




## Article

# SSFAN: A Compact and Efficient Spectral-Spatial Feature Extraction and Attention-Based Neural Network for Hyperspectral Image Classification

Chunyang Wang <sup>1</sup>, Chao Zhan <sup>1</sup>, Bibo Lu <sup>1,\*</sup>, Wei Yang <sup>2</sup>, Yingjie Zhang <sup>3</sup>, Gaige Wang <sup>4</sup> and Zongze Zhao <sup>5</sup>

<sup>1</sup> School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China; wcy@hpu.edu.cn (C.W.); 212309010018@home.hpu.edu.cn (C.Z.)

<sup>2</sup> Center for Environmental Remote Sensing, Chiba University, Chiba 2638522, Japan; yangwei@chiba-u.jp

<sup>3</sup> State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Luoyu Road No.129, Wuhan 430079, China; 2023186190082@whu.edu.cn

<sup>4</sup> School of Computer Science and Technology, Ocean University of China, Qingdao 266100, China; wgg@ouc.edu.cn

<sup>5</sup> School of Surveying and Land Information Engineering, Henan Polytechnic University, Jiaozuo 454000, China; zongze@hpu.edu.cn

\* Correspondence: lubibo@hpu.edu.cn

**Abstract:** Hyperspectral image (HSI) classification is a crucial technique that assigns each pixel in an image to a specific land cover category by leveraging both spectral and spatial information. In recent years, HSI classification methods based on convolutional neural networks (CNNs) and Transformers have significantly improved performance due to their strong feature extraction capabilities. However, these improvements often come with increased model complexity, leading to higher computational costs. To address this, we propose a compact and efficient spectral-spatial feature extraction and attention-based neural network (SSFAN) for HSI classification. The SSFAN model consists of three core modules: the Parallel Spectral-Spatial Feature Extraction Block (PSSB), the Scan Block, and the Squeeze-and-Excitation MLP Block (SEMB). After preprocessing the HSI data, it is fed into the PSSB module, which contains two parallel streams, each comprising a 3D convolutional layer and a 2D convolutional layer. The 3D convolutional layer extracts spectral and spatial features from the input hyperspectral data, while the 2D convolutional layer further enhances the spatial feature representation. Next, the Scan Block module employs a layered scanning strategy to extract spatial information at different scales from the central pixel outward, enabling the model to capture both local and global spatial relationships. The SEMB module combines the Spectral-Spatial Recurrent Block (SSRB) and the MLP Block. The SSRB, with its adaptive weight assignment mechanism in the SToken Module, flexibly handles time steps and feature dimensions, performing deep spectral and spatial feature extraction through multiple state updates. Finally, the MLP Block processes the input features through a series of linear transformations, GELU activation functions, and Dropout layers, capturing complex patterns and relationships within the data, and concludes with an argmax layer for classification. Experimental results show that the proposed SSFAN model delivers superior classification performance, outperforming the second-best method by 1.72%, 5.19%, and 1.94% in OA, AA, and Kappa coefficient, respectively, on the Indian Pines dataset. Additionally, it requires less training and testing time compared to other state-of-the-art deep learning methods.



**Citation:** Wang, C.; Zhan, C.; Lu, B.; Yang, W.; Zhang, Y.; Wang, G.; Zhao, Z. SSFAN: A Compact and Efficient Spectral-Spatial Feature Extraction and Attention-Based Neural Network for Hyperspectral Image Classification. *Remote Sens.* **2024**, *16*, 4202. <https://doi.org/10.3390/rs16224202>

Academic Editors: Salah Bourennane, Gangyao Kuang, Siqian Zhang, Xin Su and Olga Sykioti

Received: 2 September 2024

Revised: 1 November 2024

Accepted: 7 November 2024

Published: 11 November 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** deep learning; hyperspectral image classification; attention mechanisms; convolutional neural networks; spectral-spatial learning; loss function

## 1. Introduction

Hyperspectral remote sensing technology captures hundreds of spectral bands from a target area using sensors or imaging spectrometers, thereby acquiring both spatial and

spectral information simultaneously. In recent years, advancements in hyperspectral sensors and spectral imaging technology have significantly enriched the information contained in hyperspectral images (HSIs) [1]. These HSIs not only provide detailed two-dimensional spatial information of the target but also include one-dimensional spectral information, making them highly applicable in various fields such as biomedical imaging [2], mineral exploration [3], food safety [4], disaster prevention and mitigation [5], urban development [6], military reconnaissance [7], and precision agriculture [8]. To fully leverage the potential of HSI data, researchers have explored various data processing techniques, such as denoising [9,10], spectral unmixing [11], and target detection and classification [12–14]. Among these techniques, land cover classification has garnered significant attention. The primary objective of HSI classification is to use the rich spatial and spectral information in HSI to classify each pixel according to land cover types. Despite the valuable opportunities provided by such rich data, effectively extracting and distinguishing relevant features remains a significant challenge. Consequently, researchers continue to explore various approaches to address the challenges in feature extraction for HSI classification.

Traditional machine learning techniques were mostly used in the early stages of HSI classification attempts, which were centered on the extraction of spectrum information. Various techniques are frequently employed in HSI classification, such as random forest [15], k-nearest neighbor [16], support vector machines [17,18], and Bayesian estimation methods [19]. HSIs, on the other hand, contain a great deal of redundant information in their vast amounts of spectral information—typically hundreds of bands. In light of this, the researchers developed reduced dimensionality and feature extraction methods, such as principal component analysis (PCA) [20,21], independent component analysis (ICA) [22], and linear discriminant analysis (LDA) [23,24]. These methods map the original spectral features into the new space using linear or nonlinear transforms in order to achieve reduced dimensionality and feature extraction, which significantly lowers the number of spectral features. The process of feature extraction significantly lowers the model's running complexity and enhances the effectiveness of conventional machine learning models in HSI classification tasks. Model classification performance is limited by these approaches' limited ability to interpret spatial information, notwithstanding their effectiveness in extracting spectral features. Thus, there has been a lot of interest in spectrum spatial feature extraction techniques, and to improve the extraction of spatial features from HSIs, researchers have created mathematical morphological operators. The techniques of morphological profile (MP) [25], extended morphological profile (EMP) [26], and extended multiattribute profile (EMAP) [27] leverage the integration of spatial and spectral information through various methodologies. These approaches facilitate the identification of the size and shape of distinct objects within an image, consequently enhancing the accuracy of classification outcomes. However, these methods, as the starting stage of HSI classification, have shown effectiveness in understanding the data and its features, but they show limitations when facing the complexity of real HSI data limitations, especially in how to fuse spatial and spectral information more effectively.

Deep learning techniques simulate the hierarchical functioning of the human visual system by constructing deep network models with hierarchical structures based on the characteristics of input data and artificial neural networks. These models can independently learn high-level, discriminative features from the data. With the advancement of deep learning, leveraging powerful computational resources and abundant data, recent algorithms such as CNNs [28,29], Transformer [30,31], and Mamba [32,33] have been employed in hyperspectral image (HSI) classification, demonstrating excellent performance in this task. CNNs are particularly effective at extracting spatial features and learning feature representations automatically, thereby improving image classification accuracy and offering robust feature extraction capabilities. Various CNN architectures have been proposed for extracting both spectral and spatial features, including 1D CNNs [34], 2D CNNs [35], 1D-2D CNNs [36], 3D CNNs [37], and 2D-3D CNNs (Hybrid CNNs) [38]. 1D

CNNs [39,40] are primarily used for spectral feature extraction, while 2D CNNs [41,42] delve into deep spatial features of pixels within spectrally compressed image blocks. 3D CNNs [43,44] are employed to extract both spectral and spatial features from HSI data, and Hybrid CNNs [45,46] leverage the advantages of 2D and 3D CNNs for a more comprehensive extraction of multi-scale and multi-dimensional information in HSIs. In spatial convolutional neural networks, the DHCNet [47] model introduces variability convolution and adaptive pooling operations that can dynamically adjust their size based on input spatial information, addressing the limitation of fixed-position convolution kernels in traditional CNNs, which cannot adapt to spatial structures. Zhong et al. [48] proposed a spatial-spectral residual network, SSRN, for HSI classification, leveraging the information of front-layer features as complements to back-layer features, significantly enhancing feature utilization. In the spectral-spatial convolutional neural network, Roy et al. [38] introduced HybridSN, capable of more efficient learning of spectral-spatial features and more abstract spatial features, contributing to improved classification accuracy. Li et al. [49] proposed a dual-channel 2D CNN architecture that considers both local and global spatial features while capturing spectral features, adaptively combining feature weights from two parallel streams to enhance the network's expressive capabilities. Additionally, FADCNN [50] presents a spatial-spectral dense convolutional neural network framework that employs a feedback attention mechanism, facilitating improved extraction and integration of spectral and spatial features, as well as refining these features to leverage semantic information. Despite the good classification results achieved by CNNs as HSI feature extractors, they face limitations in processing complex high-dimensional data, insufficient integration of spatial and spectral information, and a high demand for training samples.

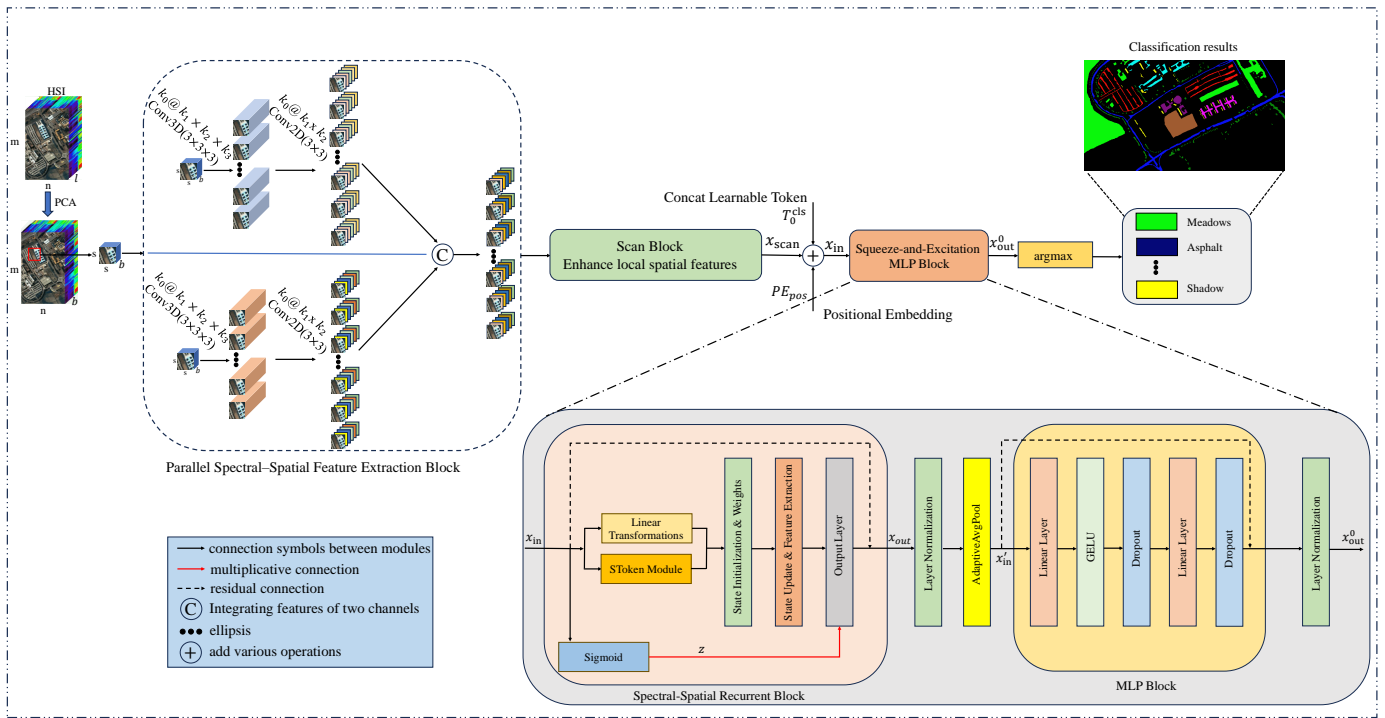
Following the success of CNNs, graph convolutional networks (GCNs) have increasingly been applied in HSI classification due to their advantages in processing graph-structured data [51]. By constructing relational graphs among pixels, GCNs effectively model the complex interactions between spatial and spectral information, thereby enhancing classification performance. Qin et al. [52] proposed a second-order GCN, extending the standard GCN structure to fully utilize the inter-band relationships in hyperspectral images, improving classification accuracy. Wan et al. [53] applied superpixel segmentation, dividing hyperspectral images into multiple superpixels and feeding these as nodes into a GCN, effectively extracting both internal and neighboring information of superpixels and enhancing classification results. Additionally, the dynamic multiscale graph convolutional network classifier (DMSGer) [54] was proposed to capture pixel-level and region-level features simultaneously, strengthening classification performance in hyperspectral images. By modeling at multiple scales, DMSGer can better capture complex spatial features, thereby improving the ability to differentiate between classes. However, GCNs still face limitations in graph construction, particularly for large-scale graphs where computational costs become prohibitive, making it challenging for GCNs to classify or identify materials in large-scale hyperspectral scenes efficiently.

The Transformer architecture, initially introduced for natural language processing [55], has been creatively adapted for computer vision, leading to the development of Vision Transformer [56]. This innovation has expanded the application of Transformers into the field of HSI analysis. Unlike the CNN approach, which focuses on local spatial information, the Transformer's self-attention mechanism allows for the effective control of global sequence information by matching the positional encoding of data. This mechanism efficiently captures remote dependencies, providing a comprehensive understanding of the complex relationships between spatial and spectral features in HSI. HSI-BERT [57] represents a pioneering application of Transformer-based models in HSI classification. It treats each pixel in the HSI cube as a Transformer token to capture the global context, demonstrating competitive accuracy. Hong et al. [58] recognized the critical role of long-range dependencies in spectral bands and proposed SpectralFormer, a model that utilizes a pure Transformer architecture specialized in processing spectral features and establishing long-range dependencies. Tang et al. [59] proposed a Transformer network with a

dual-attention mechanism, capturing spectral and spatial features separately, and achieved superior classification results through the introduction of a jump-connection mechanism.

As research progressed, it was found that fusing CNN and Transformer for feature extraction could achieve better classification results. For instance, the SSFTT [60] method preprocessed HSI data using 3D and 2D convolutions. 3D convolution was used to capture both spectral and spatial information features, while 2D convolution focused on extracting purely spatial features. A Gaussian-weighted feature tagger was then used to generate input tokens, which were fed into the Transformer encoder for classification by a linear layer. SSFTT successfully addressed the deep semantic feature extraction problem in HSI classification and became an important benchmark for subsequent landmark Transformer-based HSI classification research. Roy et al. [61] proposed a novel morphFormer network for HSI classification, enhancing feature interaction through the combination of an attention mechanism with learnable spectral and spatial morphology convolutions, leading to significantly improved classification performance. Despite its impressive performance, the Transformer architecture has several drawbacks in real-world applications. Its multi-layer structure and complexity require significant processing power during training and inference [62]. Moreover, effective training of the Transformer model often necessitates a substantial amount of labeled data, which can be costly and difficult to obtain for hyperspectral data, especially when samples are limited or imbalanced. This can lead to overfitting and challenges in applying the model to new data. Additionally, the intricate self-attention mechanism and high computational complexity of the Transformer model result in poor real-time performance [63]. Designing a hyperspectral network with few parameters, good classification performance, and high real-time performance presents a significant challenge.

This paper introduces the novel HSI classification model SSFAN, as illustrated in Figure 1. SSFAN integrates advanced spectral-spatial feature extraction and deep learning algorithms. The model is composed of three key components: the Parallel Spectral-Spatial Feature Extraction Block (PSSB), the Scan Block, and the Squeeze-and-Excitation MLP Block (SEMB), designed to effectively extract and process spectral and spatial information, thereby enhancing the classification accuracy of HSI. The HSI data are initially preprocessed and fed into the PSSB, which includes two parallel streams. Each stream incorporates a 3D convolutional layer followed by a 2D convolutional layer. This process utilizes 3D convolution to extract spectral and spatial information from the input hyperspectral data, and then enhances the spatial feature representation through 2D convolution. The Scan Block is responsible for extracting spatial information at different scales from the center pixel, outward, employing a layered scanning strategy. This enables the model to capture both local and global spatial relationships. The SEMB consists of the Spectral-Spatial Recurrent Block (SSRB) and an MLP Block, which employs a deep residual structure combined with LayerNorm. This structure enhances the nonlinear representation of features while maintaining model stability. The SSRB introduces the SToken Module, a mechanism for adaptive weight assignment that facilitates flexible handling of time steps and feature dimensions through multilayered linear transformations and parameterization operations. Multiple state update operations are utilized to extract deeper spectral-spatial features. Finally, the MLP Module processes the input features through a series of linear transformations, activation functions (GELU), and Dropout layers, enabling the capture of complex patterns and relationships in the input data. The classification is completed through an argmax layer. The SSFAN model stands out by significantly reducing the number of parameters and MACs compared to other state-of-the-art models, thereby accelerating the training and inference speeds and enhancing the model's deployment capabilities under limited computational resources. The codes are available at <https://github.com/one-boy-zc/SSFAN> (accessed on 25 October 2024).



**Figure 1.** Overall architecture of the proposed SSFAN network for HSI classification. The framework is composed of three main sections: the Spectral-Spatial Feature Extraction Block, the Scan Block, and the Squeeze-and-Excitation MLP Block, each with detailed internal composition.

The contributions of this work are summarized as follows:

- (1) A Parallel Spectral-Spatial Feature Extraction Block was proposed, which can increase classification accuracy and extract spectral-spatial information more fully.
- (2) The Scan Block, designed for image spreading, allows the model to capture both local and global spatial relationships through a layered scanning method.
- (3) The combination of SSRB and MLP Block in the SEMB introduces an adaptive weight assignment mechanism, facilitating the extraction of deeper spectral-spatial features through multi-layer linear transformations and parameterization operations.
- (4) SSFAN significantly reduces the number of parameters and MACs compared to Transformer-based models, speeding up training and inference and improving deployment capabilities.

## 2. Materials and Methods

The SSFAN model for HSI classification is composed of three primary components: the Scan Block, the Parallel Spectral-Spatial Feature Extraction Block, and the Squeeze-and-Excitation MLP Block, as depicted in Figure 1.

### 2.1. HSI Data Preprocessing

Given the raw Hyperspectral Imaging (HSI) data  $I \in \mathbb{R}^{m \times n \times l}$ , where  $l$  represents the number of spectral bands, and  $m \times n$  denotes the spatial resolution size, each pixel in  $I$  is characterized by  $l$  spectral dimensions and is associated with a one-hot vector  $Y = (y_1, y_2, \dots, y_C)$ , where  $C$  is the number of feature classes. Rich spectrum information is present in  $l$  spectral bands, but they also result in a great deal of information redundancy, which greatly raises the computing cost. Therefore, the computational and spectral dimensions are reduced by using Principal Component Analysis (PCA) [21]. PCA maintains the spatial dimensions of the HSI while reducing its spectral dimensions from  $l$  to  $b$ . The particular procedure is:

$$I_b = (I_l - \mu) \times V_b \quad (1)$$

where  $\mu \in \mathbb{R}^l$  is the mean vector of each spectral channel,  $V_b \in \mathbb{R}^{l \times b}$  is the eigenvector matrix corresponding to the first  $b$  largest eigenvalues of the covariance matrix, and  $I_l \in \mathbb{R}^{m \times n \times l}$  and  $I_b \in \mathbb{R}^{m \times n \times b}$  denote the original hyperspectral data with  $l$  bands and the dimensionality-reduced hyperspectral data with  $b$  bands, respectively.

The HSI data were then subjected to 3D-patch extraction following spectral down-scaling via PCA.  $I_b$  was used to generate each neighboring 3D-patch ( $P \in \mathbb{R}^{s \times s \times b}$ ), where  $s \times s$  represents the window size. Every 3D-patch has a center pixel set to  $(x_i, x_j)$ , where  $0 \leq i < m$  and  $0 \leq j < n$ . The label of each 3D-patch's center pixel determines the real label of each patch. However, some pixel values in the patch are unavailable when extracting the region surrounding a pixel that is situated at the edge of the image because there isn't any pixel data beyond the boundary. Consequently, a padding operation with a padding width of  $(s - 1)/2$  is carried out on these pixels, based on the method in SSFTT [60]. At some point,  $m \times n$  determines how many 3D-patches there are in  $I_b$ . The width and height of each patch are  $[x_i - (s - 1)/2, x_i + (s - 1)/2]$ ,  $[x_j - (s - 1)/2, x_j + (s - 1)/2]$ , and  $b$  is their spectral dimension. Every sample is split into train and test datasets once background pixels with zero labels are eliminated.

## 2.2. Parallel Spectral–Spatial Feature Extraction Block

After data preprocessing, the spectral-spatial information in each sample patch is extracted using the Parallel Spectral-Spatial Feature Extraction Block (PSSB). In contrast to the conventional single pathway, which is unable to sufficiently extract the spectral-spatial information [60], the PSSB is comprised of two parallel streams, each having a 2D and 3D convolutional layer. The inputs of both pathways are identical sample patches, which combine the features that were extracted from the two streams. The PSSB description follows one pathway since the two streams have the exact identical configuration. The input of each sample patch ( $\mathbb{R}^{s \times s \times b}$ ) is fed into the 3D convolutional layer.

The process of 3D convolution is detailed in Figure 2. Given  $v_{i,j}^{\alpha,\beta,\gamma}$  is the value at the  $(\alpha, \beta, \gamma)$  position of the  $i$ -th convolution kernel's  $j$ -th output, and  $w_{i,j}^{\alpha+h,\beta+w,\gamma+r}$  is the value at the  $i$ -th convolution kernel's  $j$ -th output used for the  $j$ th output. kernel  $(\alpha + h, \beta + w, \gamma + r)$  at the weight value, and  $p^{\alpha+h,\beta+w,\gamma+r}$  is the value at  $(\alpha + h, \beta + w, \gamma + r)$  in the sample patch, which is then given by the computational formula for 3D convolution:

$$v_{i,j}^{\alpha,\beta,\gamma} = \phi \left( \sum_{h=0}^{H_i-1} \sum_{w=0}^{W_i-1} \sum_{r=0}^{R_i-1} w_{i,j}^{\alpha+h,\beta+w,\gamma+r} p^{\alpha+h,\beta+w,\gamma+r} + b_{i,j} \right) \quad (2)$$

where  $H_i$  and  $W_i$  represent the height and width of the sample patch, respectively,  $R_i$  represents the number of bands in the sample patch,  $\phi(\cdot)$  denotes the activation function (ReLU), and  $b_{i,j}$  denotes the  $i$ -th bias used for the  $j$ -th output. Theoretically,  $k_0$  3D convolutional kernels with a size of  $k_1 \times k_2 \times k_3$  make up the 3D convolutional layer.  $k_0$  3D feature cubes with spectral-spatial information will be generated after the 3D convolutional layer. After the 3D convolutional layer,  $k_0$  3D feature cubes containing spectral-spatial information will be generated. The size of each cube is shown in Equation (3), and the total size of all feature cubes is shown in Equation (4).

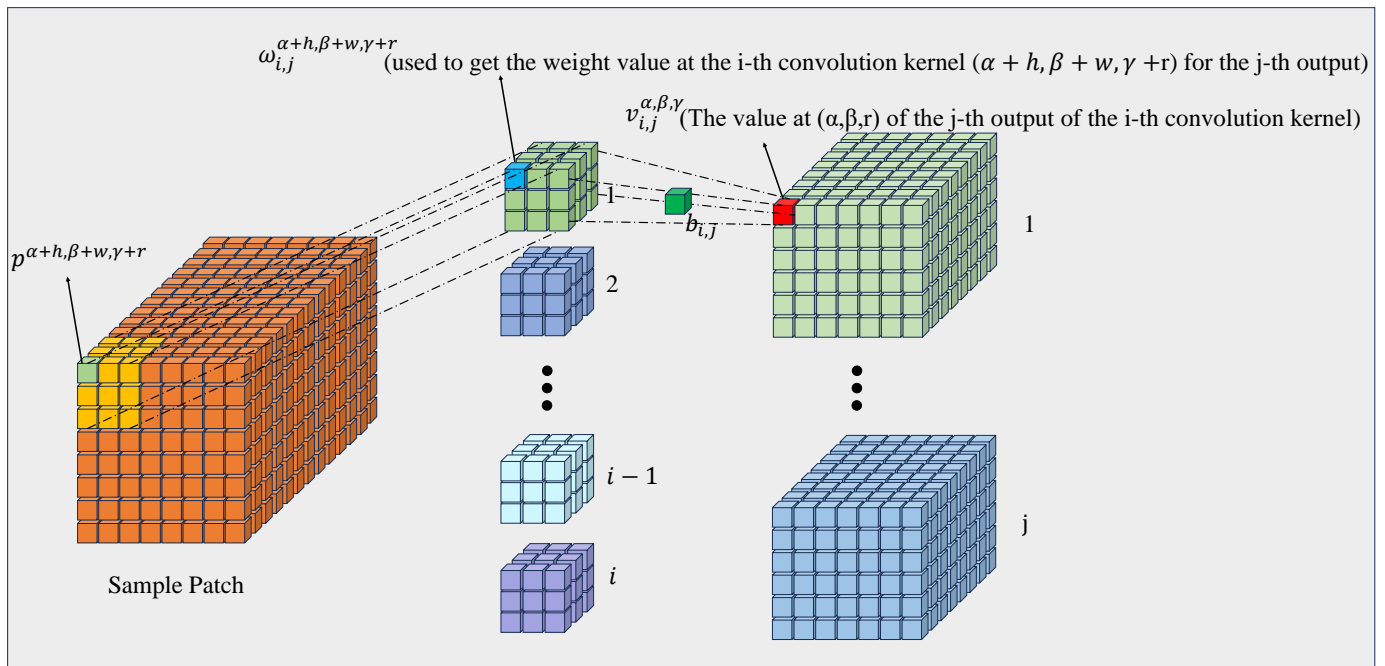
$$\text{Each-cube-size} = (s - k_1 + 1) \times (s - k_2 + 1) \times (b - k_3 + 1) \quad (3)$$

$$\text{Total-cubes-size} = k_0 @ (s - k_1 + 1) \times (s - k_2 + 1) \times (b - k_3 + 1) \quad (4)$$

After the rearrangement operation and fed as an input to the next 2D convolutional layer, which has a size of  $(s - k_1 + 1) \times (s - k_2 + 1) \times k_0(b - k_3 + 1)$ , in order to enhance the spectral spatial features. At the spatial location  $(\alpha, \beta)$  on the  $j$ th feature map in the  $i$ -th layer of the 2D convolutional layer, the activation value  $v_{i,j}^{\alpha,\beta}$  is defined as follows:

$$v_{i,j}^{\alpha,\beta} = \phi\left(\sum_k \sum_{h=0}^{H'_i-1} \sum_{w=0}^{W'_i-1} w_{i,j,k}^{h,w} m_k^{\alpha+h,\beta+w} + b_{i,j}\right) \quad (5)$$

where  $H'_i$  and  $W'_i$  represent the width and height of the 2D convolutional kernel, and  $w_{i,j,k}^{h,w}$  represents the weight parameter at the  $k$ -th feature map  $(h, w)$ .  $\phi(\cdot)$  denotes the activation function (ReLU).  $m_k^{\alpha+h,\beta+w}$  denotes the value of the  $k$ th feature map at  $\alpha + h, \beta + w$ . The total number of feature maps after 2D convolutional layer processing is  $k'_0 \times [s - 2 \times (k_1 + 1)] \times [s - 2 \times (k_2 + 1)]$ , with  $k'_0$  being the number of 2D convolutional kernels, and each convolutional kernel having a size of  $k_1 \times k_2$ , which are all set to 3.



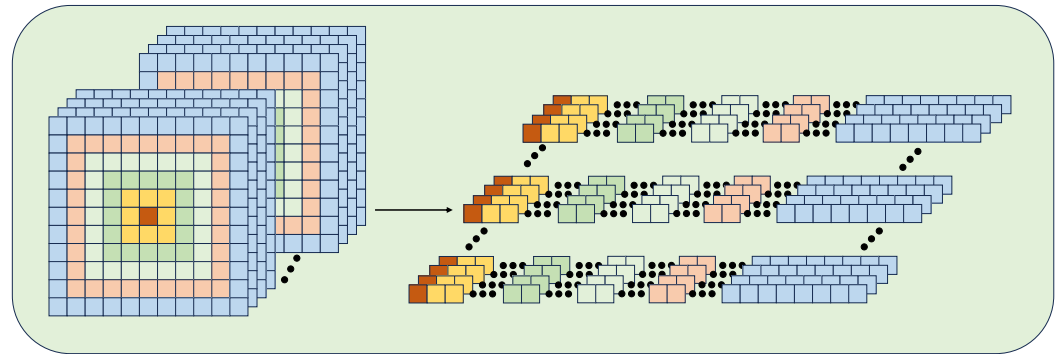
**Figure 2.** Visualizing 3D Convolution Process.

Moreover, PSSB effectively addresses the high-dimensional nature of HSI data, mitigating the computational and representational shortcomings of single-channel methods in dealing with high-dimensional data. By integrating features from two identical channels, PSSB circumvents the limitation that single-channel methods cannot fully extract spectral and spatial features. This feature fusion enhances the model's capability to capture complex HSI data and facilitates more effective extraction of spectral-spatial information, thereby improving the classifier's accuracy, particularly in the classification of edge pixels or mixed pixels.

### 2.3. Scan Block

The Scan Block is an enhanced version of the spreading operation, designed to extract multi-scale features from the central region and its surroundings, enabling the model to capture local information at different scales. The input to the Scan Block is the multi-channel feature map output from the PSSB. Initially, the Scan Block calculates the midpoints of the input tensor's height and width ( $cen$ ) to determine the center index. The input tensor's dimensions are then reshaped from  $(B, c, w, h)$  to  $(B, h, w, c)$  to facilitate manipulation of spatial and spectral information. This reshaping makes it easier to handle the spatial dimensions  $h$  and  $w$  (the second and third dimensions of  $x$ ), aligning them for efficient slicing operations. Next, an output tensor,  $x_{scan}$ , is initialized to store the extracted values, with dimensions  $(B, h \times w, c)$ . This output tensor represents a multi-channel spread of the feature map. The center region,  $cen$ , is first assigned directly to  $x_{scan}[:, 0, :]$ . Subsequently,

features from regions of varying scales are extracted iteratively, starting from the center and gradually expanding outward. In each  $i$ -th layer of this loop, a region containing  $2 \times i$  rows and  $2 \times i$  columns is extracted, resulting in a total of  $4 \times i$  pixels being added, with five loops in total. As shown in Figure 3, this process transforms the feature map from 2D to 1D without altering the number of channels. The color distribution in the figure illustrates how values in the original 2D feature map are accurately mapped to the corresponding positions in  $x_{\text{scan}}$  after the spreading process. By progressively extracting regions at different scales, the Scan Block effectively captures local features of varying sizes, enhancing spatial information processing. This multi-scale feature extraction improves the model's spatial perception, ultimately enhancing overall performance.



**Figure 3.** Visualizing the Scan Block Process.

Before inputting the output of the Scan Block to the next module, it is necessary to add position information and a Learnable Token to the output. The Learnable Token is added to facilitate the subsequent classification work. With the Learnable Token, the model can gather the global information of the whole sequence, which can provide a useful global feature for the classification task. Position Embedding is used to label the positional information of each semantic tokens, allowing the model to process sequences with spatial or sequential sensitivity, which is crucial for the model's comprehension. The output after adding the position information and a Learnable Token is  $x_{in}$  with a size of  $64 \times 122 \times 16$ .

#### 2.4. Squeeze-and-Excitation MLP Block

The Squeeze-and-Excitation MLP Block (SEMB) consists of two core modules: the Spectral-Spatial Recurrent Block (SSRB) and the MLP Block. The combination of these two modules enables a more comprehensive extraction and aggregation of the information in the hyperspectral data, which makes the model able to complex spectral dimensions with better expressiveness and classification accuracy. For the input feature  $x_{in}$ , the shape is  $[B, L, D]$ ,  $B$  is the batch size,  $L$  is the length of the sequence, and  $D$  is the feature dimension.  $x_{in}$  is input to the SSRB module for processing by the self-attention mechanism to extract important spectral and spatial features. The output of the SSRB passes through the AdaptiveAvgPool and Layer Normalization layers before being passed to the MLP Block for further deep extraction of feature information from the sequence. Each module contains a residual linkage mechanism, which not only helps the effective transfer of information, but also mitigates the gradient vanishing problem and ensures the robust training of the model. Next, the working principle and implementation of these two key modules will be introduced in detail.

##### 2.4.1. Spectral-Spatial Recurrent Block

As shown in Figure 1, Spectral-Spatial Recurrent Block (SSRB) combines various approaches such as Linear Layer, Attention Mechanism, and State Update etc. SSRB consists of six main parts: Sigmoid, Linear Transformations, SToken Module, State Initialization & Weights, State Update & Feature Extraction, and Output Layer. Firstly,  $x_{in}$  is input into the Sigmoid function and compressed into the range of  $[0, 1]$  to get the activation value  $z$ ,



which is mainly used for the subsequent output adjustment. Next is the State Initialization & Weights module, which is used to initialize some tensors for subsequent processing. Two weight matrices  $\delta_0$  and  $A_0$  are randomly generated in the initial stage to participate in the subsequent recursive operations. Then,  $x_{in}$  is input into two linear transformations  $LN_B$  and  $LN_C$  to generate two intermediate tensors  $B_0$  and  $C_0$  with the same dimensions as  $x_{in}$ , respectively. Meanwhile,  $x_{in}$  is input to SToken Module to get the feature vector  $T$ . The computation of  $T$  will be described in detail in the subsequent section, and the formulae for  $B$  and  $C$  are as follows:

$$B = W_B \cdot x + b_B \quad (6)$$

$$C = W_C \cdot x + b_C \quad (7)$$

where the bias terms are  $b_B$  and  $b_C$ , and the weight matrices of the linear layer are  $W_B$  and  $W_C$ . Then,  $\delta$  and  $A$  are calculated. To obtain  $\delta$ , the input  $x_{in}$  is linearly varied, and the result is calculated using a Sigmoid nonlinear activation function, which can be computed as follows:

$$\delta = \text{Sigmoid}(LN_\delta(x) + \delta_0) \quad (8)$$

where  $LN_\delta(\cdot)$  denotes the linear transformation.  $A$  is obtained from  $\delta_0$  and  $A_0$  for subsequent state updates, calculated as follows:

$$A = \delta_0 \cdot A_0. \quad (9)$$

$B$  is obtained by the  $B_0$  and  $\delta$  Einstein summation conventions, calculated as:

$$B = \delta \cdot B_0. \quad (10)$$

The state update and feature extraction are carried out in the State Update & Feature Extraction module following the completion of the initialization tensor. The state update formula is as follows, assuming that the initialization state  $s$  is a zero vector, processing the state recursively at each time step, and conducting recursive updates for each time step  $t$ ,  $t$  in the range  $[0, L - 1]$ , where  $L$  is the length of  $x_{in}$ .

$$s(t + 1) = A_t \cdot s(t) + B_t \cdot x_t \quad (11)$$

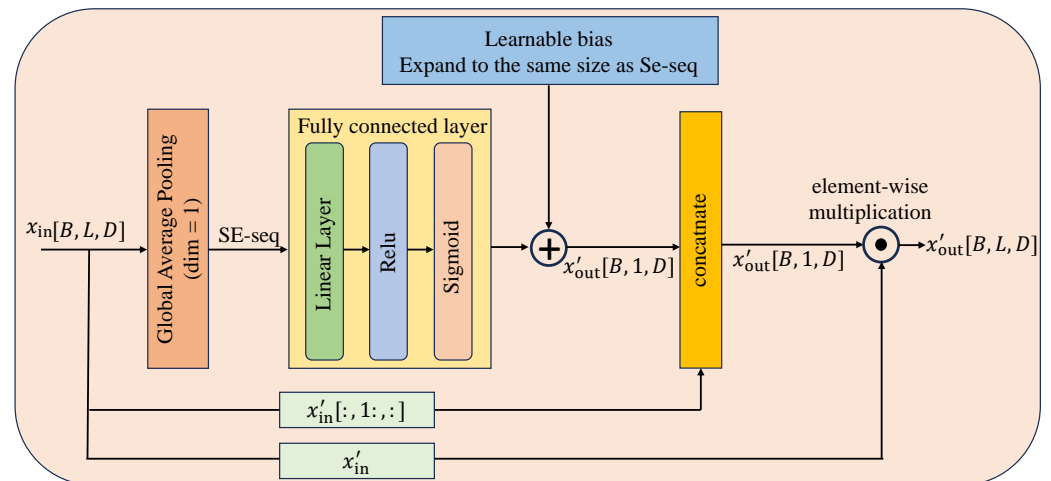
where  $A_t$  is  $A[:, t]$ ,  $B_t$  is  $B[:, t]$ ,  $x_t$  is  $x_{in}[:, t, :]$ , and the output prediction is obtained from  $s(t + 1)$ , computed as:

$$y_{pred,t} = C_t \cdot s(t + 1) + T_t \quad (12)$$

where  $C_t$  is  $C[:, t]$ ,  $T_t$  is  $T[:, t, :]$ , and  $x_t$  is  $x_{in}[:, t, :]$ . After the recursive processing is completed, the predictions of all time steps are stitched together to obtain the final output  $x_{out}$  and adjusted with  $z$ , computed as:

$$x_{out} = [y_{pred,0}, y_{pred,1}, \dots, y_{pred,L-1}]. \quad (13)$$

The SToken Module plays a crucial role in integrating attention mechanisms within the SSRB framework, particularly in enhancing input sequence features through an attention mechanism. This module effectively captures significant spectral information features and fuses them with the original input by applying a Squeeze-and-Excitation (SE) attention computation on the input data sequence and combining it with a trainable bias parameter. The SToken Module is designed to implement an SE attention mechanism, which operates on the input sequence's dimensionality to perform attention computation, and then adjusts the input features' weights, enabling the model to focus more on important features. Figure 4 visualizes the SToken Module process.



**Figure 4.** Visualizing the SToken Module Process.

The module begins by extracting a compressed feature from the input sequence using a global average pooling operation. This operation averages the sequence in the sequence dimension, resulting in a feature vector that represents the average feature of the  $i$ th sample over the sequence dimension, denoted as SE-seq <sub>$i$</sub> . Subsequently, this feature vector is fed into a fully connected layer, which consists of a linear layer, a Sigmoid function, and a ReLU activation function. The Sigmoid function is used to scale the features within the  $[0, 1]$  interval, while the ReLU activation provides a nonlinear mapping. A learnable bias is then initialized and extended to the same shape as the SE-seq, followed by a fully connected layer. The computational formula for this process is:

$$x'_{out} = Sigmoid(ReLU(W_1 \cdot SE-seq + b_1)) + bias \quad (14)$$

where  $b_1$  represents the bias added in the fully connected layer, and  $W_1$  is the weight of the fully connected layer. The resulting  $x'_{out}$  has a shape of  $[B, 1, D]$ , which is then concatenated with the second to the last element of the original input  $x'_{in}$ , effectively incorporating the adjusted features into the sequence while preserving the original sequence features. The shape of the adjusted  $x'_{out}$  is restored to  $[B, L, D]$ . Finally,  $x'_{out}$  is element-wise multiplied with  $x'_{in}$ , a process analogous to the attention mechanism, which assigns greater weight to important parts of the sequence, thereby highlighting key features, computed as:

$$x'_{out} = x'_{out} \odot x'_{in}. \quad (15)$$

This operation adjusts and weights the features of the input sequence, improving the model's focus on important features and facilitating subsequent state updates.

#### 2.4.2. MLP Block

Before formally entering the MLP Block, the output  $x_{out}$  in the Spectral-Spatial Recurrent Block is normalized and adaptive mean pooling is performed in order to homogenize the features over the sequence dimension. This adaptive average pooling layer serves as a dimensionality reduction operation on the input tensor  $x_{out}$ , transforming it into a compact feature representation for subsequent processing in the MLP Block. The shape of  $x_{out}$  is  $[B, L, D]$ , and the transpose operation is first applied to swap dimensions 1 and 2, changing the shape to  $[B, D, L]$ . This step is taken to enable the adaptive average pooling operation to perform pooling operations on the sequence dimension. Subsequently, the tensor enters the adaptive average pooling operation, which performs global average pooling over the entire sequence  $L$  for each feature dimension  $D$ . The average of all elements in each sequence is used as the output of that sequence, resulting in a shape of  $[B, D, 1]$ . Finally, a squeeze operation is applied, removing the dimension of size 1, altering the tensor's shape to

$[B, D]$ . The output after adaptive average pooling, denoted as  $x'_{in}$ , serves as the input to the MLP Block.

The MLP Block comprises two linear layers, followed by a GELU activation function and two Dropout layers, each applied after the linear layers. After the first linear layer, a GELU activation function is applied, followed by dropout. The input then proceeds through the second linear layer and dropout before reaching the final output. This MLP layer is followed by a LayerNormalize layer, which aids in mitigating gradient explosion and vanishing gradients, facilitating faster training. The output  $x^0_{out}$  after the MLP Block has a size of  $64 \times 16$ , and the final result is obtained through the argmax function in the numpy library.

### 2.5. Implementation

Compared to the Transformer model, SSFAN has fewer parameters and multiply-add cumulative number of operations, and is able to accomplish the classification task more quickly and to a high standard. In this paper, we take the Pavia University dataset with 9 classes of features and size of  $610 \times 340 \times 103$  as an example to illustrate the proposed SSFAN.

After PCA dimensionality reduction and patch partitioning, each patch has a size of  $15 \times 15 \times 30$ . The PSSB consists of two identical parallel paths; we will analyze the data flow using the first path as an example. In the 3D convolution layer of the first path, each patch generates eight feature cubes of size  $13 \times 13 \times 28$ , with each patch utilizing eight  $3 \times 3 \times 3$  convolution kernels. The purpose of the 3D convolution layer is to extract rich spectral information from each patch. After rearranging the eight feature cubes, a feature cube consisting of features of size  $13 \times 13 \times 224$  is generated. Next, 64 two-dimensional convolution kernels are used to perform 2D convolution, resulting in 64 feature maps of size  $11 \times 11$ . Each feature map is processed through a Scan operation, flattening it into 64 feature vectors of size  $121 \times 16$ , where 16 represents the number of channels. Simultaneously, a learnable token vector, cls-tokens, initialized to all zeros, of size  $64 \times 1 \times 16$  is created and concatenated with the output of the Scan Block. Subsequently, positional information is added, yielding  $x_{in} \in \mathbb{R}^{64 \times 122 \times 16}$ , which is then input into the Squeeze-and-Excitation MLP Block, passing sequentially through the SSRB and MLP Block modules. In the SSRB, the input undergoes SToken attention mechanism and sequence modeling, and the resulting output is compressed into a single global representation through adaptive pooling. After passing through the MLP Block for nonlinear transformation, the final classification result is obtained. The overall process of the proposed SSFAN method is illustrated in Algorithm 1.

### 2.6. Loss Function

The cross-entropy loss function [64] is widely utilized in HSI classification. Its primary benefit lies in its ability to accurately assess the discrepancy between the model's predicted probability distribution and the actual labeling distribution, which aids in achieving rapid convergence during model training. Nonetheless, the cross-entropy loss function's sensitivity to class imbalance and outliers can impact model performance in certain scenarios, necessitating supplementary strategies to address these issues in real-world applications. In order to achieve this, this paper designs a hybrid cross-entropy loss function ( $\mathcal{L}_{mix}$ ), which is more sensitive to the category distribution and robust. It combines the Normalized Generalized Cross Entropy ( $\mathcal{L}_{NGCE}$ ) and the Normalized Cross Entropy ( $\mathcal{L}_{NCE}$ ) loss functions.

**Algorithm 1** SSFAN Model

---

**Input:** Input HSI data  $I \in \mathbb{R}^{m \times n \times l}$  and ground-truth  $Y \in \mathbb{R}^{m \times n}$ ; after PCA bands number  $b = 30$ ; test dataset comprises  $\mu = 90\%$  of the total; patch size  $s = 15$ ;

**Output:** Output the predicted categories for the test dataset.

- 1: Set Batchsize to 100; Optimizer Adam, learning rate  $lr = 0.001$ ; Training epochs  $epoch = 100$ ;
- 2: After PCA transformation,  $I \in \mathbb{R}^{m \times n \times b}$  is obtained.
- 3: Generate sample patches  $patch$  from  $I \in \mathbb{R}^{m \times n \times b}$  and divide them into training and testing datasets.
- 4: Generate train dataloader and test dataloader.
- 5: **for**  $i = 1$  to  $\in$  **do**
- 6:   Execute 3D convolutional layer and 2D convolutional layer to obtain  $x_1$ .
- 7:   Execute another 3D convolutional layer and 2D convolutional layer to obtain  $x_2$ ;
- 8:    $x = x_1 + x_2$
- 9:   Perform Scan Block.
- 10:   Initialize learnable tokens, connect them to the output of the Scan Block, and embed the position to obtain the  $x_{in}$ .
- 11:   Perform Spectral-Spatial Recurrent Block;
- 12:   Perform MLP Block.
- 13: **end for**
- 14: Use a trained model to predict categories in the test dataset.
- 15: **return** Predicted label.

---

NGCE is a generalized version of the standard cross-entropy loss function, introducing a parameter  $q$  to control the sensitivity of the loss function. Given the predicted probability  $p$  and the target category  $y$ , NGCE is formulated as:

$$\mathcal{L}_{NGCE} = \frac{1 - (\sum_{k=1}^C y_k \cdot p_k)^q}{C - \sum_{k=1}^C p_k^q} \quad (16)$$

where  $C$  denotes the number of categories;  $y_k$  is the one-hot coding of the target label, with a value of 1 at the index of only the correct category and 0 for the rest;  $p_k$  is the probability value of the model's prediction output; and  $q$  is a hyperparameter controlling the shape of the loss function, which degrades to standard cross-entropy when  $q = 1$ , and increases the robustness of the loss to incorrect predictions when  $q < 1$ . The numerator computes the difference between the predicted probability and the target label and adjusts the degree of nonlinearity of the loss by a  $q$  power, and the denominator is the number of categories minus the sum of the  $q$  powers of the predicted probabilities for each category, ensuring that the loss adjusts to category imbalances. As  $q$  tends to 0, NGCE will tend to be more forgiving of uniform predictions for each category, but will be more sensitive to extreme mispredictions. Therefore, NGCE is suitable for scenarios that require robustness to outliers. In this paper, we demonstrate through multi-group ablation experiments that the model works best when the value of  $q$  takes 0.7.

NCE is a normalized version of the standard cross-entropy loss, which mainly balances the effect of category imbalance on the loss by normalizing the denominator term. The formula of NCE is:

$$\mathcal{L}_{NCE} = - \frac{\sum_{k=1}^C y_k \cdot \log(p_k)}{- \sum_{k=1}^C \log(p_k)} \quad (17)$$

where  $y_k$ ,  $C$ , and  $p_k$  have the same meaning as in Equation (16). In Equation (17), the numerator is the standardized standard deviation loss, which calculates the model's prediction loss for the correct category; the denominator is the logarithmic summation of the prediction probabilities of all categories, which serves as a normalizer and adjusts the prediction probabilities of all categories, preventing the problem of exploding or disappearing gradients in

the case of imbalance of category distributions. The main purpose of the NCE is to adjust the value of the loss function to make it more stable in the different category distributions.

The hybrid loss function proposed in this paper combines NGCE and NCE by introducing two hyperparameters  $\alpha$  and  $\beta$  to control the weights of the two in the final loss, respectively. The hybrid loss function is formulated as:

$$\mathcal{L}_{mix} = \alpha \cdot \mathcal{L}_{NGCE} + \beta \cdot \mathcal{L}_{NCE}. \quad (18)$$

By adjusting the values of  $\alpha$  and  $\beta$ , we can control the relative significance of NGCE and NCE in the loss. If the model is highly sensitive to outliers, increasing the value of  $\alpha$  can enhance the impact of NGCE, and if the model struggles with category imbalance, increasing the value of  $\beta$  can enhance the impact of NCE. NGCE offers robustness against misclassification, while NCE addresses category imbalance through normalization. The combination of the two makes this loss function more stable and adaptable in various data distributions and anomaly scenarios. In this paper, we demonstrate through multiple sets of ablation experiments that the model achieves its best performance when  $\alpha$  and  $\beta$  are both set to 1.0.

### 3. Dataset and Experimental Setup

#### 3.1. Datasets Description

In order to evaluate the effectiveness of the proposed SSFAN model more comprehensively, we selected three widely used hyperspectral datasets, namely Indian Pines (IP), Pavia University (PU) and WHU-Hi-LongKou (WHLK) [65].

- (1) **Indian Pines:** The IP dataset, obtained by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) in June 1992, is a HSI depicting an agricultural region in north-western Indiana, USA. The data consists of  $145 \times 145$  pixels with a spatial resolution of 20 m and includes 220 continuous spectral bands ranging from 400 nm to 2500 nm. For practical purposes, 20 bands with low signal-to-noise ratios were excluded, resulting in 200 remaining bands. The imagery mainly focuses on agricultural and forested regions, containing 10,249 labeled samples across 16 categories, such as corn, grassland, and forest. Due to its high spectral dimensionality, diverse classes, and uneven sample distribution, the IP dataset is often used to assess the performance and reliability of HSI classification algorithms. Figure 5 shows the false color and ground truth maps related to the IP dataset. Moreover, Table 1 offers a detailed summary of the sample count per class, along with the distribution of training and testing samples.
- (2) **Pavia University:** The PU dataset consists of a HSI captured by the Reflective Optics System Imaging Spectrometer (ROSIS) sensor over the University of Pavia in Italy. The image measures  $610 \times 340$  pixels and has a spatial resolution of 1.3 m. It features 115 spectral bands that span wavelengths from 430 nm to 860 nm, with 103 valid bands remaining after filtering out noisy data. The dataset is categorized into 9 classes, including buildings, roads, and vegetation, totaling 42,776 labeled samples. Due to its high resolution and extensive spectral information, the PU dataset is commonly used as a standard for evaluating the effectiveness of HSI classification algorithms in urban settings. Figure 6 shows the false color and ground truth maps related to the PU dataset. Moreover, Table 1 offers a detailed summary of the sample count per class, along with the distribution of training and testing samples.
- (3) **WHU-Hi-LongKou:** The WHLK dataset was gathered from an agricultural region in Longkou Town, Hubei Province, China, on 17 July 2018, between 1:49 p.m. and 2:37 p.m. Data collection was performed using a DJI Matrice 600 Pro drone, which was fitted with a Headwall Nano-Hyperspec imaging sensor that has an 8 mm focal length. The area studied featured six different crop types: corn, cotton, sesame, broadleaf soybean, narrowleaf soybean, and rice. The UAV flew at an altitude of 500 m, taking images with a resolution of  $550 \times 400$  pixels and capturing 270 spectral bands with wavelengths from 400 to 1000 nanometers. The images have a spatial resolution of

about 0.463 m. Figure 7 shows the false color and ground truth maps related to the WHLK dataset. Moreover, Table 1 offers a detailed summary of the sample count per class, along with the distribution of training and testing samples.

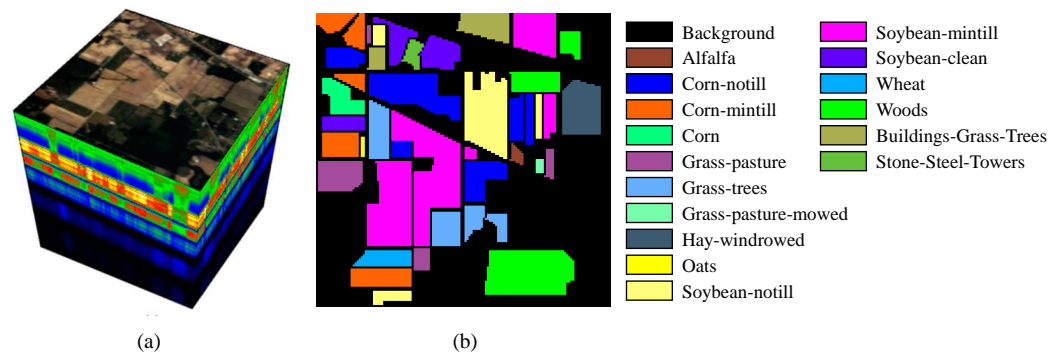


Figure 5. IP dataset. (a) False-color map. (b) Ground-truth map.

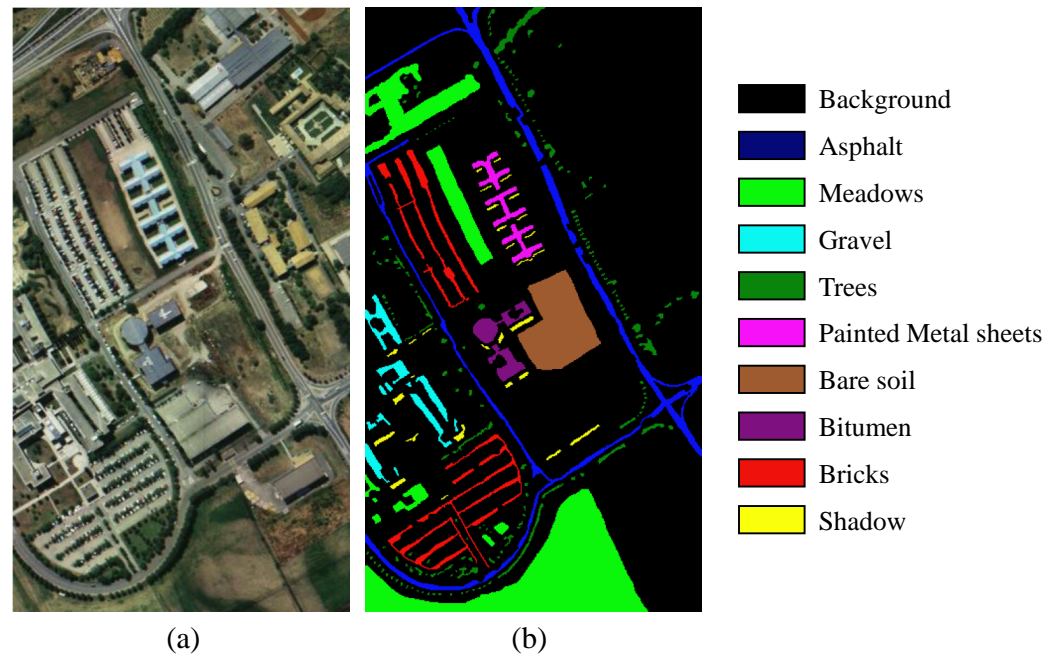


Figure 6. PU dataset. (a) False-color map. (b) Ground-truth map.

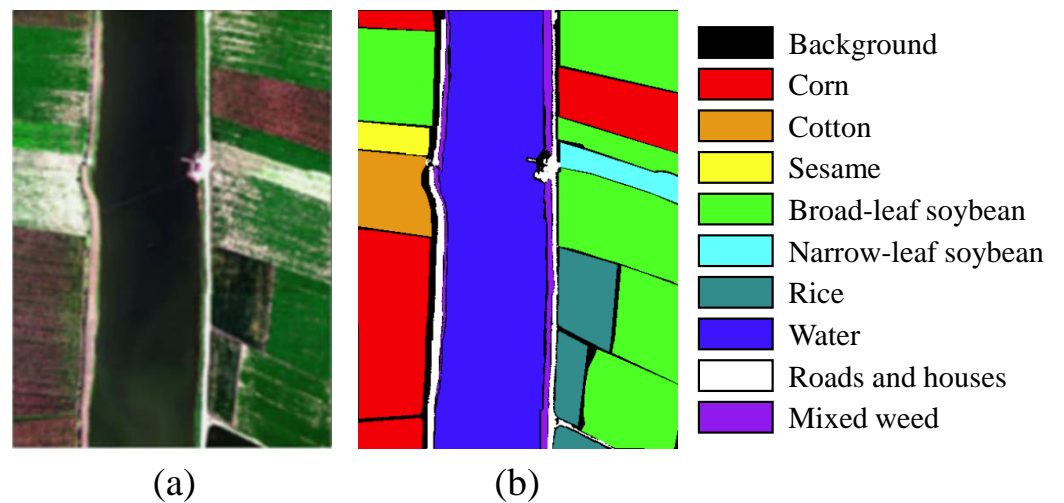


Figure 7. WHLK dataset. (a) False-color map. (b) Ground-truth map.

**Table 1.** The quantities of training and testing samples for the IP dataset, the PU dataset, and the WHLK dataset are presented.

NO.	IP			PU			WHLK		
	Class	Training Samples	Test Samples	Class	Training Samples	Test Samples	Class	Training Samples	Test Samples
1	Alfalfa	5	41	Asphalt	332	6299	Corn	3452	31,059
2	Corn-notill	143	1285	Meadows	932	17,717	Cotton	838	7536
3	Corn-mintill	83	747	Gravel	105	1994	Sesame	304	2727
4	Corn	24	213	Trees	153	2911	Broad-leaf soybean	6322	56,890
5	Grass-pasture	48	435	Mental sheets	67	1278	Narrow-leaf soybean	416	3736
6	Grass-trees	73	657	Bare soil	251	4778	Rice	1186	10,668
7	Grass-pasture-mowed	3	25	Bitumen	67	1263	Water	6706	60,350
8	Hay-windrowed	48	430	Bricks	184	3498	Roads and houses	713	6411
9	Oats	2	18	Shadow	47	900	Mixed weed	522	4707
10	Soybean-notill	97	875						
11	Soybean-mintill	245	2210						
12	Soybean-clean	59	534						
13	Wheat	20	185						
14	Woods	126	1139						
15	Buildings-Grass-Trees	39	347						
16	Stone-Steel-Towers	9	84						
-	Total	1024	9225	Total	2138	40,638	Total	20,459	184,084

### 3.2. Evaluation Indicators

This experiment mainly adopts three evaluation metrics widely used in hyperspectral classification: overall accuracy (OA), average accuracy (AA) and kappa coefficient (Kappa). In practice, obtaining labeled data is usually more difficult than obtaining unlabeled data, so the number of labeled training samples is often more limited. The test set is typically obtained from more extensive unlabeled real scene data during the model deployment phase. As a result, we split the dataset into two groups: a test dataset and a train dataset, with a ratio of 1:9. In the end, the model must be used and its performance confirmed on a sizable amount of test data, even though it can only rely on a small amount of labeled data for training. Five experiments were carried out for each method in the comparison and ablation experiments to prevent errors caused by randomly choosing training and test samples when dividing the dataset. The better the model's performance, as indicated by the larger mean and smaller standard deviation in the experimental results.

OA is a metric used to assess the overall performance of the classification model on all test samples. It is defined as the ratio of the number of samples that the model correctly classified to the total number of samples. This indicates the overall accuracy of the model's classification on the entire dataset. When used on datasets with a fairly balanced distribution of sample categories, OA is a more understandable statistic for rapidly assessing the model's overall performance. However, if the categories are imbalanced, it may obscure the model's performance on smaller categories. With  $N$  total samples and  $M$  being the number of samples that the model properly classified, the overall accuracy can be computed as follows:

$$OA = \frac{M}{N}. \quad (19)$$

AA is a commonly used performance evaluation metric to measure the average classification accuracy of a classification model over all categories, which can be a good indicator of the ability of a classification model to recognize each category, especially in the case of category imbalance. There are  $K$  categories, and for each category  $i$ , its classification accuracy  $A_i$  is defined as:

$$A_i = \frac{TP_i}{TP_i + FN_i} \quad (20)$$

where  $TP_i$  is the number of correctly categorized samples in the  $i$  th class, and  $FN_i$  is the number of samples in the  $i$  th class that have been incorrectly categorized as other classes. Thus AA is calculated as:

$$AA = \frac{1}{K} \sum_{i=1}^K A_i. \quad (21)$$

Kappa is a statistical metric used to evaluate the consistency between the predicted results of a classification model and the true labels. Compared to OA, Kappa takes into account chance consistency (i.e., the correct classification results that could have been produced when the classifier made a random guess), and as a result, it is more dependable when there is an imbalance in categories. The Kappa statistic assesses how much better the classifier's performance is compared to random guessing. It evaluates the performance of the model by calculating the difference between the actual accuracy and the desired random accuracy. Let  $P_e$  be the desired stochastic accuracy, i.e., the correctness of the model's prediction when randomly classifying, which is calculated from the marginal distributions of the true and predicted categories. The calculation formula is:

$$P_e = \sum_{i=1}^K \left( \frac{N_{true_i}}{N} \cdot \frac{N_{predict_i}}{N} \right) \quad (22)$$

here,  $N_{true_i}$  denotes the number of samples that actually belong to the  $i$ th class, while  $N_{predict_i}$  represents the number of samples predicted to be in the  $i$ th class. The Kappa statistic is then computed as follows:

$$Kappa = \frac{OA - P_e}{1 - P_e}. \quad (23)$$

In addition to using evaluation metrics for model results, we introduce evaluation metrics for the model itself: parameters [66] and Multiply-Accumulate Operations (MACs) [67,68]. The number of parameters directly affects the storage requirements and computational complexity of the model. Larger models require more memory and computational resources for training and inference. MACs refers to the total number of multiplication and addition operations. It is a measure of the computational complexity of the model, especially in the inference process. When comparing experiments, the introduction of model parameters and MACs helps to fully assess the complexity and computational requirements of the model. The number of model parameters provides an indication of the model's capacity, which helps to determine its learning capability and storage requirements. MACs, on the other hand, measure the computational burden during inference and help to compare the computational efficiency of different models.

### 3.3. Comparison Methods and Implement Details

This research chooses and compares several comparison approaches for HSI classification in order to show the efficacy of the proposed SSFAN method. They fall into two categories specifically: transformer-based and convolutional neural network-based. 1D CNN [34], 2D CNN [35], 3D CNN [37], SSRN [48], and Hybrid CNN [38] are among the CNN-based comparison techniques; SpectralFormer [58] and SSFTT [60] are Transformer-based techniques.

We use Pytorch [69,70] framework to write the model and training code to train the model on NVIDIA GeForce RTX 2080 Ti. Three datasets, IP, PU and WHLK, which are very widely used and representative, are selected for training. The IP and PU datasets are mainly used for comparison and ablation experiments, while WHLK is mainly used in the Discussion section to discuss the model's ability to handle large datasets. A combination of NGCE and NCE is used for the loss function, which is set to have a  $q$  value of 0.7 for NGCE and 1.0 for both loss function weight parameters. The network parameters were updated using the Adam optimizer, and the learning rate was set to 0.001. The entire training process was set to 100 epochs, and every 5 epochs we evaluated the performance of the model on the test set. The batch size of training is set to 100 to ensure convergence and reasonable utilization of computational resources during the training process. Hyperspectral image



data is limited, so a larger test set with diverse samples is essential to ensure robust model generalization. This approach enhances the stability, representativeness, and validity of the evaluation while helping to mitigate overfitting in high-dimensional data. To optimize generalization performance, 10% of the dataset is used for training, while the remaining 90% is reserved for testing.

#### 4. Experimental Results and Analysis

##### 4.1. Compare Experiment

The classification performance of various models on the Indian Pine dataset is displayed in Table 2. The tabular data shows that the SSFAN model outperforms other models in terms of classification accuracy across the majority of classes, particularly in more complicated classes like Alfalfa, Corn-mintill, Grass-pasture, and Soybean-clean. Compared to 1D CNN (43.75%), the SSFAN model achieves a classification accuracy of 96.85% on the Alfalfa class. The accuracy of classification compared to 1D CNN (43.75%), 2D CNN (48.78%), and 3D CNN (41.46%) is noticeably greater. In contrast, the SpectralFormer and SSFTT models have accuracies of 92.30% and 92.39% on this class, which are also excellent but still lower than the performance of SSFAN. For the Corn-notill category, the SSFAN model achieves a classification accuracy of 99.33%, which is slightly lower than that of SpectralFormer (99.72%) and SSFTT (99.61%), but the overall performance is still very good, showing SSFAN's robustness in dealing with spectral-spatial features. In the Corn-mintill class, the SSFAN model again performs well, achieving an accuracy of 97.31%, outperforming all other models. In comparison, 1D CNN only achieved 56.72% accuracy, 3D CNN 79.36%, and SSRN 94.23%, indicating that SSFAN is more capable of feature extraction on complex classes. In the Grass-pasture class, the SSFAN model leads all other models with an accuracy of 99.68%, indicating its significant advantage in dealing with vegetation-based features. Similarly, the SSFAN model also performs well in the Grass-pasture-mowed and Hay-windrowed categories, achieving 99.23% and 100.00% accuracies, respectively. In the Oats class, although the SSFAN model achieved 95.66% accuracy, which is slightly lower than SSFTT's 97.89%, it still outperforms most of the other models and shows high classification ability. In terms of OA, AA, and the Kappa, the SSFAN model exhibits strong performance, achieving an overall accuracy of 98.68%. This is significantly higher than the 79.37% and 84.47% recorded by the 1D CNN and 2D CNN, respectively, and the 91.03% obtained by the 3D CNN. The SSFAN model also surpasses the Transformer-based models, SpectralFormer and SSFTT, by 2.22% and 1.72% in OA, 4.00% and 5.19% in AA, and 2.94% and 1.94% in the Kappa, respectively. SSFAN demonstrates a notable improvement in classification accuracy across the majority of classes, particularly in challenging categories like Corn-mintill and Grass-pasture. This enhancement is primarily attributed to the model's innovative approach to spectral-spatial feature extraction, which incorporates an attention mechanism and optimizes the feature extraction module.

Table 3 shows the classification performance of various models on the PU dataset. Based on the PU dataset, the experimental findings demonstrate that SSFAN achieves an OA of 99.83%, which is much greater than the other models. SpectralFormer and SSFTT had OAs of 99.11% and 99.51%, respectively, among them. SSFAN outperforms SpectralFormer by 0.72% and SSFTT by 0.32% percentage points. This suggests that SSFAN can improve overall classification accuracy by more effectively extracting information from spectral and spatial variables. SpectralFormer (98.35%) and SSFTT (98.95%) perform substantially worse on AA than SSFAN (99.58%). Compared to SpectralFormer, SSFAN outperforms it by 1.23% and SSFTT by 0.63% percentage points on AA. This outcome demonstrates how stable and consistent SSFAN is when handling different kinds of data, particularly when it comes to categorization accuracy. The highest performance of any model is achieved by SSFAN, with a Kappa coefficient of 99.77%. By contrast, SSFAN outperforms SpectralFormer by 1.52% and SSFTT by 0.42%. The Kappa coefficients of SpectralFormer and SSFTT are 98.25% and 99.35%, respectively. This suggests that in terms of classification accuracy and consistency, SSFAN has a considerable advantage when

processing data that is imbalanced between categories and complex scenarios. In terms of specific classes, the SSFAN model demonstrates excellent performance in several classes. For example, the accuracy on the Asphalt class reaches 99.93%, which is significantly higher than the accuracy in other models. Specifically, with 98.99% and 99.79% for SpectralFormer and SSFTT, SSFAN improves 0.94% and 0.14%, respectively. This indicates that SSFAN shows a significant advantage in dealing with the Asphalt surface, a category with high complexity. On the Gravel class, SSFAN achieves a classification accuracy of 100.00%, which is significantly higher than that of SpectralFormer (98.98%) and SSFTT (99.71%), with an improvement of 1.02% and 0.29% percentage points, respectively. This indicates that SSFAN has excellent performance on fine-grained classification and complex texture categories. In the Bare Soil class, SSFAN achieves an accuracy of 99.98%, outperforming other models. Especially, compared with the 99.43% of Hybrid CNN and the 99.85% of SSFTT, SSFAN improves 0.55% and 0.13%, respectively, which further validates SSFAN's ability to deal with the soil category with high classification accuracy. In the Bitumen class, the classification accuracy of SSFAN is 99.99%, which is significantly higher than that of 1D CNN (19.40%) and 2D CNN (70.92%), which demonstrates its superior performance in dealing with high-contrast categories. In the Shadows class, the classification accuracy of SSFAN is 99.95%, which is higher than that of SpectralFormer (97.40%) and SSFTT (96.50%), with an improvement of 2.55% and 3.45% percentage points, respectively, which demonstrates the strong performance of SSFAN in dealing with Shadows, which is a category with complex spectral features. The experimental results of the SSFAN model on the PU dataset demonstrate its significant advantages in HSI classification tasks. SSFAN not only outperforms other models in metrics such as OA, AA, and Kappa, but also shows higher accuracy and stability in class-specific classification tasks. These results demonstrate the power of the SSFAN model in dealing with complex spectral-spatial data.

**Table 2.** An evaluation of the classification performance of the IP dataset utilizing different methods. (The optimal result is highlighted in bold, with comparisons made by first examining the mean and then the standard deviation).

Class	1D CNN [34]	2D CNN [35]	3D CNN [37]	SSRN [48]	Hybrid CNN [38]	SpectralFormer [58]	SSFTT [60]	SSFAN (Ours)
Alfalfa	43.75 ± 0.01	48.78 ± 0.09	41.46 ± 0.15	83.15 ± 0.00	87.80 ± 0.15	92.30 ± 0.14	92.39 ± 0.04	<b>96.85 ± 0.04</b>
Corn-notill	77.93 ± 0.12	78.13 ± 0.05	90.51 ± 0.01	95.31 ± 0.02	94.39 ± 0.03	99.72 ± 0.12	<b>99.61 ± 0.02</b>	99.33 ± 0.00
Corn-mintill	56.72 ± 0.05	83.51 ± 0.01	79.36 ± 0.18	94.23 ± 0.09	96.52 ± 0.01	96.12 ± 0.01	95.95 ± 0.00	<b>97.31 ± 0.02</b>
Corn	45.18 ± 0.00	47.42 ± 0.12	46.01 ± 1.44	90.68 ± 0.02	83.89 ± 0.02	98.98 ± 0.10	99.06 ± 0.00	<b>99.72 ± 0.00</b>
Grass-pasture	87.57 ± 0.01	75.12 ± 0.04	95.17 ± 0.34	97.79 ± 0.05	98.16 ± 0.14	95.14 ± 0.00	94.94 ± 0.00	<b>99.68 ± 0.00</b>
Grass-trees	98.63 ± 0.00	92.99 ± 0.06	99.70 ± 0.16	98.67 ± 0.01	<b>99.54 ± 0.02</b>	97.18 ± 0.01	97.80 ± 0.00	99.15 ± 0.00
Grass-pasture-mowed	65.11 ± 0.03	60.00 ± 0.05	88.00 ± 0.05	97.92 ± 0.04	92.97 ± 0.00	96.15 ± 0.14	97.85 ± 0.04	<b>99.23 ± 0.01</b>
Hay-windrowed	97.36 ± 0.01	98.37 ± 0.00	<b>100.00 ± 0.00</b>	99.26 ± 0.00	<b>100.00 ± 0.00</b>	<b>100.00 ± 0.00</b>	99.73 ± 0.00	<b>100.00 ± 0.00</b>
Oats	37.14 ± 1.34	66.67 ± 0.34	48.89 ± 0.13	89.49 ± 0.01	86.27 ± 0.02	94.69 ± 0.25	<b>97.89 ± 0.04</b>	95.66 ± 0.04
Soybean-notill	66.03 ± 0.01	87.77 ± 0.01	86.06 ± 0.07	97.48 ± 0.06	<b>97.94 ± 0.04</b>	93.13 ± 0.21	93.63 ± 0.00	97.67 ± 0.02
Soybean-mintill	82.49 ± 0.00	89.09 ± 0.03	97.51 ± 0.07	98.16 ± 0.03	<b>99.50 ± 0.16</b>	95.92 ± 0.36	96.82 ± 0.01	97.45 ± 0.01
Soybean-clean	73.49 ± 0.34	63.67 ± 0.14	74.91 ± 0.06	93.07 ± 0.11	94.57 ± 0.01	96.66 ± 0.01	95.08 ± 0.01	<b>99.25 ± 0.00</b>
Wheat	99.30 ± 0.02	100.00 ± 0.01	99.46 ± 0.04	98.59 ± 0.01	94.59 ± 0.05	99.01 ± 0.02	98.25 ± 0.03	<b>99.89 ± 0.00</b>
Woods	93.78 ± 0.04	95.33 ± 0.11	99.74 ± 0.12	<b>99.72 ± 0.18</b>	99.29 ± 0.02	98.45 ± 0.07	99.75 ± 0.00	99.51 ± 0.00
Buildings-Grass-Trees-Drives	55.39 ± 0.08	66.76 ± 0.02	84.10 ± 0.07	93.31 ± 0.13	92.35 ± 0.06	97.17 ± 0.21	<b>97.87 ± 0.01</b>	97.82 ± 0.01
Stone-Steel-Towers	81.54 ± 0.01	91.57 ± 0.05	93.98 ± 0.00	93.79 ± 0.10	<b>97.98 ± 0.00</b>	91.10 ± 0.03	91.14 ± 0.08	95.65 ± 0.02
OA (%)	79.37 ± 0.02	79.37 ± 0.02	91.03 ± 0.24	94.78 ± 0.16	96.62 ± 0.18	96.46 ± 0.02	96.96 ± 0.42	<b>98.68 ± 0.23</b>
AA (%)	70.87 ± 0.75	77.83 ± 2.33	82.18 ± 1.33	94.67 ± 0.41	96.66 ± 0.13	94.45 ± 1.98	93.26 ± 1.75	<b>98.45 ± 0.32</b>
Kappa (×100)	76.28 ± 0.04	82.24 ± 1.14	89.68 ± 1.23	94.08 ± 0.12	96.29 ± 0.08	95.53 ± 0.78	96.53 ± 0.48	<b>98.47 ± 0.27</b>

**Table 3.** An evaluation of the classification performance of the PU dataset utilizing different methods. (The optimal result is highlighted in bold, with comparisons made by first examining the mean and then the standard deviation).

Class	1D CNN [34]	2D CNN [35]	3D CNN [37]	SSRN [48]	Hybrid CNN [38]	SpectralFormer [58]	SSFTT [60]	SSFAN (Ours)
Asphalt	84.91 ± 0.01	93.28 ± 0.31	93.38 ± 0.01	95.35 ± 0.21	95.51 ± 0.15	98.99 ± 0.01	99.79 ± 0.00	<b>99.93 ± 0.00</b>
Meadows	94.14 ± 0.08	94.90 ± 0.18	93.99 ± 0.11	94.69 ± 0.00	99.49 ± 0.00	99.16 ± 0.00	99.96 ± 0.00	<b>99.99 ± 0.00</b>
Gravel	47.38 ± 0.11	75.55 ± 0.13	90.19 ± 0.23	96.48 ± 0.13	94.18 ± 0.12	98.98 ± 0.01	99.71 ± 0.00	<b>100.00 ± 0.00</b>
Trees	82.28 ± 0.09	93.87 ± 0.19	91.29 ± 0.09	96.37 ± 0.19	<b>99.55 ± 0.00</b>	97.37 ± 0.02	98.07 ± 0.00	98.77 ± 0.00
Painted metal sheets	99.76 ± 0.17	97.98 ± 0.07	95.47 ± 0.02	99.69 ± 0.00	96.71 ± 0.01	98.18 ± 0.01	99.78 ± 0.00	<b>100.00 ± 0.00</b>
Bare Soil	77.67 ± 0.10	70.05 ± 0.02	93.85 ± 0.06	97.49 ± 0.02	99.43 ± 0.0	99.25 ± 0.00	99.85 ± 0.00	<b>99.98 ± 0.00</b>
Bitumen	19.40 ± 0.22	70.92 ± 0.12	81.45 ± 0.14	95.36 ± 0.01	<b>99.99 ± 0.00</b>	99.07 ± 0.01	99.26 ± 0.00	99.75 ± 0.00
Self-Blocking Bricks	70.01 ± 0.03	90.30 ± 0.23	92.73 ± 0.03	91.49 ± 0.02	95.97 ± 0.01	97.84 ± 0.02	98.14 ± 0.00	<b>99.35 ± 0.00</b>
Shadows	98.55 ± 0.04	97.89 ± 0.15	95.46 ± 0.14	95.90 ± 0.12	95.22 ± 0.16	97.40 ± 0.00	96.50 ± 0.00	<b>99.95 ± 0.00</b>
OA (%)	83.50 ± 0.13	90.19 ± 0.56	92.01 ± 0.16	95.87 ± 0.02	98.16 ± 0.12	99.11 ± 0.14	99.51 ± 0.09	<b>99.83 ± 0.04</b>
AA (%)	74.90 ± 0.24	87.31 ± 0.21	90.45 ± 0.15	95.86 ± 0.23	97.35 ± 0.34	98.35 ± 0.33	98.95 ± 0.14	<b>99.58 ± 0.08</b>
Kappa (×100)	77.90 ± 0.02	87.52 ± 0.12	90.87 ± 0.02	95.78 ± 0.17	97.57 ± 0.16	98.25 ± 0.15	99.35 ± 0.12	<b>99.77 ± 0.03</b>

Figures 8 and 9 illustrate the classification performance of various models on the IP dataset and the PU dataset, respectively. As can be seen from the figures, 1D CNN mainly utilizes the spectral information for classification but ignores the spatial information, and there is significant noise and misclassification in the image, especially in the boundary region between different categories, indicating the limitation of 1D CNN in capturing spatial features. 2D CNN focuses on spatial feature extraction but does not make full use of the spectral information, and has a better boundary than 1D CNN in terms of boundary clarity and coherence, but there are still some misclassifications, especially in regions with high heterogeneity. 3D CNN utilizes both spectral and spatial information, and thus provides more accurate classification results with significantly less noise compared to 1D and 2D CNN. However, there are still classification errors in some complex regions, suggesting that 3D CNN, although effective, may still have some deficiencies when dealing with complex spectral-spatial information. SSRN is a model based on spectral-spatial residual networks, and the classification accuracy of SSRN is higher than that of the previous CNN models, especially in terms of the category boundaries and details that are closer to the real labels. Hybrid CNN combines the advantages of 1D and 2D CNNs, taking into account both spectral and spatial features. The classification results of this model are relatively smoother and the improvement of classification accuracy is obvious, especially in the complex region, which is better than the simple 1D or 2D CNN. SpectralFormer and SSFTT are the models based on the Transformer architecture, which emphasize the global dependence of spectral information. The classification outcomes indicate that SpectralFormer is more effective at handling intricate spectral data, with its results aligning more closely with the actual labels in terms of overall structure. However, it may fall short in accurately capturing certain details and spatial characteristics. SSFTT adopts the advanced spectral-spatial feature extraction and fusion technology, which better preserves the spatial details, and the classification results are clear and close to the real labels. Compared with the previous models, SSFTT performs more stably when dealing with complex regions. The SSFAN model provides the best classification results, fusing spectral and spatial features, and is further optimized based on the use of the attention mechanism. The classification outcomes indicate that the model excels in accuracy and detail preservation, closely resembling the actual labeled maps, which highlights its robust classification capabilities. SSFAN achieves the highest performance by effectively utilizing both spectral and spatial features while maintaining detail, significantly surpassing both the CNN and Transformer models. The figure shows that the SSFAN model performs exceptionally well on the PU dataset, almost completely matching the classification results of the actual labels, highlighting its outstanding capability in HSI classification tasks. The rest of the models also achieve good results, such as Hybrid CNN with smoother classification performance after fusing the advantages of 1D and 2D CNN. SpectralFormer slightly underperforms the other methods in details although it improves the global dependence on spectral information. SSFTT performs better in details and accuracy, close to the real labeled image.

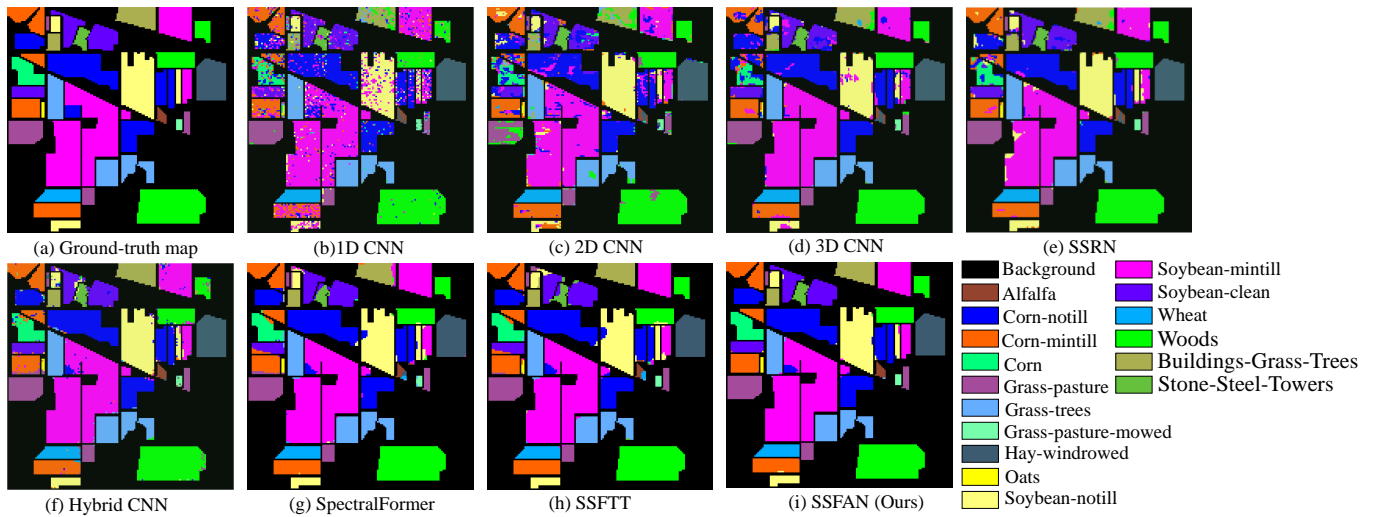


Figure 8. Classification map of IP dataset with different models.

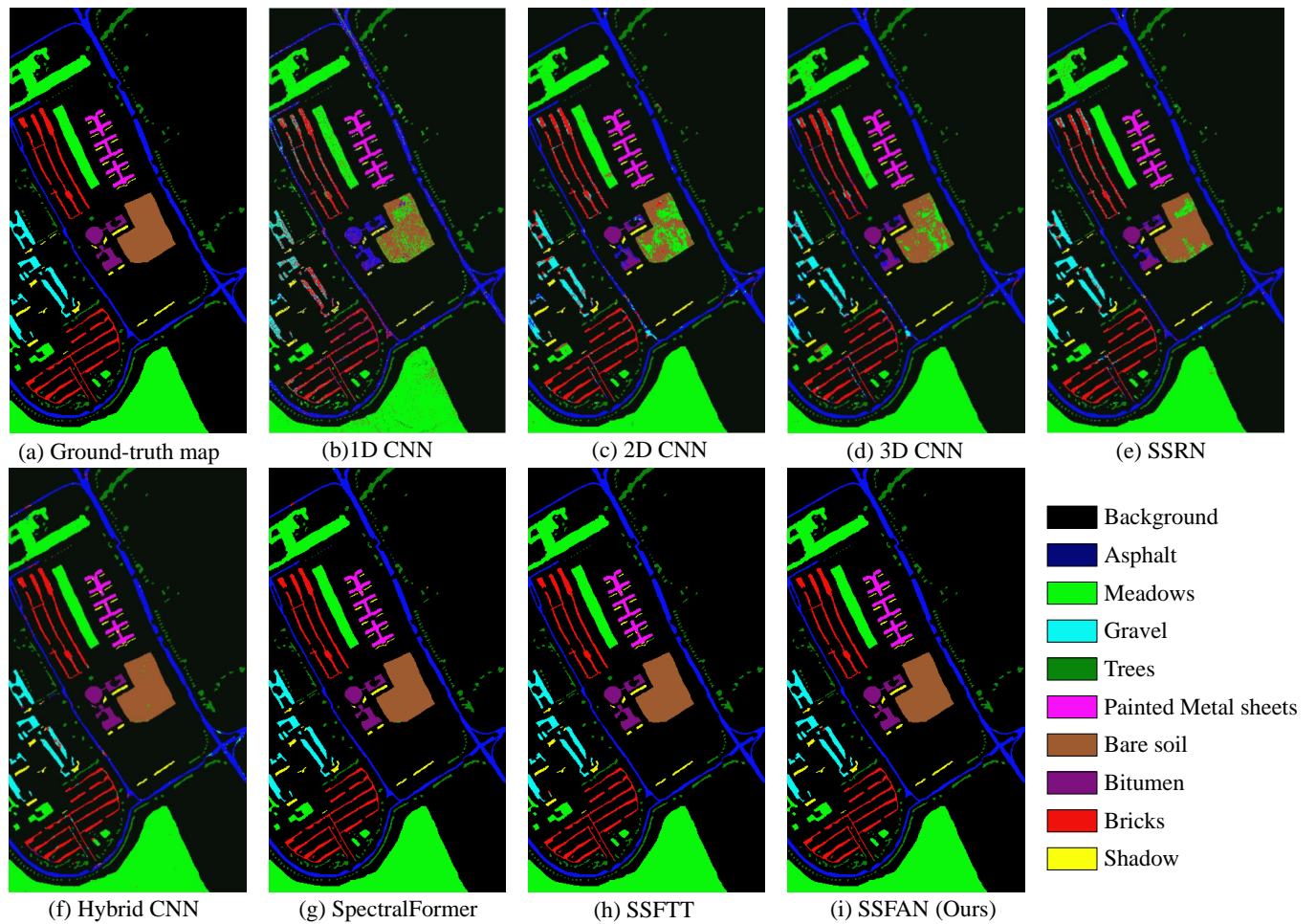
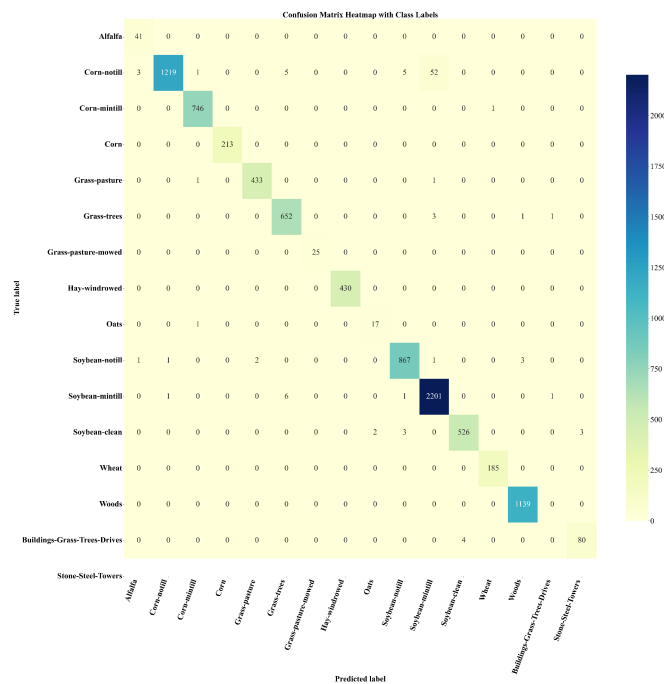


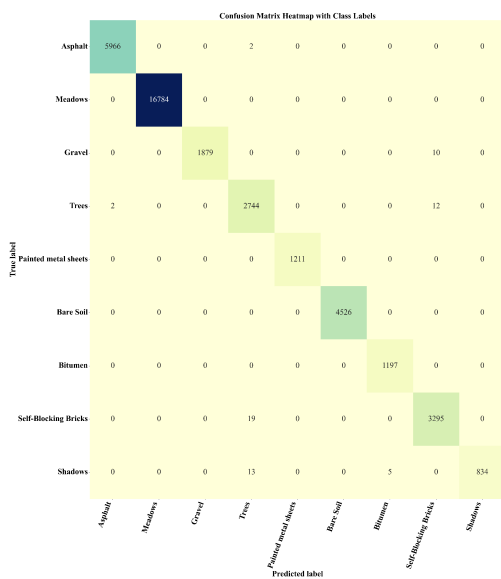
Figure 9. Classification map of PU dataset with different models.

The heat map of the confusion matrix illustrates the accuracy of the classification model's predictions for each class using color gradients. It allows for quick identification of misclassifications, aids in assessing model performance, and highlights areas for potential improvement, making it a valuable visual tool for diagnosing and optimizing classification issues. Figure 10 presents the confusion matrix derived from experiments conducted on three datasets. In Figure 10a, the heat map for the IP dataset shows OA, AA, and Kappa

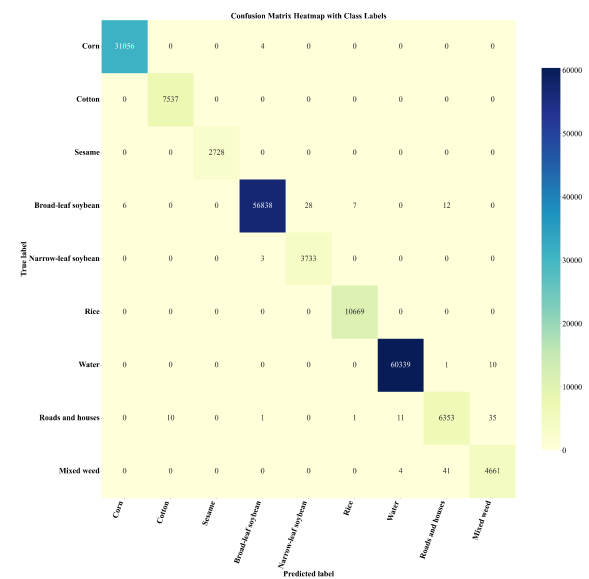
values of 98.87%, 98.77%, and 98.71%, respectively. Figure 10b displays the heat map for the PU dataset, with OA, AA, and Kappa values of 99.83%, 99.58%, and 99.78%, respectively. Figure 10c depicts the heat map for the WHLK dataset, showing OA, AA, and Kappa values of 99.90%, 99.77%, and 99.87%, respectively. The confusion matrix heat map allows for a quick assessment of the samples across the five classifications. For instance, in Figure 10a, 52 samples correctly identified as Corn-notill are mistakenly classified as Soybean-notill, which has the highest error rate in the IP dataset. In Figure 10b, 19 samples correctly labeled as Self-Blocking Bricks are incorrectly categorized as Trees, the class with the highest error rate in the PU dataset. Similarly, in Figure 10c, 35 samples correctly identified as Roads and houses are misclassified as Mixed weed, the class with the highest error rate in the WHLK dataset. Analyzing the heat map of the confusion matrix provides a clear understanding of the model’s performance across each class.



(a) Indian Pines dataset confusion matrix heatmap



(b) Pavia University dataset confusion matrix heatmap



(c) LongKou dataset confusion matrix heatmap

Figure 10. Heat map of SSFAN’s confusion matrix on three different datasets. (a) IP. (b) PU. (c) WHLK.

#### 4.2. Ablation Experiment

To thoroughly assess the effectiveness of the proposed SSFAN method, we performed ablation experiments on the PU dataset, examining various combinations of the model's components. We evaluated three distinct combinations to determine the contribution of each component to SSFAN's classification accuracy. The results of all experiments are presented in Table 4. Specifically, the model was divided into three components: PSSB, Scan Block, and SEMB. The PSSB plays a crucial role in the overall model. When the PSSB module is removed, the SSFAN model experiences a reduction in OA, AA, and Kappa values by 12.71%, 16.24%, and 10.54%, respectively, when tested on the PU dataset. We also replaced the Scan Block with a simple flattening operation for ablation experiments, which led to a decrease in OA, AA, and Kappa values by 3.11%, 2.42%, and 3.14%, respectively. This result indicates that the Scan Block, by extracting multi-scale features from the central region and its surroundings, enhances the model's ability to capture local information of varying sizes, thereby improving classification performance. In the third scenario, which excluded the SEMB module, the SSFAN model experienced a decrease in OA, AA, and Kappa values by 6.06%, 5.34%, and 5.01%, respectively, when tested on the PU dataset. The SEMB module effectively captures complex sequential features and dependencies, with adaptive feature weight assignment, enabling SSFAN to handle more complex features, particularly in large-scale hyperspectral datasets. Overall, the comprehensive analysis of experimental results further confirms the effectiveness of the SSFAN model.

**Table 4.** Ablation analysis of the SSFAN model based on the PU dataset. (The optimal result is highlighted in bold, with comparisons made by first examining the mean and then the standard deviation).

Cases	Component			Indicators		
	PSSB	Scan Block	SEMB	OA (%)	AA (%)	Kappa ( $\times 100$ )
1	✗	✓	✓	87.12 $\pm$ 1.14	83.34 $\pm$ 1.45	89.23 $\pm$ 1.13
2	✓	✗	✓	96.72 $\pm$ 0.68	97.16 $\pm$ 0.34	96.63 $\pm$ 0.56
3	✓	✓	✗	93.77 $\pm$ 0.23	94.24 $\pm$ 1.01	94.76 $\pm$ 0.47
4	✓	✓	✓	<b>99.83 <math>\pm</math> 0.04</b>	<b>99.58 <math>\pm</math> 0.08</b>	<b>99.77 <math>\pm</math> 0.05</b>

## 5. Discussion

### 5.1. Advantages of Parallel Spectral–Spatial Feature Extraction Block

Compared with the single-channel Spectral-Spatial Feature Extraction Block (SSSB), the parallel-channel Parallel Spectral-Spatial Feature Extraction Block (PSSB) can extract richer spectral-spatial information. To assess the performance of this module, we carried out several experiments using three distinct datasets, evaluating the No Spectral-Spatial Feature Extraction Block (NSSB), the single streams Spectral-Spatial Feature Extraction Block (SSSB), and the parallel streams Spectral-Spatial Feature Extraction Block (PSSB). The experimental findings are presented in Table 5. The table indicates that the Spectral-Spatial Feature Extraction Block greatly improves the classification performance. Specifically, the OA metrics on the IP, Pavia University, and WHLK datasets are improved by 11.91%, 12.71%, and 12.32%, respectively, using PSSB compared to NSSB, and the AA and Kappa metrics also show similar increases. These ablation experimental results indicate that the PSSB module has a significant performance enhancement in the overall model, further validating its importance and effectiveness in the classification task. To further illustrate the advantages of the parallel streams, we performed comparative experiments between PSSB and SSSB. The results indicate that PSSB surpasses SSSB in all three metrics: OA, AA and Kappa, highlighting the enhanced effectiveness of the parallel channel in extracting spectral-spatial features.

**Table 5.** Ablation Experiment of Parallel Spectral–Spatial Feature Extraction Block. (The optimal result is highlighted in bold, with comparisons made by first examining the mean and then the standard deviation).

Module	IP				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
NSSB	86.77 $\pm$ 1.33	76.65 $\pm$ 2.78	85.47 $\pm$ 1.56	<b>64.65 <math>\pm</math> 0.38</b>	<b>1.34 <math>\pm</math> 1.23</b>
SSSB	98.57 $\pm$ 0.66	97.91 $\pm$ 1.45	98.37 $\pm$ 0.76	69.80 $\pm$ 0.53	1.68 $\pm$ 0.53
PSSB	<b>98.68 <math>\pm</math> 0.23</b>	<b>98.45 <math>\pm</math> 0.32</b>	<b>98.47 <math>\pm</math> 0.27</b>	78.66 $\pm$ 0.15	1.87 $\pm$ 0.01
Module	PU				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
NSSB	87.12 $\pm$ 1.14	83.34 $\pm$ 1.45	89.23 $\pm$ 1.13	<b>245.72 <math>\pm</math> 4.56</b>	<b>6.12 <math>\pm</math> 1.34</b>
SSSB	99.80 $\pm$ 0.03	<b>99.62 <math>\pm</math> 0.09</b>	99.65 $\pm$ 0.17	272.05 $\pm$ 1.29	7.31 $\pm$ 0.08
PSSB	<b>99.83 <math>\pm</math> 0.04</b>	99.58 $\pm$ 0.08	<b>99.77 <math>\pm</math> 0.05</b>	310.72 $\pm$ 0.52	8.17 $\pm$ 0.06
Module	WHLK				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
NSSB	87.56 $\pm$ 1.08	85.10 $\pm$ 1.10	86.34 $\pm$ 1.31	<b>980.66 <math>\pm</math> 78.34</b>	<b>30.45 <math>\pm</math> 6.26</b>
SSSB	99.68 $\pm$ 0.20	<b>99.72 <math>\pm</math> 0.07</b>	99.84 $\pm$ 0.07	1364.08 $\pm$ 6.19	36.34 $\pm$ 1.20
PSSB	<b>99.88 <math>\pm</math> 0.02</b>	99.70 $\pm$ 0.05	<b>99.85 <math>\pm</math> 0.03</b>	1452.56 $\pm$ 51.53	37.05 $\pm$ 2.22

To validate the effectiveness of the proposed PSSB, which employs two Spectral–Spatial Feature Extraction Blocks, we conducted a series of comparative experiments on the PU hyperspectral dataset, with the results presented in Table 6. From the results, the OA and Kappa value using PSSB reached 99.83% and 99.77%, respectively, further demonstrating the effectiveness of PSSB. Meanwhile, as the number of Spectral–Spatial Feature Extraction Blocks increased, the model’s parameter count gradually increased; however, its performance did not improve and instead exhibited a declining trend. This may indicate that the model is experiencing overfitting.

**Table 6.** Effect of Different Number of Spectral–Spatial Feature Extraction Block on Experimental Results on PU Dataset. (The optimal result is displayed in bold, compare the mean first, then compare the standard deviation).

Number of Spectral–Spatial Feature Extraction Block	Evaluation Indicators			
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Parameters
1	99.80 $\pm$ 0.03	<b>99.62 <math>\pm</math> 0.09</b>	99.65 $\pm$ 0.17	<b>22.80 K</b>
2 (Ours)	<b>99.83 <math>\pm</math> 0.04</b>	99.58 $\pm$ 0.08	<b>99.77 <math>\pm</math> 0.05</b>	39.86 K
3	99.81 $\pm$ 0.01	99.55 $\pm$ 0.04	99.75 $\pm$ 0.02	59.10 K
4	99.76 $\pm$ 0.14	99.50 $\pm$ 0.24	99.69 $\pm$ 0.19	77.51 K

## 5.2. Discussion About Patch Size

HSIs not only contain rich spectral information, but also carry spatial information. Classification using a pixel point alone may ignore the spatial context around the pixel point, while segmenting the image into patches can effectively utilize the local spatial information. Each patch’s size, or the amount of the extension from the center pixel to the outermost pixel, is determined by its patch size. A more detailed categorization might come from a smaller patch size, but the accuracy of the classification might suffer from insufficient contextual data. Using a larger patch size gathers more contextual details, enhancing the reliability of the classification. However, this also raises computational complexity, potentially resulting in longer training durations or increased memory requirements. Therefore, it is crucial to choose an appropriate patch size. To validate the effectiveness of different patch size values used in this study, a series of experiments were carried out on three datasets with varying patch size values, as presented in Table 7. The results shows that a patch size of 11 consistently leads to the fastest training and testing times; however, the other three metrics do not achieve optimal values. As the value of patch size increases, the training time and testing time both increase, while the values of the remaining three metrics first increase and then decrease. There are two main reasons for this: first, as the patch size grows, the spatial information may increase, but the “purity” of the spectral information

could diminish. This happens because a larger patch might include spectral features from various categories, causing the spectral characteristics of the central pixel to be mixed with those of the surrounding pixels. Second, increasing the patch size also raises the dimensionality of the input data, which can complicate the model and increase the risk of overfitting. Therefore, the decision to set the patch size at 15 in this study is based on experimental evidence.

**Table 7.** The impact of different patch sizes on three datasets. (The optimal result is highlighted in bold, with comparisons made by first examining the mean and then the standard deviation).

Patch Size	IP				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
11 $\times$ 11	98.66 $\pm$ 0.36	98.24 $\pm$ 0.23	98.45 $\pm$ 0.24	75.62 $\pm$ 2.69	1.68 $\pm$ 0.33
13 $\times$ 13	<b>98.75 <math>\pm</math> 0.15</b>	98.23 $\pm$ 0.16	<b>98.47 <math>\pm</math> 0.23</b>	77.44 $\pm$ 1.94	1.74 $\pm$ 0.08
15 $\times$ 15	98.68 $\pm$ 0.23	98.45 $\pm$ 0.32	98.47 $\pm$ 0.27	78.66 $\pm$ 0.15	1.87 $\pm$ 0.01
17 $\times$ 17	98.65 $\pm$ 0.31	<b>98.48 <math>\pm</math> 0.53</b>	98.42 $\pm$ 0.21	79.07 $\pm$ 0.57	2.02 $\pm$ 0.14
19 $\times$ 19	98.12 $\pm$ 0.90	97.62 $\pm$ 0.66	98.37 $\pm$ 0.36	80.01 $\pm$ 0.17	2.25 $\pm$ 0.09
Patch Size	PU				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
11 $\times$ 11	99.52 $\pm$ 0.30	99.41 $\pm$ 0.30	99.37 $\pm$ 0.22	<b>303.12 <math>\pm</math> 3.21</b>	<b>8.05 <math>\pm</math> 0.16</b>
13 $\times$ 13	99.44 $\pm$ 0.19	99.44 $\pm$ 0.25	99.43 $\pm$ 0.18	306.33 $\pm$ 3.50	8.47 $\pm$ 0.39
15 $\times$ 15	<b>99.83 <math>\pm</math> 0.04</b>	<b>99.58 <math>\pm</math> 0.08</b>	<b>99.77 <math>\pm</math> 0.05</b>	310.72 $\pm$ 0.52	8.17 $\pm$ 0.06
17 $\times$ 17	99.47 $\pm$ 0.26	99.58 $\pm$ 0.15	99.46 $\pm$ 0.25	329.81 $\pm$ 9.97	8.87 $\pm$ 0.48
19 $\times$ 19	99.56 $\pm$ 0.15	99.54 $\pm$ 0.13	99.37 $\pm$ 0.16	365.12 $\pm$ 4.28	9.60 $\pm$ 0.15
Patch Size	WHLK				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
11 $\times$ 11	99.56 $\pm$ 0.17	99.66 $\pm$ 0.16	99.37 $\pm$ 0.39	<b>1315.36 <math>\pm</math> 57.84</b>	<b>31.45 <math>\pm</math> 1.65</b>
13 $\times$ 13	99.66 $\pm$ 0.05	99.52 $\pm$ 0.21	99.47 $\pm$ 0.38	1411.16 $\pm$ 60.54	34.74 $\pm$ 1.81
15 $\times$ 15	<b>99.88 <math>\pm</math> 0.02</b>	99.70 $\pm$ 0.05	<b>99.85 <math>\pm</math> 0.03</b>	1452.56 $\pm$ 51.53	37.05 $\pm$ 2.22
17 $\times$ 17	99.79 $\pm$ 0.15	<b>99.72 <math>\pm</math> 0.22</b>	99.60 $\pm$ 0.28	1609.76 $\pm$ 110.09	46.23 $\pm$ 2.55
19 $\times$ 19	99.74 $\pm$ 0.14	99.69 $\pm$ 0.16	99.51 $\pm$ 0.23	1706.76 $\pm$ 105.00	57.43 $\pm$ 4.44

### 5.3. Discussion About Loss Function

The choice of loss function directly affects the optimization process and final performance of the model. Different tasks and data distributions may require different loss functions. In order to verify the effectiveness of the hybrid loss function proposed in this paper, a series of experiments are conducted and the results are shown in Table 8. We use the regular cross-entropy loss function (*CE*) as a comparison in order to mimic the scenario without applying the hybrid loss function (*NCE + NGCE*) described in this research, i.e., to conduct loss function ablation experiments. On three datasets, we conducted four sets of experiments: *CE*, *NCE*, *NGCE* and *NCE + NGCE*. The experimental findings indicate that among the four different loss functions, the hybrid loss function introduced in this paper yields the best performance. Notably, on the PU dataset, the *NCE + NGCE* hybrid loss function stands out as the most effective, achieving optimal results in the three key evaluation metrics: Overall Accuracy (OA), Average Accuracy (AA), and Kappa. Due to the inclusion of two loss functions in our proposed hybrid loss function, both training and testing times will be extended. However, when applied to the WHLK dataset, these times are shorter compared to other combinations of loss functions. This may be attributed to the extensive data and experimental samples available in the WHLK dataset. The *NCE + NGCE* hybrid loss function yields the best performance across the three primary evaluation metrics: OA, AA and Kappa. *NCE + NGCE* hybrid function's advantage is more obvious on the data with large samples. Multiple sets of experiments on three datasets demonstrate the effectiveness of the hybrid loss function proposed in this paper.



**Table 8.** Ablation Experiment of Loss Function. (The optimal result is highlighted in bold, with comparisons made by first examining the mean and then the standard deviation).

Loss Function	IP				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
CE	98.60 $\pm$ 0.44	<b>98.81 <math>\pm</math> 0.55</b>	98.40 $\pm$ 0.51	<b>70.60 <math>\pm</math> 0.47</b>	1.68 $\pm$ 0.01
NCE	98.42 $\pm$ 0.90	98.77 $\pm$ 0.51	98.23 $\pm$ 1.01	71.06 $\pm$ 0.84	1.67 $\pm$ 0.02
NGCE	97.46 $\pm$ 0.39	94.47 $\pm$ 3.62	97.10 $\pm$ 0.44	70.62 $\pm$ 0.19	<b>1.66 <math>\pm</math> 0.01</b>
NCE + NGCE	<b>98.68 <math>\pm</math> 0.23</b>	98.45 $\pm$ 0.32	<b>98.47 <math>\pm</math> 0.27</b>	78.66 $\pm$ 0.15	1.87 $\pm$ 0.01
Loss Function	PU				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
CE	99.82 $\pm$ 0.07	99.54 $\pm$ 0.08	99.76 $\pm$ 0.09	276.76 $\pm$ 3.10	7.18 $\pm$ 0.08
NCE	99.79 $\pm$ 0.02	99.48 $\pm$ 0.09	99.72 $\pm$ 0.02	<b>272.96 <math>\pm</math> 3.22</b>	<b>7.14 <math>\pm</math> 0.12</b>
NGCE	99.20 $\pm$ 1.40	97.42 $\pm$ 4.97	98.94 $\pm$ 1.86	275.86 $\pm$ 1.03	7.33 $\pm$ 0.08
NCE + NGCE	<b>99.83 <math>\pm</math> 0.04</b>	<b>99.58 <math>\pm</math> 0.08</b>	<b>99.77 <math>\pm</math> 0.05</b>	310.72 $\pm$ 0.52	8.17 $\pm$ 0.06
Loss Function	WHLK				
	OA (%)	AA (%)	Kappa ( $\times 100$ )	Training Time (s)	Test Time (s)
CE	99.85 $\pm$ 0.01	99.66 $\pm$ 0.03	<b>99.87 <math>\pm</math> 0.01</b>	1493.53 $\pm$ 60.12	38.92 $\pm$ 1.55
NCE	99.76 $\pm$ 0.29	99.58 $\pm$ 0.13	99.68 $\pm$ 0.38	1477.56 $\pm$ 48.55	38.13 $\pm$ 1.31
NGCE	99.86 $\pm$ 0.01	<b>99.74 <math>\pm</math> 0.07</b>	99.81 $\pm$ 0.02	1444.56 $\pm$ 45.33	37.45 $\pm$ 1.44
NCE + NGCE	<b>99.88 <math>\pm</math> 0.02</b>	99.70 $\pm$ 0.05	99.85 $\pm$ 0.03	<b>1452.56 <math>\pm</math> 51.53</b>	<b>37.05 <math>\pm</math> 2.22</b>

#### 5.4. Training Time and Test Time

Training time is the stage when the model learns on the training dataset, which directly affects the model's fitting ability and complexity, while testing time is the stage when the model is deployed to make predictions on new data, which affects the model's real-time performance and application scenarios. The balance between training time and testing time is an important consideration in designing an efficient model, which is directly related to the usability and practicality of the model. In order to verify the effectiveness of the model proposed in this paper, we conducted a series of experiments, the results of which are shown in Table 9. In terms of training time, the SSFAN model performs best on all datasets. Specifically, on the IP dataset, the training time of SSFAN is 78.66 s, showing a significant advantage over other models. For example, compared to 2D CNN (87.51 s), SSFAN's training time is reduced by 8.85 s, which indicates that SSFAN is more efficiently trained when dealing with this dataset. On the dataset, the training time of SSFAN is 310.72 s, which is slightly higher than that of 2D CNN (114.63 s), but still shows a large time saving compared to SSRN (704.40 s) and 3D CNN (688.85 s). On the WHLK dataset, the training time of SSFAN is 1452.56 s, which is not as long as SpectralFormer (1567.77 s) and Hybrid CNN (2145.48 s), but is significantly lower than SSRN (5689.40 s) and 3D CNN (2490.97 s), showing its higher training efficiency. SSFAN also shows excellent performance in terms of testing time. On the IP dataset, the test time of SSFAN is 1.87 s, which is significantly lower than that of 2D CNN (3.32 s) and all other models. This shows that SSFAN not only performs well in the training phase, but also provides fast response time in the testing phase. SSFAN took 8.17 s to test on the PU dataset, while 2D CNN and SpectralFormer took 4.82 and 19.54 s, respectively. Although SSFAN's test time on this dataset is not the shortest, its overall performance is still optimal. On the WHLK dataset, the test time of SSFAN is 37.05 s, which shows a significant advantage over 2D CNN (78.56 s) and Hybrid CNN (134.45 s), and the gap is more obvious when compared to SSRN (267.67 s) and 3D CNN (222.56 s). These results indicate that the SSFAN model can provide higher computational efficiency when dealing with complex datasets, which is especially important for large-scale data processing in practical applications. Its shorter training and testing time not only improves the practical efficiency of the experiment, but also reduces the consumption of resources, providing a reliable solution for efficient data processing.

**Table 9.** The duration of training and evaluation for various models across three distinct datasets. (The optimal result is highlighted in bold, with comparisons made by first examining the mean and then the standard deviation).

Compare Methods	IP		PU		WHLK	
	Training Times (s)	Test Time (s)	Training Times (s)	Test Time (s)	Training Times (s)	Test Time (s)
2D CNN	87.51 ± 1.33	3.32 ± 0.04	<b>114.63 ± 4.33</b>	<b>4.82 ± 0.16</b>	<b>560.50 ± 58.74</b>	78.56 ± 2.45
3D CNN	435.65 ± 10.34	6.90 ± 0.34	688.85 ± 12.33	14.62 ± 0.16	2490.97 ± 30.74	222.56 ± 8.34
SSRN	704.40 ± 11.45	8.83 ± 0.41	876.67 ± 15.17	24.23 ± 1.67	5689.40 ± 108.67	267.67 ± 12.56
Hybrid CNN	396.56 ± 7.28	7.93 ± 0.53	559.23 ± 12.33	28.40 ± 2.17	2145.48 ± 73.43	134.45 ± 5.59
SpectralFormer	266.71 ± 3.28	6.44 ± 1.33	579.13 ± 14.23	19.54 ± 1.78	1567.77 ± 64.54	180.34 ± 4.65
SSFTT	137.40 ± 0.15	5.29 ± 0.15	367.66 ± 1.67	9.85 ± 0.16	1496.65 ± 34.74	67.56 ± 4.47
SSFAN(Ours)	<b>78.66 ± 0.15</b>	<b>1.87 ± 0.01</b>	310.72 ± 0.52	8.17 ± 0.06	1452.56 ± 51.53	<b>37.05 ± 2.22</b>

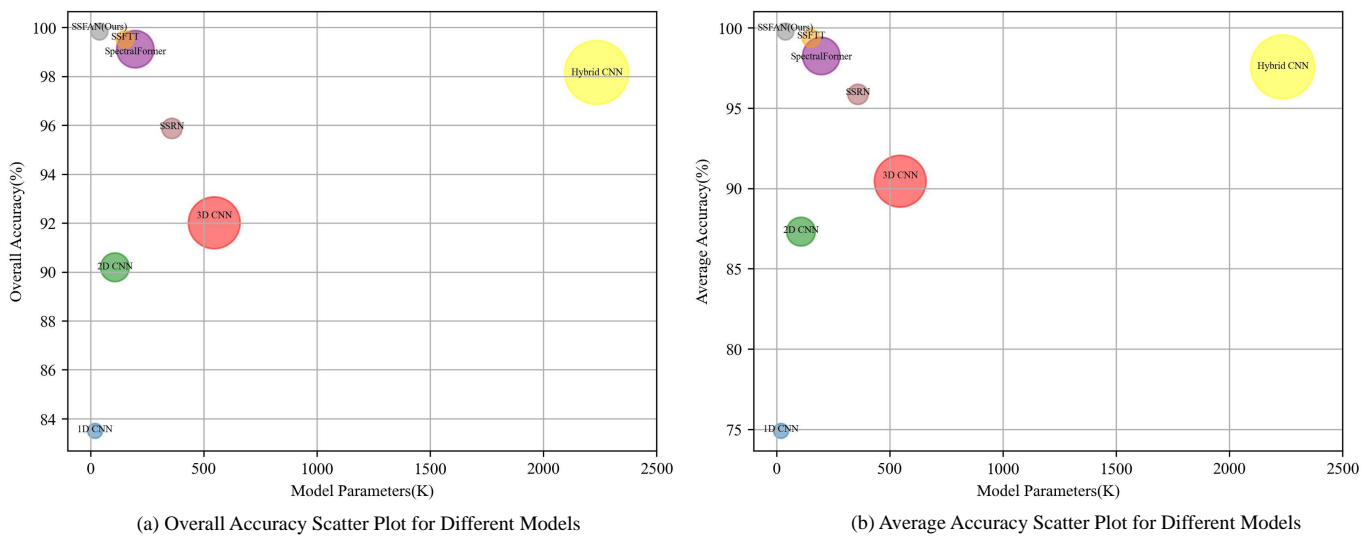
### 5.5. Discussion About Parameters and MACs

Parameters and MACs are two key metrics when discussing the efficiency of deep learning models. The number of parameters influences the model size and the memory needed for training, whereas MACs indicate the computational complexity during the inference process. To demonstrate the effectiveness of the models presented in this paper, we compare the parameter counts and MACs across different models, with the experimental results summarized in Table 10. In terms of the number of parameters, the SSFAN model has the least 39.86K parameters, showing a significant advantage over other models. For example, the 1D CNN, although also smaller in the number of parameters (20.43 K), is far inferior to SSFAN in terms of performance metrics. In terms of computational complexities (MACs), SSFAN also exhibits lower computational complexity (10.35 M), which shows significant computational savings compared to 2D CNN (30.45 M) and 3D CNN (98.67 M). This shows that SSFAN has a strong advantage in terms of computational efficiency and is able to achieve efficient performance with reduced consumption of computational resources. The SSFAN model surpasses all other models evaluated on the PU dataset. Its benefits in terms of parameter count and computational complexity enable it to deliver effective classification results even with restricted resources. Regarding accuracy, SSFAN achieves the best results in OA, AA and Kappa, demonstrating its overall excellence in classification tasks. This enhancement in performance is due not only to the optimization of the model's architecture but also to its effectiveness in feature extraction and data handling. Figure 11 demonstrates the scatter plot of SSFAN model on PU dataset, where the size of the circle represents the size of the MACs value, from which it can be seen that the SSFAN model is able to maintain the highest OA and AA values with fewer number of parameters and MACs values.

**Table 10.** Parameters, MACs, and Classification Performance for Various Models on the PU Dataset. (The optimal result is highlighted in bold, with comparisons made by first examining the mean and then the standard deviation).

Evaluation Indicators	Parameters	MACs	OA (%)	AA (%)	Kappa (×100)
1D CNN	<b>20.43 K</b>	<b>8.14 M</b>	83.50 ± 0.13	74.90 ± 0.24	77.90 ± 0.02
2D CNN	108.45 K	30.45 M	90.19 ± 0.56	87.31 ± 0.21	87.52 ± 0.12
3D CNN	546.78 K	98.67 M	92.01 ± 0.16	90.45 ± 0.15	90.87 ± 0.02
SSRN	360.01 K	15.09 M	95.87 ± 0.02	95.86 ± 0.23	95.78 ± 0.17
Hybrid CNN	2234.98 K	150.56 M	98.16 ± 0.12	97.35 ± 0.34	97.57 ± 0.16
SpectralFormer	198.12 K	51.43 M	99.11 ± 0.14	98.35 ± 0.33	98.25 ± 0.15
SSFTT	153.22 K	11.43 M	99.51 ± 0.09	98.95 ± 0.14	99.35 ± 0.12
SSFAN (Ours)	39.86 K	10.35 M	<b>99.83 ± 0.04</b>	<b>99.58 ± 0.08</b>	<b>99.77 ± 0.03</b>

As shown in Figure 11, the scatter plots of different models on the Pavia University dataset are demonstrated, where the size of the scatter represents the size of the MACs values, from which it can be seen that the SSFAN model is able to maintain the highest OA and AA values with fewer number of parameters and MACs values, which illustrates that this paper's proposed effectiveness of the model.



**Figure 11.** The scatterplot illustrates the relationship between the quantity and performance of various model parameters. (The size of each scatter point corresponds to the value of the MACs, with smaller MAC values indicating superior performance).

### 5.6. Discussion of Model Robustness

To assess the robustness [71] of the SSFAN model proposed in this study, we introduced various levels of Gaussian, Salt-and-Pepper, and Poisson noise into the dataset. Gaussian noise consists of random noise with amplitudes that follow a normal distribution, Poisson noise represents discrete events that occur randomly, while Salt-and-Pepper noise causes abrupt shifts in pixel values to extreme levels. By adjusting the parameters for each noise type, we progressively increased noise intensity to evaluate model performance across these different noise levels. The experimental results are summarized in the Table 11.

For Gaussian noise, the mean was set to zero, with standard deviation (std) values tested at 1, 5, and 10. As the standard deviation increased, classification accuracy metrics showed slight improvements. At an std of 1, the OA was 99.78%, AA reached 99.62%, and the Kappa was 99.79%. When the std rose to 10, these metrics increased slightly, with OA reaching 99.86%, AA at 99.67%, and Kappa at 99.81%. This indicates that moderate Gaussian noise does not hinder model robustness; instead, it may enhance the model's generalization ability. For salt-and-pepper noise, the noise level was controlled by setting the parameters salt\_prob and pepper\_prob, both tested at levels of 0.1, 0.3, and 0.5. As noise levels increased, classification accuracy metrics gradually declined. For instance, with salt\_prob and pepper\_prob at 0.1, OA reached 99.23%, AA was 98.32%, and Kappa stood at 98.98%. When these parameters were raised to 0.5, OA fell to 95.01%, AA dropped to 92.05%, and Kappa decreased to 93.34%. These results suggest that high levels of salt-and-pepper noise significantly impact model accuracy, likely due to the noise's disruptive effect on image pixels. For Poisson noise, intensity was controlled by adjusting the scale parameter, tested at values of 1, 10, and 20. It was observed that classification accuracy did not decrease significantly as the scale increased. At a scale of 1, OA was 99.83%, AA reached 99.63%, and Kappa was 99.78%. With the scale at 10, OA rose slightly to 99.87%, AA reached 99.71%, and Kappa was 99.83%. At a scale of 20, OA showed a slight decrease to 99.73%. This indicates that Poisson noise has a minimal effect on model performance and may even positively contribute to model generalization. In summary, the results demonstrate that moderate levels of Gaussian and Poisson noise can enhance model generalization and robustness. In contrast, high levels of salt-and-pepper noise significantly reduce model performance, suggesting a more destructive impact on the data.

**Table 11.** Effect of Different Noise Types on Classification Performance Metrics for the PU Dataset.

Different Noise			Evaluation Indicators		
			OA (%)	AA (%)	Kappa ( $\times 100$ )
Gaussian Noise (mean, std)	mean: 0	std: 1	99.78 $\pm$ 0.13	99.62 $\pm$ 0.10	99.79 $\pm$ 0.03
	mean: 0	std: 5	99.85 $\pm$ 0.04	99.66 $\pm$ 0.07	99.80 $\pm$ 0.05
	mean: 0	std: 10	99.86 $\pm$ 0.03	99.67 $\pm$ 0.03	99.81 $\pm$ 0.04
Salt and Pepper Noise (salt_prob, pepper_prob)	salt_prob: 0.1	pepper_prob: 0.1	99.23 $\pm$ 0.28	98.32 $\pm$ 0.47	98.98 $\pm$ 0.37
	salt_prob: 0.3	pepper_prob: 0.3	97.75 $\pm$ 0.82	96.86 $\pm$ 0.65	97.55 $\pm$ 0.20
	salt_prob: 0.5	pepper_prob: 0.5	95.01 $\pm$ 1.81	92.05 $\pm$ 2.16	93.34 $\pm$ 2.49
Poisson Noise (scale)	scale: 1	-	99.83 $\pm$ 0.05	99.63 $\pm$ 0.13	99.78 $\pm$ 0.07
	scale: 10	-	99.87 $\pm$ 0.01	99.71 $\pm$ 0.08	99.83 $\pm$ 0.01
	scale: 20	-	99.73 $\pm$ 0.12	99.48 $\pm$ 0.16	99.64 $\pm$ 0.16
No Noise	-	-	99.83 $\pm$ 0.04	99.58 $\pm$ 0.08	99.77 $\pm$ 0.05

### 5.7. Limitations and Future Perspectives

The SSFAN model innovatively integrates the PSSB, Scan Block, and SEMB modules to create a compact and lightweight architecture capable of efficiently extracting and processing spectral and spatial information. This results in excellent classification performance and robust real-time processing capability. However, there remains room for further optimization. Although the Scan module introduces an innovative spatial feature arrangement, its manually defined center scanning process may limit its ability to capture complex spatial relationships, particularly when dealing with hyperspectral data that exhibit significant spatial variability. This scanning method lacks flexibility, which may hinder the model's ability to fully utilize spatial information at each pixel. Furthermore, while the SSRB sequence processing mechanism within the SEMB module effectively integrates sequence information, its computational complexity remains high when processing long sequences, potentially increasing model inference time and computational resource demands. In data preprocessing stage, this study did not account for potential overlap between training and testing datasets caused by the use of patches.

Future research will explore more flexible and efficient spatial feature extraction methods to enhance the model's ability to capture spatial features at multiple scales. Additionally, optimizing the computational efficiency of the SSRB module will be a primary focus, particularly by introducing more lightweight sequence processing mechanisms to reduce the model's complexity and computational cost. Further studies will also investigate the effective integration of multimodal data or other complementary information to provide more contextual insights, thereby enhancing the accuracy of hyperspectral image classification. Moreover, we should explore new dataset partitioning methods to minimize overlap between training and testing datasets during data preprocessing, which is essential for reliable model evaluation.

## 6. Conclusions

In this paper, a lightweight SSFAN method based on spectral-spatial feature extraction and attention mechanism is proposed to improve the classification performance of HSI. SSFAN mainly consists of three modules, namely, PSSB, Scan Block, and SEMB, which can effectively capture and process the spectral and spatial information in HSIs. The PSSB preliminarily extracts the spectral and spatial features of the HSIs, which lays a solid foundation for subsequent processing. The PSSB initially extracts the spectral and spatial features in HSIs, laying a solid foundation for subsequent processing; the Scan Block captures both local and global spatial relationships, enhancing the ability to understand complex scenes and ensuring that the model can fully utilize the spatial context information in the images, thus improving the classification accuracy; the SEMB module, consisting of the SSRB and the MLP Block, is mainly used for further processing and fusion of the spectral and spatial features, of which the SSRB is used for further processing and fusion of the spectral and spatial features. By introducing the adaptive weight allocation mechanism SToken Module, it is able to flexibly process the time step and dimension of features, and deeply extract complex spectral and spatial features through multiple state

updates. The experimental results show that compared with other models, SSFAN can still maintain the highest classification accuracy while reducing the number of parameters and the value of MACs, which significantly accelerates the training and inference speed of the model, and improves its potential for practical applications in environments with limited computational resources.

**Author Contributions:** Conceptualization, C.Z.; Methodology, C.W., C.Z. and W.Y.; Software, C.Z. and B.L.; Validation, C.Z. and Y.Z.; Formal analysis, C.Z.; Investigation, C.Z.; Resources, C.W., C.Z. and G.W.; Data curation, C.Z. and Z.Z.; Writing—original draft, C.Z.; Writing—review & editing, C.Z.; Visualization, C.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was supported by the Chunhui Program Cooperative Research Project of Chinese Ministry of Education (HZKY20220279), Henan Provincial Science and Technology Research Project (232102211019, 222102210131), the Key Research Project Fund of Institution of Higher Education in Henan Province (23A520029), Henan Polytechnic University for the Double First-Class Project of Surveying and Mapping Disciplines (GCCYJ202413), and Japan Society for the Promotion of Science (JSPS) KAKENHI Grant (No.23K18517).

**Institutional Review Board Statement:** Not applicable

**Informed Consent Statement:** Not applicable

**Data Availability Statement:** The Indian Pines and Pavia University datasets are available at: [http://www.ehu.es/ccwintco/index.php?title=Hyperspectral\\_Remote\\_Sensing\\_Scenes](http://www.ehu.es/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes) (accessed on 5 August 2024). The WHU-Hi-LongKou datasets are available at [http://rsidea.whu.edu.cn/resource\\_WHUHi\\_sharing.htm](http://rsidea.whu.edu.cn/resource_WHUHi_sharing.htm) (accessed on 5 August 2024).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Bioucas-Dias, J.M.; Plaza, A.; Camps-Valls, G.; Scheunders, P.; Nasrabadi, N.; Chanussot, J. Hyperspectral remote sensing data analysis and future challenges. *IEEE Geosci. Remote Sens. Mag.* **2013**, *1*, 6–36. [[CrossRef](#)]
2. Noor, S.S.M.; Michael, K.; Marshall, S.; Ren, J.; Tschannerl, J.; Kao, F.J. The properties of the cornea based on hyperspectral imaging: Optical biomedical engineering perspective. In Proceedings of the IEEE 2016 International Conference on Systems, Signals and Image Processing (IWSSIP), Bratislava, Slovakia, 23–25 May 2016; pp. 1–4.
3. Wang, J.; Zhang, L.; Tong, Q.; Sun, X. The Spectral Crust project—Research on new mineral exploration technology. In Proceedings of the IEEE 2012 4th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Shanghai, China, 4–7 June 2012; pp. 1–4.
4. Dale, L.M.; Thewis, A.; Boudry, C.; Rotar, I.; Dardenne, P.; Baeten, V.; Pierna, J.A.F. Hyperspectral imaging applications in agriculture and agro-food product quality and safety control: A review. *Appl. Spectrosc. Rev.* **2013**, *48*, 142–159. [[CrossRef](#)]
5. Veraverbeke, S.; Dennison, P.; Gitas, I.; Hulley, G.; Kalashnikova, O.; Katagis, T.; Kuai, L.; Meng, R.; Roberts, D.; Stavros, N. Hyperspectral remote sensing of fire: State-of-the-art and future perspectives. *Remote Sens. Environ.* **2018**, *216*, 105–121. [[CrossRef](#)]
6. Zhong, Y.; Cao, Q.; Zhao, J.; Ma, A.; Zhao, B.; Zhang, L. Optimal decision fusion for urban land-use/land-cover classification based on adaptive differential evolution using hyperspectral and LiDAR data. *Remote Sens.* **2017**, *9*, 868. [[CrossRef](#)]
7. Ardouin, J.P.; Lévesque, J.; Rea, T.A. A demonstration of hyperspectral image exploitation for military applications. In Proceedings of the IEEE 2007 10th International Conference on Information Fusion, Quebec, QC, Canada, 9–12 July 2007; pp. 1–8.
8. Gevaert, C.M.; Suomalainen, J.; Tang, J.; Kooistra, L. Generation of spectral-temporal response surfaces by combining multispectral satellite and hyperspectral UAV imagery for precision agriculture applications. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 3140–3146. [[CrossRef](#)]
9. Sun, L.; He, C.; Zheng, Y.; Tang, S. SLRL4D: Joint Restoration of Subspace Low-Rank Learning and Non-Local 4-D Transform Filtering for Hyperspectral Image. *Remote Sens.* **2020**, *12*, 2979. [[CrossRef](#)]
10. He, C.; Sun, L.; Huang, W.; Zhang, J.; Zheng, Y.; Jeon, B. TSLRLN: Tensor subspace low-rank learning with non-local prior for hyperspectral image mixed denoising. *Signal Process.* **2021**, *184*, 108060. [[CrossRef](#)]
11. Sun, L.; Wu, F.; Zhan, T.; Liu, W.; Wang, J.; Jeon, B. Weighted nonlocal low-rank tensor decomposition method for sparse unmixing of hyperspectral images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 1174–1188. [[CrossRef](#)]
12. Yang, S.; Shi, Z. Hyperspectral image target detection improvement based on total variation. *IEEE Trans. Image Process.* **2016**, *25*, 2249–2258. [[CrossRef](#)]
13. Sun, L.; Wu, Z.; Liu, J.; Xiao, L.; Wei, Z. Supervised spectral-spatial hyperspectral image classification with weighted Markov random fields. *IEEE Trans. Geosci. Remote Sens.* **2014**, *53*, 1490–1503. [[CrossRef](#)]
14. Sun, L.; Ma, C.; Chen, Y.; Zheng, Y.; Shim, H.J.; Wu, Z.; Jeon, B. Low rank component induced spatial-spectral kernel method for hyperspectral image classification. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 3829–3842. [[CrossRef](#)]

15. Jain, V.; Phophalia, A. Exponential weighted random forest for hyperspectral image classification. In Proceedings of the IEEE IGARSS 2019—2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 28 July–2 August 2019; pp. 3297–3300.
16. Zhao, S.; Bai, Y.; Shao, S.; Liu, W.; Ge, X.; Li, Y.; Liu, B. SELM: Self-Motivated Ensemble Learning Model for Cross-Domain Few-Shot Classification in Hyperspectral Images. *IEEE Geosci. Remote Sens. Lett.* **2024**, *21*, 5503805. [[CrossRef](#)]
17. Ye, Q.; Huang, P.; Zhang, Z.; Zheng, Y.; Fu, L.; Yang, W. Multiview learning with robust double-sided twin SVM. *IEEE Trans. Cybern.* **2021**, *52*, 12745–12758. [[CrossRef](#)] [[PubMed](#)]
18. Ye, Q.; Zhao, H.; Li, Z.; Yang, X.; Gao, S.; Yin, T.; Ye, N. L1-Norm distance minimization-based fast robust twin support vector  $k$ -plane clustering. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 4494–4503. [[CrossRef](#)]
19. Zhang, J.; Liu, L.; Zhao, R.; Shi, Z. A Bayesian meta-learning-based method for few-shot hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *61*, 1–13. [[CrossRef](#)]
20. Licciardi, G.; Marpu, P.R.; Chanussot, J.; Benediktsson, J.A. Linear versus nonlinear PCA for the classification of hyperspectral data based on the extended morphological profiles. *IEEE Geosci. Remote Sens. Lett.* **2011**, *9*, 447–451. [[CrossRef](#)]
21. Prasad, S.; Bruce, L.M. Limitations of principal components analysis for hyperspectral target recognition. *IEEE Geosci. Remote Sens. Lett.* **2008**, *5*, 625–629. [[CrossRef](#)]
22. Villa, A.; Benediktsson, J.A.; Chanussot, J.; Jutten, C. Hyperspectral image classification with independent component discriminant analysis. *IEEE Trans. Geosci. Remote Sens.* **2011**, *49*, 4865–4876. [[CrossRef](#)]
23. Ye, Q.; Yang, J.; Liu, F.; Zhao, C.; Ye, N.; Yin, T. L1-norm distance linear discriminant analysis based on an effective iterative algorithm. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *28*, 114–129. [[CrossRef](#)]
24. Bandos, T.V.; Bruzzone, L.; Camps-Valls, G. Classification of hyperspectral images with regularized linear discriminant analysis. *IEEE Trans. Geosci. Remote Sens.* **2009**, *47*, 862–873. [[CrossRef](#)]
25. Fauvel, M.; Benediktsson, J.A.; Chanussot, J.; Sveinsson, J.R. Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. *IEEE Trans. Geosci. Remote Sens.* **2008**, *46*, 3804–3814. [[CrossRef](#)]
26. Benediktsson, J.A.; Palmason, J.A.; Sveinsson, J.R. Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 480–491. [[CrossRef](#)]
27. Dalla Mura, M.; Villa, A.; Benediktsson, J.A.; Chanussot, J.; Bruzzone, L. Classification of hyperspectral images by using extended morphological attribute profiles and independent component analysis. *IEEE Geosci. Remote Sens. Lett.* **2010**, *8*, 542–546. [[CrossRef](#)]
28. Xiao, F.; Xiang, H.; Cao, C.; Gao, X. Neural Architecture Search-based Few-shot Learning for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5513715. [[CrossRef](#)]
29. Wang, Z.; Zhao, S.; Zhao, G.; Song, X. Dual-Branch Domain Adaptation Few-Shot Learning for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5506116. [[CrossRef](#)]
30. Bai, J.; Wen, Z.; Xiao, Z.; Ye, F.; Zhu, Y.; Alazab, M.; Jiao, L. Hyperspectral image classification based on multibranch attention transformer networks. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–17. [[CrossRef](#)]
31. Yu, H.; Xu, Z.; Zheng, K.; Hong, D.; Yang, H.; Song, M. MSTNet: A multilevel spectral–spatial transformer network for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–13. [[CrossRef](#)]
32. Huang, L.; Chen, Y.; He, X. Spectral-spatial mamba for hyperspectral image classification. *arXiv* **2024**, arXiv:2404.18401. [[CrossRef](#)]
33. Li, Y.; Luo, Y.; Zhang, L.; Wang, Z.; Du, B. MambaHSI: Spatial-Spectral Mamba for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5524216. [[CrossRef](#)]
34. Zhao, W.; Du, S. Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 4544–4554. [[CrossRef](#)]
35. Sun, H.; Zheng, X.; Lu, X.; Wu, S. Spectral–spatial attention network for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *58*, 3232–3245. [[CrossRef](#)]
36. Huang, L.; Chen, Y. Dual-path siamese CNN for hyperspectral image classification with limited training samples. *IEEE Geosci. Remote Sens. Lett.* **2020**, *18*, 518–522. [[CrossRef](#)]
37. Xu, Q.; Xiao, Y.; Wang, D.; Luo, B. CSA-MSO3DCNN: Multiscale octave 3D CNN with channel and spatial attention for hyperspectral image classification. *Remote Sens.* **2020**, *12*, 188. [[CrossRef](#)]
38. Roy, S.K.; Krishna, G.; Dubey, S.R.; Chaudhuri, B.B. HybridSN: Exploring 3-D–2-D CNN feature hierarchy for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2019**, *17*, 277–281. [[CrossRef](#)]
39. Zhang, H.; Li, Y. Spectral-spatial classification of hyperspectral imagery based on deep convolutional network. In Proceedings of the IEEE 2016 International Conference on Orange Technologies (ICOT), Melbourne, VIC, Australia, 18–20 December 2016; pp. 44–47.
40. Hu, W.; Huang, Y.; Wei, L.; Zhang, F.; Li, H. Deep convolutional neural networks for hyperspectral image classification. *J. Sens.* **2015**, *2015*, 258619. [[CrossRef](#)]
41. Tian, C.; Zhang, Y.; Zuo, W.; Lin, C.W.; Zhang, D.; Yuan, Y. A heterogeneous group CNN for image super-resolution. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 6507–6519. [[CrossRef](#)]
42. Li, Y.; Zhang, H.; Shen, Q. Spectral–spatial classification of hyperspectral imagery with 3D convolutional neural network. *Remote Sens.* **2017**, *9*, 67. [[CrossRef](#)]

43. Chen, Y.; Jiang, H.; Li, C.; Jia, X.; Ghamisi, P. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 6232–6251. [[CrossRef](#)]
44. Zhang, H.; Li, Y.; Jiang, Y.; Wang, P.; Shen, Q.; Shen, C. Hyperspectral classification based on lightweight 3-D-CNN with transfer learning. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 5813–5828. [[CrossRef](#)]
45. Ahmad, M.; Shabbir, S.; Raza, R.A.; Mazzara, M.; Distefano, S.; Khan, A.M. Hyperspectral image classification: Artifacts of dimension reduction on hybrid CNN. *arXiv* **2021**, arXiv:2101.10532.
46. Liu, X.; Wang, H.; Liu, J.; Sun, S.; Fu, M. HSI classification based on multimodal CNN and shadow enhance by DSR spatial-spectral fusion. *Can. J. Remote Sens.* **2021**, *47*, 773–789. [[CrossRef](#)]
47. Lin, S.; Xiao-wei, W.; Long-po, Y.; Zong-fang, M. Hyperspectral Image Classification Based on the Fusion of Superpixels and Deformable Features. In Proceedings of the IEEE 2024 36th Chinese Control and Decision Conference (CCDC), Xi'an, China, 25–27 May 2024; pp. 1742–1746.
48. Zhong, Z.; Li, J.; Luo, Z.; Chapman, M. Spectral-spatial residual network for hyperspectral image classification: A 3-D deep learning framework. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 847–858. [[CrossRef](#)]
49. Yue, G.; Zhang, L.; Zhou, Y.; Wang, Y.; Xue, Z. S2TNet: Spectral-Spatial Triplet Network for Few-Shot Hyperspectral Image Classification. *IEEE Geosci. Remote Sens. Lett.* **2024**, *21*, 5501705. [[CrossRef](#)]
50. Yu, C.; Han, R.; Song, M.; Liu, C.; Chang, C.I. Feedback attention-based dense CNN for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 1–16. [[CrossRef](#)]
51. Hong, D.; Gao, L.; Yao, J.; Zhang, B.; Plaza, A.; Chanussot, J. Graph convolutional networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 5966–5978. [[CrossRef](#)]
52. Qin, A.; Shang, Z.; Tian, J.; Wang, Y.; Zhang, T.; Tang, Y.Y. Spectral-spatial graph convolutional networks for semisupervised hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 241–245. [[CrossRef](#)]
53. Wan, S.; Gong, C.; Zhong, P.; Pan, S.; Li, G.; Yang, J. Hyperspectral image classification with context-aware dynamic graph convolutional network. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 597–612. [[CrossRef](#)]
54. Yang, Y.; Tang, X.; Zhang, X.; Ma, J.; Liu, F.; Jia, X.; Jiao, L. Semi-supervised multiscale dynamic graph convolution network for hyperspectral image classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 6806–6820. [[CrossRef](#)]
55. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
56. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
57. He, J.; Zhao, L.; Yang, H.; Zhang, M.; Li, W. HSI-BERT: Hyperspectral image classification using the bidirectional encoder representation from transformers. *IEEE Trans. Geosci. Remote Sens.* **2019**, *58*, 165–178. [[CrossRef](#)]
58. Hong, D.; Han, Z.; Yao, J.; Gao, L.; Zhang, B.; Plaza, A.; Chanussot, J. SpectralFormer: Rethinking hyperspectral image classification with transformers. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 1–15. [[CrossRef](#)]
59. Tang, P.; Zhang, M.; Liu, Z.; Song, R. Double attention transformer for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2023**, *20*, 1–5. [[CrossRef](#)]
60. Sun, L.; Zhao, G.; Zheng, Y.; Wu, Z. Spectral-spatial feature tokenization transformer for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–14. [[CrossRef](#)]
61. Roy, S.K.; Deria, A.; Shah, C.; Haut, J.M.; Du, Q.; Plaza, A. Spectral-spatial morphological attention transformer for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–15. [[CrossRef](#)]
62. Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y. A survey on vision transformer. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 87–110. [[CrossRef](#)]
63. Zhuang, B.; Liu, J.; Pan, Z.; He, H.; Weng, Y.; Shen, C. A survey on efficient training of transformers. *arXiv* **2023**, arXiv:2302.01107.
64. LeCun, Y.; Bottou, L.; Orr, G.B.; Müller, K.R., Efficient backprop. In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 9–50.
65. Zhong, Y.; Hu, X.; Luo, C.; Wang, X.; Zhao, J.; Zhang, L. WHU-Hi: UAV-borne hyperspectral with high spatial resolution (H2) benchmark datasets and classifier for precise crop identification based on deep convolutional neural network with CRF. *Remote Sens. Environ.* **2020**, *250*, 112012. [[CrossRef](#)]
66. Bamber, D.; Van Santen, J.P. How many parameters can a model have and still be testable? *J. Math. Psychol.* **1985**, *29*, 443–473. [[CrossRef](#)]
67. Kumar, M.S.; Kumar, D.A.; Samundiswary, P. Design and performance analysis of Multiply-Accumulate (MAC) unit. In Proceedings of the IEEE 2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014], Nagercoil, India, 20–21 March 2014; pp. 1084–1089.
68. Schober, P.; Najafi, M.H.; TaheriNejad, N. High-accuracy multiply-accumulate (MAC) technique for unary stochastic computing. *IEEE Trans. Comput.* **2021**, *71*, 1425–1439. [[CrossRef](#)]
69. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.

- 
70. Rogozhnikov, A. Einops: Clear and reliable tensor manipulations with einstein-like notation. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021.
  71. Taori, R.; Dave, A.; Shankar, V.; Carlini, N.; Recht, B.; Schmidt, L. Measuring robustness to natural distribution shifts in image classification. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18583–18599.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.