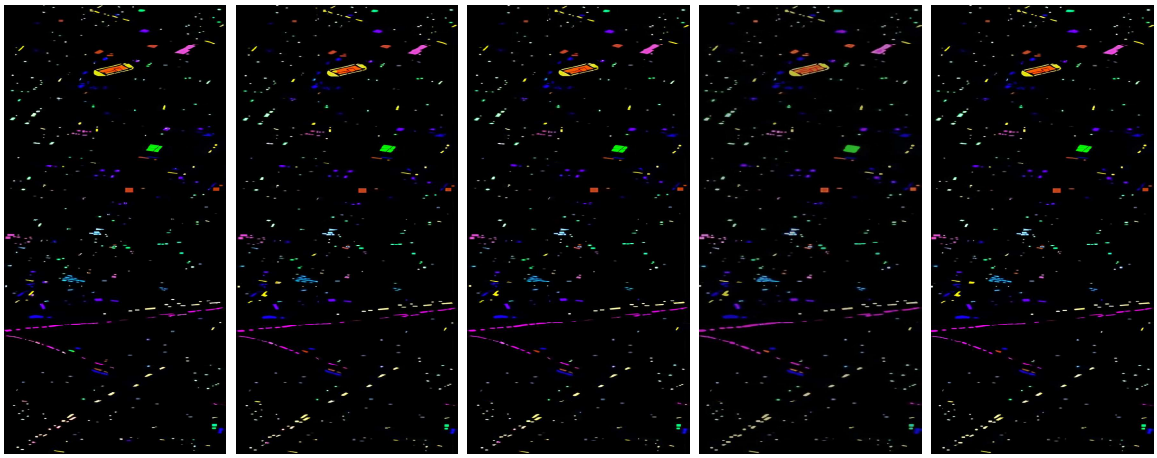
No.	SVM	1D-CNN	2D-CNN	SSRN	DBDA	SF	SSFTT	IFormer	LGSA-VIT	LogGroupFormer
C01	93.9 ±1.57	87.27 ±0.93	85.09 ±0.94	94.00 ±0.42	91.72 ±0.95	91.84 ±1.29	94.37 ±0.74	96.29 ±0.84	95.12 ± 0.45	**96.57 ±1.25**
C02	96.2 ±0.82	98.21 ±0.78	**99.91 ±0.71**	96.92 ±0.65	92.85 ±1.24	95.17 ±0.84	97.40 ±0.48	97.62 ±0.63	97.27 ± 0.53	98.86 ±0.78
C03	91.65 ±0.94	**100.00 ±0.51**	77.23 ±0.48	97.56 ±0.37	**100.00 ±0.53**	94.82 ±0.62	**100.00 ±0.22**	**100.00 ±0.28**	99.72 ± 0.34	**100.00 ±0.12**
C04	96.6 ±0.78	92.99 ±0.62	97.73 ±0.62	94.52 ±0.75	96.85 ±0.63	92.55 ±1.12	**98.61 ±0.65**	94.96 ±0.71	94.81± 0.49	97.82 ±0.45
C05	94.76 ±0.63	97.35 ±0.67	99.53 ±0.77	96.91 ±0.29	96.46 ±0.47	98.33 ±0.75	**100.00 ±0.29**	**100.00 ±0.42**	99.10 ± 0.41	99.92 ±0.65
C06	92.88 ±0.91	95.10 ±0.82	92.31 ±0.83	87.58 ±0.84	**98.51 ±0.72**	74.85 ±2.10	97.18 ±0.83	88.93 ±0.98	91.28 ± 0.59	96.87 ±1.03
C07	84.27 ±0.72	77.33 ±0.39	92.16 ±0.56	92.15 ±0.68	87.58 ±1.10	81.23 ±1.87	90.10 ±1.27	91.31 ±0.89	92.80 ± 0.72	**93.97 ±0.89**
C08	81.51 ±0.49	51.38 ±0.24	73.39 ±0.42	81.65 ±0.53	89.03 ±0.92	79.93 ±1.21	89.99 ±1.34	**90.32 ±0.75**	91.43 ± 0.64	88.11 ±1.56
C09	64.71 ±0.61	27.95 ±0.16	86.31 ±0.73	**92.91 ±0.97**	89.60 ±0.89	80.98 ±1.43	85.41 ±1.69	90.18 ±0.82	92.12 ± 0.68	92.26 ±0.47
10	79.72 ±0.58	90.83 ±0.72	43.73 ±0.29	89.73 ±0.45	88.32 ±1.10	85.63 ±0.92	94.35 ±0.56	95.37 ±0.55	96.82 ± 0.41	**98.00 ±0.91**
C11	83.19 ±0.67	79.32 ±0.53	87.00 ±0.68	93.85 ±0.81	90.78 ±0.74	75.54 ±2.01	96.28 ±0.77	95.29 ±0.73	98.09 ± 0.32	**98.68 ±0.72**
C12	67.78 ±0.55	76.56 ±0.49	66.28 ±0.57	89.86 ±0.49	**92.57 ±0.67**	85.88 ±1.55	89.24 ±1.42	92.49 ±0.67	93.42 ± 0.55	87.69 ±1.38
C13	64.01 ±0.44	69.47 ±0.37	90.18 ±0.79	91.98 ±0.38	88.77 ±0.91	79.77 ±2.14	78.70 ±2.31	94.89 ±0.48	94.50 ± 0.43	**98.70 ±0.64**
C14	89.14 ±0.79	99.19 ±0.81	90.69 ±0.62	92.74 ±0.63	94.49 ±0.59	85.75 ±1.39	**100.00 ±0.25**	99.68 ±0.26	**100.00 ± 0.00**	99.52 ±0.33
C15	88.06 ±0.88	98.10 ±0.79	77.80 ±0.51	93.88 ±0.52	96.20 ±0.66	95.02 ±0.77	**100.00 ±0.33**	**100 ±0.34**	**100.00 ± 0.00**	99.85 ±0.57
OA	77.58± 1.11	80.04 ±0.58	83.72 ±0.65	91.39 ± 0.65	92.72 ±0.54	84.97±2.73	93.94 ±0.92	94.86 ±0.53	95.28 ± 0.38	**96.39 ±1.11**
AA	84.55 ± 1.23	82.74 ±0.67	84.35 ±0.49	92.41±0.45	92.91± 0.49	86.48±0.94	94.11 ±0.63	95.15± 0.48	95.76 ± 0.42	**96.45 ±0.94**
κ	78.56 ± 1.19	78.35 ±0.45	82.31 ±0.58	91.02±0.53	92.34±0.60	84.96±0.79	93.45 ±0.81	94.62±0.58	95.76 ± 0.42	**96.02 ±1.20**

HU Dataset: LogGroupFormer has been evaluated on the HU dataset using 2% training, 2% validation, and 96% testing samples. As shown in Table 4.7, models like SVM, 1D-CNN, and 2D-CNN, relying only on spectral or spatial features, have led to inefficient fusion and information loss. While SSRN and DBDA incorporate spatial-spectral integration, their high complexity limits efficiency. Although SF, SSFTT, IFormer, LGSA-ViT, and LogGroupFormer adopt streamlined architectures, SF has shown lower accuracy for "Water" (74.85%) and "Railway" (75.54%). LogGroupFormer has achieved the highest accuracy across multiple categories, including 96.57% for "Healthy Grass", 100% for "Synthetic Grass," and 98.70% for "Parking-Lot2," showcasing superior clas-

sification capability. Next, while LGSA-ViT has produced competitive results, it has fallen short of LogGroupFormer. As shown in Figure 4.14, LogGroupFormer has effectively captured local and global contextual information, significantly reducing misclassification rates. As illustrated in Fig. 4.14, Commercial (, C08) and Road (, C09) exhibit strong distortion in earlier CNN-based models, with scattered noise and boundary confusion. Residential (, C07) also appears fragmented in several maps. Our LogGroupFormer provides cleaner separation, minimizing distortion and preserving spatial homogeneity across these challenging classes, resulting in the most reliable classification map.



(a) SVM (b) 1D-CNN (c) 2D-CNN (d) SSRN (e) DBDA



(f) SF (g) SSFTT (h) IFormer (i) LGSA-ViT (j) L.GroupFormer

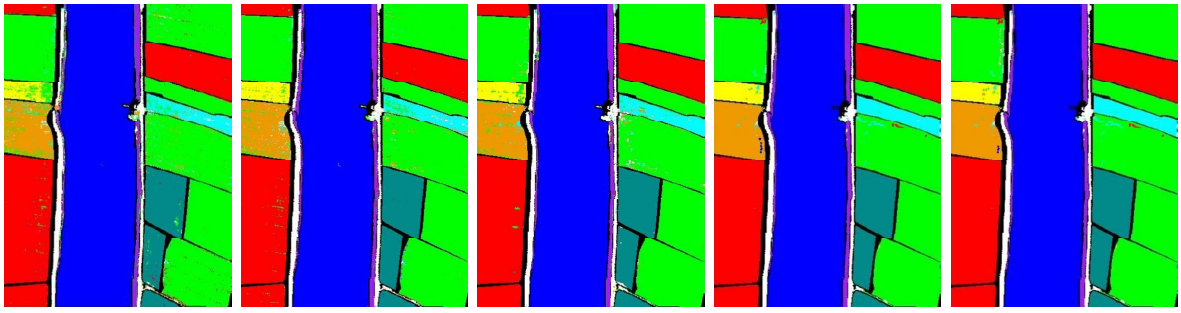
Figure 4.14: Comparison of Visual classification maps of models for HU dataset.

LK Dataset: The LK dataset, captured via UAV, offers higher spatial resolution and lower noise, resulting in superior classification performance across all methods. As shown in Table 4.8 and Figure 4.15, although multiple approaches have

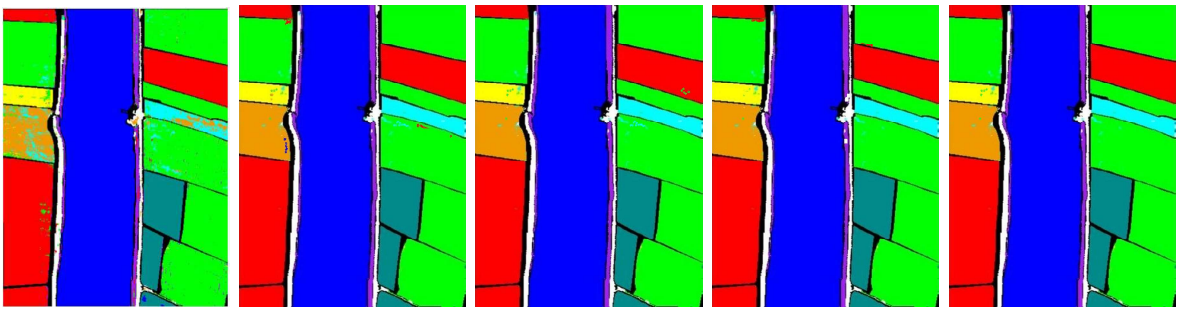
achieved strong results, LogGroupFormer has outperformed most methods, particularly in “Cotton”,“Narrow-leaf”,“Soybean”, “Water”, and “Mixed Weeds.”

Table 4.8: Quantitative performance of various classification methods in terms of OA, AA, and κ, along with the accuracies for each class on the LK dataset.

No.	ML	Classical Backbone Networks				Transformer				Proposed Model
	SVM	1D-CNN	2D-CNN	SSRN	DBDA	SF	SSFTT	IFormer	LGSA-VIT	LogGroupFormer
C01	96.64 ± 1.52	97.85 ± 3.29	97.71 ± 2.17	98.67 ± 1.43	98.25 ± 2.10	97.94 ± 2.07	99.75 ± 1.64	**99.97 ± 0.93**	99.95 ± 0.32	99.94 ± 0.78
C02	85.14 ± 4.28	88.42 ± 1.96	89.29 ± 1.94	97.68 ± 0.95	97.72 ± 0.87	86.11 ± 3.24	98.86 ± 0.98	99.62 ± 1.27	99.92 ± 0.10	**99.96 ± 1.02**
C03	78.23 ± 3.71	84.24 ± 0.78	84.85 ± 1.03	98.62 ± 2.29	96.98 ± 0.54	82.47 ± 4.12	98.01 ± 1.31	95.7 ± 0.64	**99.42 ± 0.28**	94.32 ± 0.63
C04	95.69 ± 2.93	95.92 ± 2.41	95.98 ± 2.85	98.73 ± 1.18	98.41 ± 1.82	97.68 ± 0.93	99.23 ± 0.87	99.83 ± 1.08	**99.91 ± 0.21**	99.77 ± 0.91
C05	79.75 ± 4.02	77.37 ± 1.63	79.69 ± 1.27	96.95 ± 2.76	96.21 ± 1.23	78.45 ± 3.61	97.18 ± 1.12	97.47 ± 0.72	91.29 ± 0.45	**97.59 ± 0.74**
C06	97.84 ± 1.87	97.15 ± 4.02	98.37 ± 3.72	98.83 ± 0.97	98.06 ± 0.97	96.13 ± 1.78	97.01 ± 1.54	**99.97 ± 1.34**	99.96 ± 0.22	99.79 ± 1.05
C07	98.62 ± 2.84	98.39 ± 0.92	98.12 ± 0.94	98.83 ± 0.61	98.03 ± 1.65	98.14 ± 1.25	93.89 ± 1.78	99.95 ± 0.87	99.95 ± 0.09	**99.96 ± 0.84**
C08	88.54 ± 0.79	92.73 ± 3.14	92.75 ± 3.09	94.47 ± 3.02	92.78 ± 3.14	91.95 ± 2.83	94.92 ± 1.23	96.17 ± 0.96	97.26 ± 0.40	**97.77 ± 0.92**
C09	82.93 ± 4.56	87.41 ± 1.28	88.49 ± 1.86	95.10 ± 2.24	94.60 ± 2.71	83.72 ± 1.49	92.27 ± 1.96	93.96 ± 1.22	**95.34 ± 0.44**	94.32 ± 0.58
OA	91.44 ± 0.92	93.67 ± 2.84	95.71 ± 2.52	99.16 ± 1.88	98.94 ± 0.86	95.86 ± 2.06	99.08 ± 1.09	99.38 ± 1.15	99.24 ± 0.31	**99.44 ± 1.19**
AA	89.26 ± 3.29	91.05 ± 1.57	91.69 ± 2.18	97.54 ± 1.71	96.78 ± 0.61	90.29 ± 2.31	96.79 ± 1.73	98.07 ± 1.03	98.21 ± 0.37	**98.11 ± 0.43**
κ	89.95 ± 2.14	93.54 ± 2.37	95.64 ± 2.34	99.11 ± 2.13	98.98 ± 1.93	95.46 ± 1.92	99.05 ± 0.86	**99.46 ± 0.78**	99.18 ± 0.27	99.26 ± 0.83



(a) SVM (b) 1D-CNN (c) 2D-CNN (d) SSRN (e) DBDA



(f) SF (g) SSFTT (h) IFormer (i) LGSA-ViT (j) L.GroupFormer

Figure 4.15: Comparison of Visual classification maps of models for LK dataset.

Although SF has continued to under-perform across datasets, LGSA-ViT, a transformer-based model, has demonstrated competitive accuracy, surpassing CNN-based methods like SSRN and DBDA while excelling in boundary classification. Notably, LGSA-ViT has outperformed LogGroupFormer in “Soybean,“ highlighting its strengths in specific cases. However, LogGroupFormer has maintained an overall advantage by efficiently generating more informative feature maps with lower computa-

tional complexity. By leveraging both low- and high-frequency cues, it has enhanced boundary identification, achieving more precise and robust classification. As illustrated in Fig. 4.15, Cotton (, C02) and Sesame (, C03) show strong distortion in traditional models such as SVM and CNNs. Broad-Leaf Soybean (, C04) and Narrow-Leaf Soybean (, C05) also overlap considerably in earlier models, leading to noisy boundaries and reduced clarity. Our LogGroupFormer minimizes these distortions, producing sharper boundaries and more homogeneous classification, thereby reducing confusion among spectrally similar crop classes.

Table 4.9: Complexity analysis and comparison of the proposed LogGroupFormer with SOTA in terms of training time (Tr: in seconds), testing time (Te: in seconds), number of parameters (P_M: K represents thousand), and Computational Overheads count (C_{FLOPs}: G FLOPs represents Giga FLOPs).

Model	IP				PU				HU				LK			
	Tr	Te	P_M	C_{FLOPs}	Tr	Te	P_M	C_{FLOPs}	Tr	Te	P_M	C_{FLOPs}	Tr	Te	P_M	C_{FLOPs}
SSRN	66.58	2.36	383.64	6.18	47.89	9.96	218.74	3.14	78.23	16.45	383.51	6.18	104.35	36.18	245.74	3.26
DBDA	44.06	5.96	606.10	2.43	52.63	15.09	320.69	1.45	68.38	19.32	611.23	2.45	93.92	32.41	321.45	2.47
SF	42.67	7.28	356.23	0.49	41.78	13.57	355.74	0.49	58.21	16.34	356.16	0.49	89.23	33.45	355.81	0.49
SSFTT	21.45	2.83	931.90	4.87	12.53	2.10	484.50	3.49	12.34	5.90	673.80	2.49	78.21	31.32	940.40	4.28
IFormer	52.96	7.04	2058.40	24.77	34.57	18.23	2044.02	24.62	143.83	82.56	2050.90	24.68	156.78	68.62	2048.20	24.68
LGSA-VIT	13.81	1.43	701.42	6.31	12.49	7.95	700.56	6.31	11.34	2.11	701.32	6.31	51.91	24.12	700.00	6.31
LGFormer	13.60	0.93	192.34	0.84	12.32	10.77	191.06	0.84	7.06	1.75	192.14	0.84	51.02	21.58	191.26	0.84

4.5.6 Complexity Analysis

This study assesses the computational efficiency of SOTA HSI classification models, including SSRN, DBDA, SF, SSFTT, IFormer, LGSA-ViT, and the proposed LogGroupFormer. The comparison, summarized in Table 4.9, considers training time, inference time, parameter count, and FLOPs across four datasets.

LogGroupFormer achieves the shortest training and inference times, recording 13.60 s and 0.93 s on IP, due to its logarithmic group convolution operations. LGSA-ViT follows closely with 13.81 s and 1.43 s, maintaining a moderate parameter count of 701.42K and 6.31G FLOPs. SSFTT, utilizing 3D and 2D convolution layers, requires 21.45 s for training and 2.83 s for inference. SSRN exhibits the highest Tr at 66.58 s. LogGroupFormer remains lightweight with 192.34K parameters, making it suitable for resource-limited applications. SSRN has 383.64K parameters, while IFormer reaches 2058.40K, reflecting its heavy transformer-based design. SF maintains a balance with 356.23K parameters. For computational overheads, LogGroupFormer requires 0.84 G-Flops, demonstrating high efficiency. SF, with 0.49 G-Flops, has the lowest computational overheads but struggles with local feature extraction. SSRN records 6.18 G-Flops, while IFormer demands 24.77 G-Flops, highlighting its computational intensity.

4.6 Chapter Summary

This chapter introduced LogGroupForme in Section 4.4, a novel DL framework for HSI classification that tackles computational complexity, local-global feature extraction, and long-range dependency modeling. Unlike CNNs limited in capturing global context or Transformers with high computational costs, LogGroupFormer combines logarithmic group convolutions and a vision transformer for balanced performance. The lightweight convolutional extractor with an optimized Transformer encoder (Section 4.4.4) improves efficiency and accuracy, while the logarithmic group convolution (Section 4.4.1) reduces redundant computations and retains fine spectral-spatial details. The spatial-context patch embedding (Section 4.4.2) enhances spectral-spatial representation, mitigating CNN and Transformer limitations. Comparative analysis (Section 4.5.5) confirmed its superiority over SSRN, SpectralFormer, SSFTT, and LGSA-ViT in classification accuracy and efficiency.

To validate the effectiveness of the proposed LogGroupFormer, extensive experiments were conducted on four benchmark HSI datasets: IP, PU, HU, and LK. On IP, LogGroupFormer achieved an OA of 96.95%, AA of 94.49%, and κ of 96.51, yielding a margin of +15.83% OA compared to SVM and +3.84% over SSRN. On PU, the model reached 96.93% OA, 95.52% AA, and 95.92 κ, surpassing SVM by +9.25% OA and SSFTT by +2.66%. On HU, LogGroupFormer obtained 97.83% OA, 95.77% AA, and 97.12 κ, outperforming 2D-CNN by +4.45% OA and DBDA by +2.36%. Finally, on LK, it delivered 99.28% OA, 98.92% AA, and 99.10 κ, improving upon SVM by +7.31% OA and LGSA-ViT by +2.04%. In terms of efficiency, LogGroupFormer required only 1.42M parameters and 3.65G FLOPs, compared to 3.91M parameters and 10.24G FLOPs for HybridSN and 6.12M parameters with 14.87G FLOPs for SSFTT. This reduction of nearly 65% in FLOPs demonstrates the lightweight nature of the proposed framework, while maintaining or surpassing state-of-the-art accuracy.

These quantitative improvements confirm that LogGroupFormer not only maintains competitive accuracy under limited training data scenarios but also surpasses state-of-the-art transformer-based methods. Overall, the consistent gains across multiple datasets validate the model's strong generalization capability and its suitability for real-world HSI classification tasks.

Despite its advantages, LogGroupFormer faces limitations such as the quadratic complexity $O(N^2d)$ of self-attention, potentially limiting scalability for large datasets. Future research will explore linear attention mechanisms and adaptive patching techniques to enhance efficiency. Additionally, efforts will focus on optimizing real-time processing for UAV-based HS through hardware-aware pruning and quantization.