remote sensing

MDPI

Article

Spatial-Spectral Transformer for Hyperspectral Image Classification

Xin He [1], Yushi Chen [1,*] and Zhouhan Lin [2]

[1] School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China; 19b905007@stu.hit.edu.cn
[2] Shanghai Jiao Tong University, Shanghai 201100, China; lin.zhouhan@gmail.com
* Correspondence: chenyushi@hit.edu.cn

Abstract: Recently, a great many deep convolutional neural network (CNN)-based methods have been proposed for hyperspectral image (HSI) classification. Although the proposed CNN-based methods have the advantages of spatial feature extraction, they are difficult to handle the sequential data with and CNNs are not good at modeling the long-range dependencies. However, the spectra of HSI are a kind of sequential data, and HSI usually contains hundreds of bands. Therefore, it is difficult for CNNs to handle HSI processing well. On the other hand, the Transformer model, which is based on an attention mechanism, has proved its advantages in processing sequential data. To address the issue of capturing relationships of sequential spectra in HSI in a long distance, in this study, Transformer is investigated for HSI classification. Specifically, in this study, a new classification framework titled spatial-spectral Transformer (SST) is proposed for HSI classification. In the proposed SST, a well-designed CNN is used to extract the spatial features, and a modified Transformer (a Transformer with dense connection, i.e., DenseTransformer) is proposed to capture sequential spectra relationships, and multilayer perceptron is used to finish the final classification task. Furthermore, dynamic feature augmentation, which aims to alleviate the overfitting problem and therefore generalize the model well, is proposed and added to the SST (SST-FA). In addition, to address the issue of limited training samples in HSI classification, transfer learning is combined with SST, and another classification framework titled transferring-SST (T-SST) is proposed. At last, to mitigate the overfitting problem and improve the classification accuracy, label smoothing is introduced for the T-SST-based classification framework (T-SST-L). The proposed SST, SST-FA, T-SST, and T-SST-L are tested on three widely used hyperspectral datasets. The obtained results reveal that the proposed models provide competitive results compared to the state-of-the-art methods, which shows that the concept of Transformer opens a new window for HSI classification.

Keywords: classification; convolutional neural network (CNN); hyperspectral image (HSI); Transformer

Citation: He, X.; Chen, Y.; Lin, Z. Spatial-Spectral Transformer for Hyperspectral Image Classification. *Remote Sens.* **2021**, *13*, 498. https://doi.org/10.3390/rs13030498

Academic Editor: Danfeng Hong
Received: 3 December 2020
Accepted: 27 January 2021
Published: 30 January 2021

1. Introduction

Due to the advances in imaging spectrometry, hyperspectral sensors tend to capture the intensity of reflectance of a given scene with increasingly higher spatial and spectral resolution [1]. The obtained hyperspectral image (HSI) contains both spatial features and a continuous diagnostic spectrum of different objects at the same time [2]. Thus, the obtained abundant information makes HSI useful in many areas including effective measurement of agricultural performance [3], plant diseases detection [4], identification of minerals [5], disease diagnosis and image-guided surgery [6], ecosystem measurement [7], and earth monitoring [8]. To fully use the obtained HSI, many data processing techniques have been explored, such as unmixing, detection, and classification [8].

HSI classification aims to categorize the content of each pixel in the scene [9], which is a basic procedure in applications such as identifying the type of land-cover classes in earth monitoring [10].

A great many supervised methods have been proposed for HSI classification in the last two decades [11]. In the early stage of HSI classification, the HSI classification

methods used spectral information only. A typical spectral classifier was introduced in [12], which was based on the support vector machine (SVM). SVM shows its low sensitivity to high dimensionality [13]; therefore, many SVM-based classifiers have been proposed to handle the spectral classification of HSI [14]. Hyperspectral sensors can provide abundant spatial information of the observing scene with the development of imaging technology. It is reasonable to develop spectral-spatial classifiers. Numerous morphological operations have been developed to extract the spatial features of HSI for following spatial-spectral classification, such as morphological profiles (MPs) [15], extended MPs (EMPs) [16], extended multi-attribute profile (EMAP) [17], and extinction profiles (EPs) [18]. However, the aforementioned HSI classifiers are not deep models [11].

In recent years, deep learning techniques, especially the deep convolutional neural network (CNN), have revolutionized the means of remote sensing data processing. The task of HSI classification is not an exception. In [19], stacked auto-encoder was introduced as a deep model for HSI feature extraction and classification. After that, several deep learning models such as the deep belief network [20], CNN [21,22], recurrent neural network [23,24], generative adversarial network [25,26], and capsule network [27,28] were investigated for HSI classification and obtained good classification performance.

Because of its local connection and shared weights, which makes it effective to capture local correlations, CNN is quite useful for image processing, including HSI classification. According to the input information of models, CNN-based HSI classification methods can be divided into three types: spectral CNN, spatial CNN, and spectral-spatial CNN. Spectral CNN-based HSI classification receives the pixel vector as input and uses CNN to classify the HSI only in the spectral domain. For example, Hu et al. proposed 1-D CNN with five convolutional layers to extract the spectral features of HSI [29]. Moreover, an interesting work was proposed in [30], which used CNN to extract pixel-pair features for HSI classification and obtained good classification performance.

Spatial CNN-based methods are the second type of CNN-based HSI classification methods. In addition to spectral information, the obtained HSI contains abundant spatial information; therefore, it is reasonable to use spatial CNN (2-D CNN) to extract the spatial features of HSI. Most of existing spatial CNN-based HSI classification methods were conducted on one or several principal components. For example, in [31], the cropped spatial patches of pixel-centered neighbors, which belong to the first principal component of his, were used to train a 2-D CNN for HSI classification.

Spectral-spatial CNN-based methods are the third type of CNN-based HSI classification methods, which aim for joint exploitation of spectral and spatial HSI features in a unified framework. Since the input of HSI is a cubic tensor, 3-D convolution was used for HSI classification [32]. For example, in [33], He et al. proposed a 3D deep CNN to jointly extracted spatial and spectral features by computing multiscale features. In [34], the 3-D convolutional layer and batch normalization layer were utilized to extract spectral-spatial information and regularize the model, respectively. Due to the good classification performance obtained by CNN-based methods, CNN has become the de-facto standard for HSI classification in recent years.

Existing CNN models for HSI classification have achieved state-of-the-art performance; however, there are still several limitations. First, some information of input HSI is ignored and is not well explored in CNN-based methods. CNN is a vector-based method, which considers the inputs to be a collection of pixel vectors [35]. For HSI, it intrinsically has a sequence-based data structure in the spectral domain. Therefore, using CNN can lead to information loss when dealing with hyperspectral pixel vectors [36]. Second, learning long-range sequential dependence back and forth between distant positions of bands is difficult. Since convolutional operations process a local neighborhood, the receptive field of CNN is strictly restricted by its kernel size and the number of layers, which has made it less advantageous in capturing long-range dependencies of input data [37]. Therefore, it is difficult to learn the long-range dependencies of HSI, which usually contain hundreds of spectral bands.

Very recently, a model called Transformer [38], which is based on the self-attention mechanism [39], has been proposed for natural language processing. Transformer uses attention to draw global dependency within a sequence of input. For deep learning models including Transformer, there is a common problem of vanishing-gradient, which hampers the convergence in the training procedure [40]. To alleviate the vanishing-gradient problem, a new type of Transformer, which uses dense connection to strengthen feature propagation, titled DenseTransformer, is proposed in this study.

Furthermore, two classification frameworks based on DenseTransformer are proposed for HSI classification. The first classification framework combines CNN, DenseTransformer, and multilayer perceptron. In the second classification framework, transfer learning strategy is combined with Transformer to improve the HSI classification performance with limited training samples.

The main contributions of this study are summarized as follows.

(1) A modified Transformer titled DenseTransformer is proposed, which uses dense connection to alleviate the vanishing-gradient problem in Transformer.

(2) A new classification framework, i.e., spatial-spectral Transformer (SST), is proposed for HSI classification, which combines CNN, DenseTransformer, and multilayer perceptron (MLP). In the proposed SST, a well-designed CNN is used to extract the spatial features of HSI, and the proposed DenseTransformer is used to capture sequential spectra relationships of HSI, and the MLP is used to finish the classification task.

(3) Furthermore, dynamic feature augmentation, which aims to alleviate the overfitting problem and therefore generalize the model well, is proposed and added to the SST to form a new HSI classification method (i.e., SST-FA).

(4) Another new classification framework, i.e., transferring spatial-spectral Transformer (T-SST), is proposed to further improve the performance of HSI classification. The proposed T-SST uses the pre-trained VGG-like model on a large dataset as the initialization of the used CNN in SST; therefore, it enhanced the HSI classification accuracy with limited training samples.

(5) At last, label smoothing is introduced into Transformer-based classification. Label smoothing is combined with T-SST to formulate a new HSI classification method titled T-SST-L.

The rest of this paper is organized as follows. The proposed SST and transferring SST for HSI classification are presented in Sections 2 and 3, respectively. The experimental results and discussions are reported in Section 4. Section 5 presents the conclusion of this study.

2. Spatial-Spectral Transformer for Hyperspectral Image Classification

The framework of the proposed SST for HSI classification is shown in Figure 1. In general, there are three parts in the classification method: CNN-based spatial feature extraction, modified Transformer-based spatial-spectral feature extraction, and MLP-based classification.

Firstly, for each band of HSI, a 2D patch, which contains the neighboring pixels of the pixel to be classified, is selected as input. There are n (i.e., the number of bands of HSI) patches for a training sample. After that, a well-designed CNN is used to extract the features of each 2D patch, and then the extracted features are sent to Transformer. Then, the modified Transformer is used to obtain the relationship of the sequential spatial features. At last, the obtained spatial-spectral features are used to get the classification result.

2.1. CNN-Based HSI Spatial Feature Extraction

CNN has powerful capability to extract spatial features of image, and it is widely used for image processing such as classification, detection, and segmentation. For HSI, it contains abundant spatial information. CNN is used in this study to effectively extract the spatial features of HSI.

CNN contains a wide range of different architectures. How to choose a proper architecture is important. Although HSI is a 3-D cube, a 3-D CNN is not used in this study. Instead, a 2-D CNN is used in this classification framework. Furthermore, we use 2-D CNN

separately to extract the features of each band in his, and the extracted features are fed into a Transformer.

VGGNet is a simple but effective model, which considers the depth of appropriate layers and does not increase the total number of parameters compared to previous AlexNet [41]. Therefore, we used VGG-like architecture. The original VGG contains 16 layers, which includes 13 convolutional layers and three fully connected layers. Each convolutional layer is followed by BN layer and ReLU operation, and the max pooling layer is added after the second, fourth, seventh, tenth, and 13th convolutional layer. Possibly, the usage of the whole 16 layers is not a good choice for HSI spatial feature extraction. How to design a proper CNN architecture is a key point for a successful HSI classifier. In the experimental part, we designed a VGG-like deep CNN for spatial feature extraction of HSI.

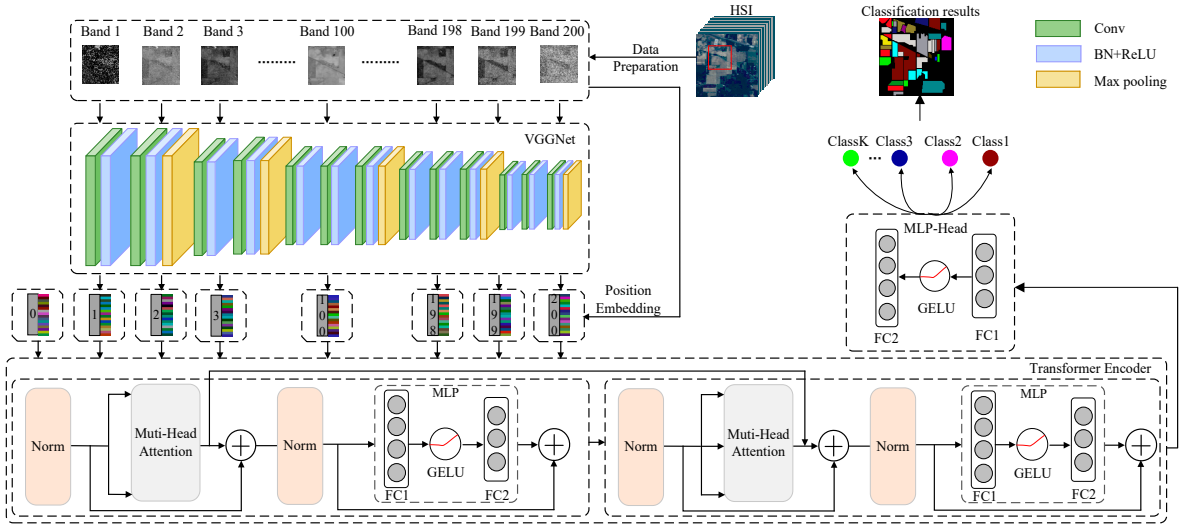

Figure 1. The framework of SST for HSI classification.

2.2. Spectral-Spatial Transformer for HSI Classification

CNN uses local connection to extract neighboring features of inputs. HSI usually contains hundreds of bands; therefore, it is difficult for CNN to obtain spectral relationships in a long distance. The self-attention mechanism can obtain the relationship of every two bands. For example, Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) contains 224 bands. Using self-attention, a matrix with the shape of 224×224 can be obtained through the learning procedure. Each element in the matrix represents the relationship between the two bands.

As shown in Figure 1, the extracted features by CNN in the last part are then sent to the Transformer to learn long-range dependencies, which mainly contains three elements.

The first element is called the position embedding, which aims to capture the positional information of the different bands. This element modifies the output features of the last part, which depends on its positions without changing these full features. In this paper, one dimensional position embedding is utilized, which considers the input features as a sequence of different bands. These generated positional embeddings are added to the features, then sent together to the next element. In addition, a learnable position embedding is prepared (i.e., number zero), whose state serves as the whole representations of the band. This learnable position embedding combines with the third part to finish the classification task.

The second element is the Transformer encoder, which is the core part of our model. The Transformer encoder contains a total of d encoder blocks, and each encoder block consists of a multi-head attention and a MLP layer, coupled with layer normalization and residual connection. In each encoder block, a normalization layer is added before each

multi-head attention and an MLP layer, and residual connections are designed after each multi-head attention and MLP layer.

Let denote the number of n bands of HSI $(b_1, b_2, \cdots b_n)$ by $B \in R^{n \times d_{model}}$, where d_{model} indicates the dimension of the extracted features by CNN. The Transformer encoder aims to capture the interaction among all n bands of HSI by encoding each band in terms of the global contextual information. Specifically, three learnable weight matrices including queries (i.e., Q), keys (i.e., K) of dimension d_k, and values (i.e., V) of dimension d_v are defined. The dot products are applied to compute the query with all keys, and then the softmax function is used to compute the weights on the values. The output of attention is defined as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

where d_k is the dimension of K.

It is beneficial to project the queries, keys, and values several times (i.e., h times) with different and learned projections, and then these results are concatenated. This process is named the multi-head attention. Each result of those parallel computations of attention is called a *head*.

$$MultiHead(Q, K, V) = concat(head_1, \cdots, head_h)W^O, \tag{2}$$

where $head_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$, $W_i^Q \in R^{d_{model} \times d_q}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$, and $W^O \in R^{h \times d_v \times d_{model}}$ are parameter matrices.

After that, the weights extracted by the multi-head attention mechanism are sent to the MLP layer, whose output features are 512 dimensions. Here, MLP is constituted by two fully connected layers with a nonlinearity named the Gaussian error linear unit (GELU) activation between. Here, GELU is the variant of the ReLU, which can be defined as follows [42]:

$$GELU = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}\left(x/\sqrt{2}\right)\right], \tag{3}$$

where $\Phi(x)$ indicates the standard Gaussian cumulative distribution function, $\text{erf(x)} = \int_0^x e^{-t^2} dt$.

Before the MLP layer, there is always a normalization layer [43], which not only reduces the training time by normalizing neurons, but also alleviates the vanishing or exploding gradient problem. For the ith summed input at the l-th layer a_i^l, the normalization layer represents as follows:

$$a_i^{-l} = \frac{g^l}{\sigma^l} \cdot \left(a_i^l - \mu^l\right) + b, \tag{4}$$

where a_i^{-l} is normalized summed input, and μ^l and σ^l represent expectation and variance at lth layer, respectively. g^l and b indicate the learned scale parameter and shift parameter, respectively.

For a deep learning model, there is a common problem titled vanishing-gradient, which hampers the convergence in the training of the deep Transformer model [40]. To alleviate the vanishing-gradient and strengthen feature propagation, short-cut connection is used to form a DenseTransformer. Specially, each layer in DenseTransformer has connections of the previous layers in the DenseTransformer. For a traditional Transformer $H_L(.)$ with L layers, there are L connections, and a DenseTransformer has $\frac{L(L+1)}{2}$ connections. The DenseTransformer encourages feature reuse and therefore mitigates the vanishing-gradient problem. Figure 2 shows the proposed DenseTransformer when $L = 3$.

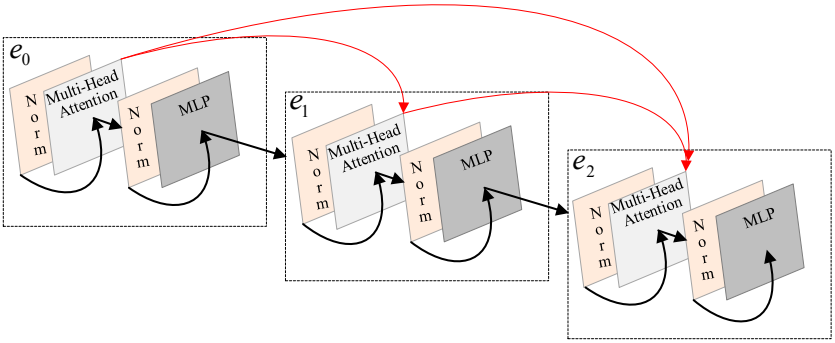

Figure 2. A DenseTransformer.

The proposed DenseTransformer consists of L layers, considering a single e_L indicates the output of the traditional Transformer $H_L(.)$ at the L-th layer. Consequently, the L-th layer of the proposed DenseTransformer receives the weights produced by the previous preceding layers $\boldsymbol{e}_0, \boldsymbol{e}_1, \cdots, \boldsymbol{e}_{L-1}$, which can be defined as follows:

$$\boldsymbol{e}_L = H_L([\boldsymbol{e}_0, \boldsymbol{e}_1, \cdots, \boldsymbol{e}_{L-1}]), \tag{5}$$

$$DenseMultiHead(Q, K, V,\ \boldsymbol{e}_L) = concat(head_1, \cdots, head_h)W^O, \tag{6}$$

where $head_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V,\ (\boldsymbol{e}_L)_i\right)$, $W_i^Q \in R^{d_{model} \times d_q}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$, and $W^O \in R^{h \times d_v \times d_{model}}$ are parameter matrices.

The third part of SST is MLP. The architecture of MLP includes two fully connected layers with a GELU operation, where the last fully connected layer (i.e., the softmax layer) aims to generate the final results for HSI classification. In softmax, for an input vector R, the probability that the input belongs to category i can be estimated as follows:

$$P(Y = i|R,\ W,\ b) = s(WR + b) = \frac{e^{W_i R + b_i}}{\sum_j e^{W_j R + b_j}}, \tag{7}$$

where W and b are weights and biases of the Softmax layer, respectively.

In the MLP, the size of the input-layer is set to be the same as the size of the output-layer of the Transformer, and the size of the output-layer is set to be the same as the total number of classes. Softmax ensures the activation of each output unit sums to 1. Therefore, the output can be deemed as a set of conditional probabilities.

2.3. Dynamic Feature Augmentation

Due to the proposed, SST is often susceptible to overfitting and therefore requires proper regularization to generalize well. In this subsection, a simple regularization technique called dynamic feature augmentation is proposed, which is implemented by randomly masking out features during training. Then, SST is combined with feature augmentation to form a new HSI classifier (i.e., SST-FA), which improves the robustness and overall classification performance of SST.

Specially, the dimension of the spatial features extracted by the VGG is high (i.e., 512-dimension), which is easy to overfit for the Transformer model. Here, a coordinate is first randomly selected in the features, then, a mask is placed around the coordinate, which decides how many features are set to zero. Note that the coordinate dynamically changes w.r.t. epochs during training, which ensures the Transformer model receives different features. The proposed SST-FA is not only easy to implement, but also able to further improve the Transformer model performance.

3. Heterogeneous Transferring Spatial-Spectral Transformer for Hyperspectral Image Classification

The collection of training samples is not only expensive but also time-consuming. Therefore, limited training samples are a common issue in HSI classification. To address this issue, transfer learning is combined with SST in this study. Transfer learning is a technique that extracts the knowledge from the source domain and transfers it to the target domain [44]. For example, in CNN-based transfer learning, the learned weights on the source domain can be used to initialize the net of the target domain. Therefore, when it is properly used, transfer learning can improve the classification performance of the target task if the number of training samples is limited.

To further improve classification performance of the proposed SST, transferring SST (T-SST) is proposed in this section. Figure 3 shows the framework of the proposed T-SST for HSI classification. In general, there are three parts in the classification method: transferring CNN-based spatial feature extraction, Transformer-based spatial-spectral feature extraction, and MLP-based classification.

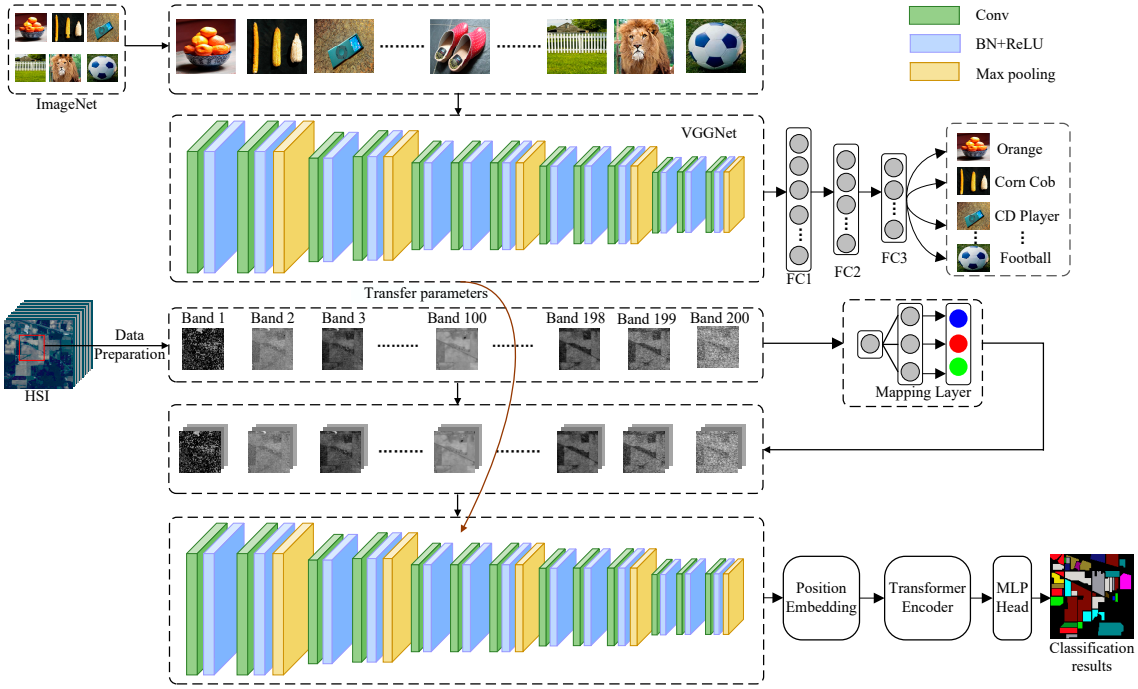

Figure 3. The framework of T-SST for HSI classification.

3.1. Heterogeneous Mapping of Two Datasets

There is a problem of simply using transfer learning for HSI classification, due to the fact that the large-scale dataset (i.e., the source dataset) has three channels, but HSI (i.e., the target dataset) contains hundreds of channels. To solve the problem caused by heterogeneous transfer learning, a mapping layer is used to handle the issue of different number of channels (i.e., bands) of the two datasets.

The pre-trained model on the large-scale ImageNet dataset has three channels of input (i.e., R, G, and B), but CNN in T-SST receives one band as input.

Let $O \in R^{W \times H}$ be the input of CNN, in which $W \times H$ represents the weight and height of a 2D patch. $O' \in R^{W \times H \times 3}$ is the mapped data for subsequent processing, and $\alpha \in R^{3 \times 1}$. Therefore,

$$O' = O \times \alpha. \tag{8}$$

There are three learnable parameters in the heterogeneous mapping. The mapping operation is combined with subsequent CNN to form an end-to-end learning system.

3.2. The Proposed T-SST for HSI Classification

Transfer learning is a technique that aims at extracting knowledge from the source domain and applying it to the target domain [44]. The learned knowledge from the source task is used to improve the performance of the target task. In deep learning-based transfer learning, deep models can learn a lot of knowledge from a large dataset such as ImageNet, and the learnt knowledge can be transferred to a new task such as HSI classification. Therefore, the proper usage of transfer learning may reduce the number of necessary training samples.

Many previous studies proved that the learnt weights in CNN of the original domain can be re-used in the new task [45]. For an image classification task, the first several layers usually extract low-level features (i.e., blobs, corners, and edges), and the low-level features are usually common in image classification tasks. Due to the similarity tasks between the ImageNet and HSI classification, the transfer learning step can be facilitated by fine-tuning on HSI classification task. Specifically, the learned weights of VGGNet on the ImageNet dataset can be utilized to initialize the network of the HSI classification and then fine-tune the weights on an HSI classification task.

Here, a new classification framework titled T-SST is proposed for HSI classification, which is a combination of transferred VGGNet, modified Transformer (i.e., DenseTransformer), and MLP. In T-SST, VGGNet with 16 layers is used, which was trained on the ImageNet dataset, and the well-trained weights of all the convolutional layers from the source task are transferred to our target task. Then, these initialized weights are fine-tuned on the HSI dataset.

Using transferred VGGNet, more robust and discriminant features can be extracted compared with the original VGGNet, which is useful for the following processing. The obtained features using transferred VGGNet are used as inputs of Transformer. Specially, a 2D patch, which contains the neighboring pixels in a band of HSI, is an input of transferred VGGNet. VGGNet uses all the convolutional layers to extract the features of the input, then the obtained features are fed into the DenseTransformer. The following MLP is used to obtain the final classification results.

3.3. The Proposed T-SST-L for HSI Classification

Without sufficient training samples, the model faces a problem of "overfitting", which means that the classification accuracy on test data will be low. This problem is expected when T-SST is applied to HSI classification, because it is a common issue that there are only limited training samples in real application. To address the overfitting issue in T-SST, label smoothing is introduced.

In classification, each training sample $\boldsymbol{x}$ has the corresponding label $y \in \{1, 2, \ldots, C\}$. C is the number of classes. Here, we use a C-dimensional one-hot vector $\boldsymbol{y}$ to represent the label of training sample $\boldsymbol{x}$,

$$\boldsymbol{y}_k = \delta_{k,y}, \tag{9}$$

where $k = 1,\ 2, \ldots, C$, $\delta_{k,y}$ represents the discrete Dirac delta function, which equals 1 for $k = y$ and 0 otherwise.

However, work in [46] has shown that, if we assign all ground truth labels as "hard labels" (i.e., the $\delta_{k,y}$), the model will struggle with many efforts to push the predicted distribution of labels towards the hard label. Moreover, this can be effectively relieved if the labels are properly smoothed, i.e., assigned tiny probability mass on the zeros in $\delta_{k,y}$. Intuitively, this happens because the model becomes too confident about its predictions. Thus, in this paper, a mechanism called label smoothing for encouraging the model to be less confident is introduced in this paper to achieve better performance. Label smoothing changes the original label $\boldsymbol{y}_k$ to $\boldsymbol{y}'_k$, which can be defined as follows:

$$\text{标签平滑：}\quad \boldsymbol{y}'_k = (1-\varepsilon)\delta_{k,y} + \left(1-\delta_{k,y}\right)\frac{\varepsilon}{C^{-1}}, \tag{10}$$

where y'_k mixes the label of training sample x and the fixed uniform distribution of the number of C^{-1} classes; ε is the smoothing factor [47].

By reducing the model to learn the full probability label of each training sample, the label smoothing mechanism can mitigate the overfitting problem and increase the generalization ability of the model in a simple form.

4. Experimental Results

4.1. Hyperspectral Datasets

In this study, the performance of proposed methods is evaluated on three public datasets, including the Salinas, Pavia University (Pavia), and Indian Pines datasets. Table 1 reports the information of all the datasets including the sensor, number of bands, spatial resolution, pixel size dimension, number of classes, and year of data acquisition. The descriptions of all the datasets are summarized below.

Table 1. Details of each HIS dataset.

Dataset	Sensor	Number of Bands	Spatial Resolution	Size	Number of Classes	Year of Data Acquisition
Salinas	AVIRIS	204	3.7 m	512 × 217	16	1998
Pavia	ROSIS	103	1.3 m	610 × 340	9	2001
Indian Pines	AVIRIS	200	20 m	145 × 145	16	1992

For the Salinas dataset, it was collected by the AVIRIS over Salinas Valley, CA, USA, in 1998. After removing 20 bands of low signal to noise ratio (SNR), 204 bands were used in the experiments.

There are 512 × 217 pixels with 3.7-m spatial resolution included in this hyperspectral image. The common 16 classes are labeled in the ground truth. The false-color composite image, the available ground-truth map, and the Scale bar are shown in Figure 4.

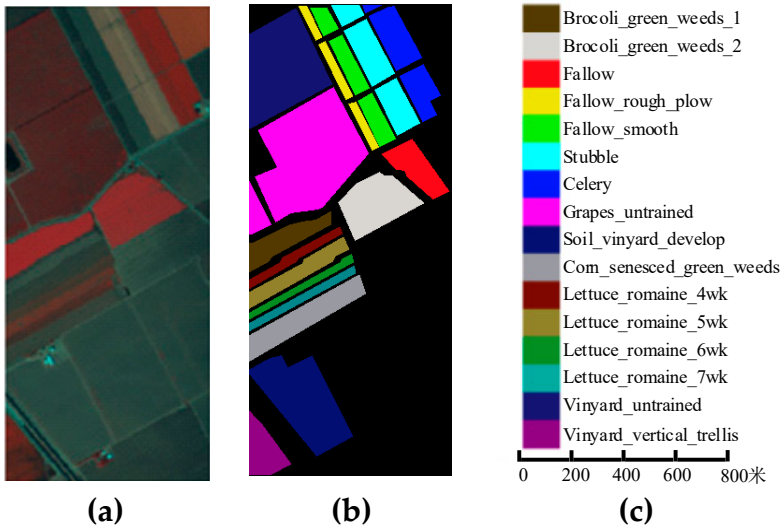

Figure 4. The Salinas dataset. (**a**) False-color composite image; (**b**) ground truth map; (**c**) scale bar.

For the Pavia dataset, it was obtained by the Reflective Optics Spectrographic Imaging System (ROSIS) sensor over an urban scene by Pavia University, Italy, in 2001. This dataset has a size of 610 × 340 pixels with 1.3 m spatial resolution, and the spectral ranges from 0.43 to 0.86 µm. Due to the SNR and water absorption, 12 bands were removed, and the remaining 103 bands were used in the experiments. There are nine urban classes to discriminate. Figure 5 shows the false-color composite image, the available ground-truth map, and the scale bar.

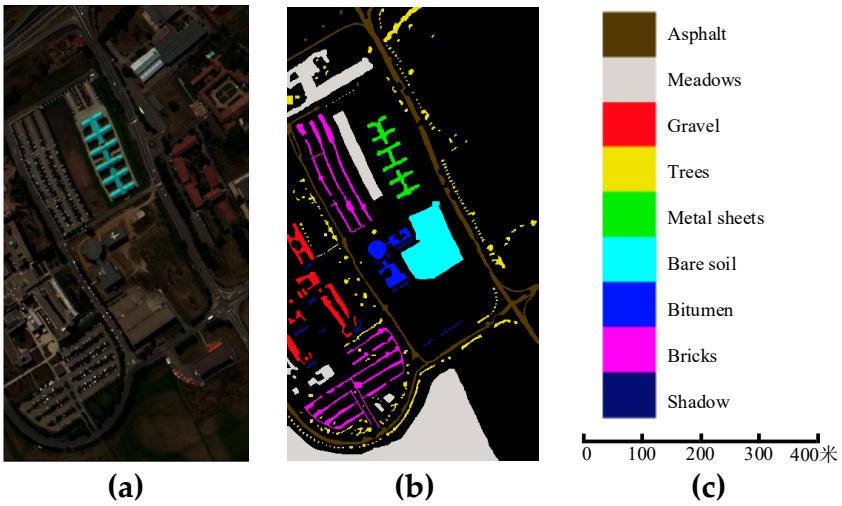

Figure 5. The Pavia dataset. (**a**) False-color composite image; (**b**) ground truth map; (**c**) scale bar.

For the Indian Pines dataset, it was captured by the AVIRIS sensor over the Indian Pines region in Northwestern Indiana, 1992. The spatial size of it is 145 × 145 with the spatial resolution of 20 m. The number of spectral bands is 200 with the wavelengths from 0.4 to 2.5 μm after discarding 20 water absorption bands. Sixteen ground truth classes are labeled in the available ground truth. The false-color composite image, the available ground-truth map, and the scale bar are shown in Figure 6.

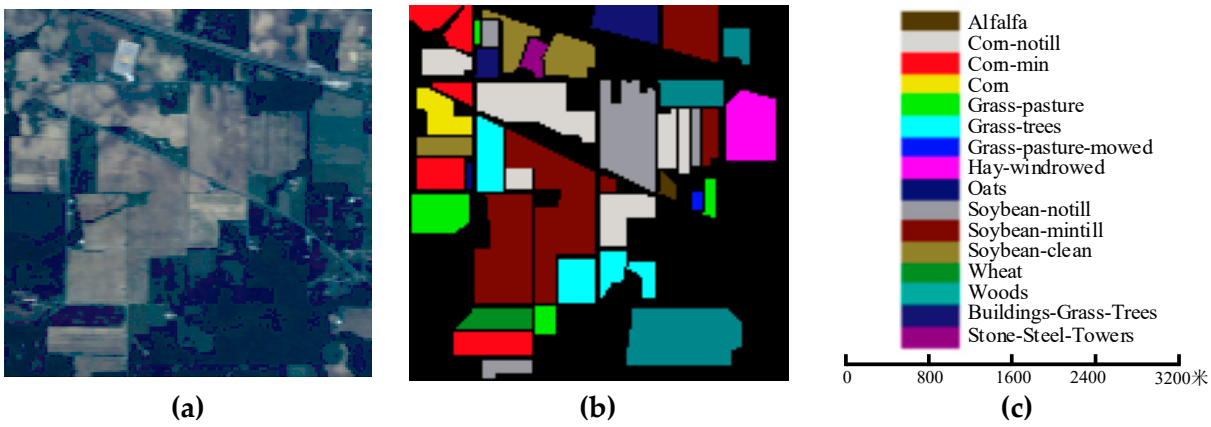

Figure 6. The Indian Pines dataset. (**a**) False-color composite image; (**b**) ground truth map; (**c**) scale bar.

Each dataset is divided into three subsets: training set, validate set, and test set. The training set consists of 200 labeled samples for model training, which are randomly selected from all the labeled samples, the validate set includes 50 labeled samples for guiding the model, and the remains are used as the test set.

The input HSI datasets are normalized into [−0.5, 0.5]. According to Figure 1, to capture the relationships in a long distance in the spectral domain, the HSI data cube consists of $\boldsymbol{n}$ bands, where the order of bands follows the spectral order. The neighborhood pixels of each sample are set to 33 × 33, and then these samples are fed into VGGNet.

4.2. Training Details

The VGGNet with 16 layers are adopted in the experiments, which contains 13 convolutional layers and three fully connected layers. The characteristic of VGGNet is mainly adopted small convolutional filters with 3 × 3 size. Moreover, the 13 convolutional layers can be divided into five groups, and each group contains two or three convolutional layers, which are followed by a max pooling layer. The architecture of VGGNet in SST is similar to

VGGNet; to reduce the overfitting for HSI classification, several convolutional layers are ignored, the first three convolutional layer groups reduce one convolutional layer, and the fourth convolutional layer reduces the first two convolutional layers. In addition, for the T-SST, the architecture of VGGNet adopts all the convolutional layers, which are used as initialized weights. For the Pavia and Indian pines datasets, dropout is added. Then, HSI training samples are exploited to fine-tune the VGGNet by the back-propagation algorithm. Due to the input of VGGNet in each band, a mapping layer is designed, whose input is each band and output is three features. Then these features are sent to the VGGNet to extract the discriminative features.

During the training procedure, the mini-batch algorithm is adopted, which is set to 128 for all the datasets [48]. For the SST, the initial learning rate is set to 8×10^{-5} for the Salinas dataset, and 9×10^{-5} for the Pavia and Indian Pines datasets, and the learning rate is reduced by 0.9 for each epoch. In the experiments, the small learning rate is found to be suitable for the SST for HSI classification. Additionally, for the T-SST, the learning rate is set to 3×10^{-4}, 9×10^{-5}, and 1×10^{-4} for the Salinas, Pavia, and Indian Pines datasets, respectively. The learning rate is reduced by 0.7, 0.9, and 0.9 for each epoch for the Salinas, Pavia, and Indian Pines datasets, respectively. Additionally, the training epoch is set to 150 for the Salinas dataset, and for the Pavia and Indian Pines datasets, the training epoch is set to 80. Furthermore, the overall accuracies (OA), average (AA) accuracies, and kappa coefficient (K) are considered to evaluate the performance of different methods.

4.3. Parameter Analysis

To give a comprehensive study of the spatial-spectral Transformer, some key parameters involved in the Transformer are analyzed in this section, including the number of attention heads, the depth of the Transformer encoder, and the smoothing factor ε of the proposed T-SST-L. For the number of attention heads and the depth of the Transformer encoder, they not only influence the robustness of the model, but also affect the complexity of the model. With the increment of the model depth, it is easy for the model to encounter the overfitting problem. For the smoothing factor ε, the value of ε could influence the performance of the model. Thus, the optimal parameter settings of these parameters are needed to investigate.

To analyze the influences of these parameters to the model, other parameters are fixed; 200 training samples are used for searching optimal parameters. Figure 7 shows the analysis results evaluated by OA (%) on the Salinas, Pavia, and Indian Pines datasets, respectively. For searching for the optimal number of attention heads, one, two, four, eight, and 16 of attention heads are chosen. This result is shown in Figure 7a: it can be seen that the best number of attention heads is two for all the datasets. For the depth of the Transformer encoder, the depths ranging from one to five are searched. Figure 7b shows that the best depth is two for all the datasets: it can be concluded that the lacking depths may cause incomplete information, while for the too deep models, the accuracies are decreased, due to the large amounts of parameters that are needed to train. In the experiments, according to these results, the number of attention heads and depths of the Transformer encoder are set to two for all the datasets to lead to better classification results.

ε is the smoothing factor of T-SST-L; to validate the influences of T-SST-L with different values of ε, the grid search method is utilized to search the optimal value of ε varying from 0.2 to 0.9. The OA of different values of ε is shown in Figure 8. As can be seen, the OAs of different values of ε are fluctuant, but the proposed T-SST-L obtains the best result when the value of ε is set to 0.9 on the three datasets. Therefore, in all experiments, the value of ε is set to 0.9 for all datasets to obtain the best performance for HSI classification.

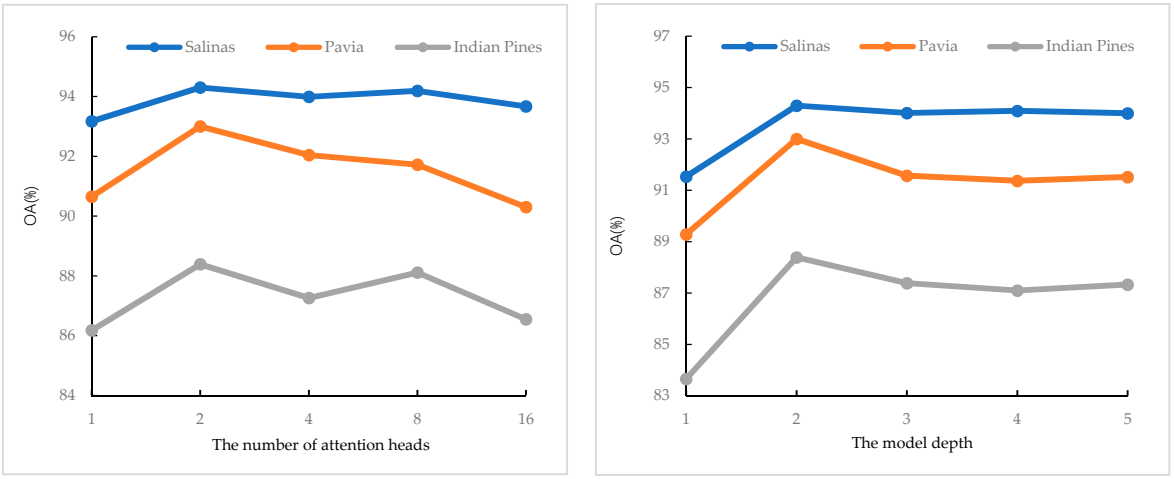

Figure 7. Results of SST varying (**a**) the number of attention heads and (**b**) the model depth.

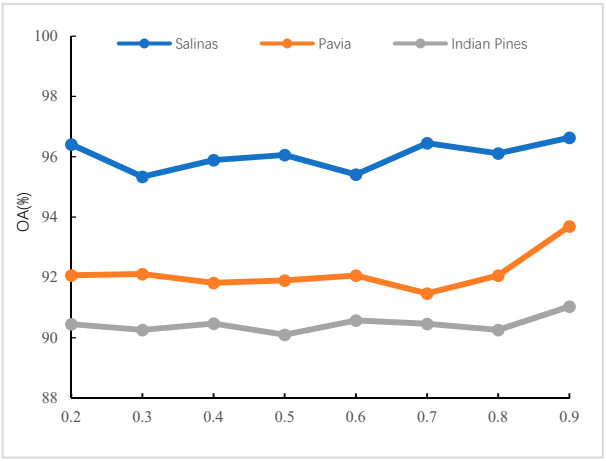

Figure 8. Analysis of the performing results of the parameters ε.

4.4. The Classification Results of SST and SST-FA for HSI Classification

In this section, the proposed SST and SST-FA are verified by using several comparison experiments, including the traditional methods (i.e., RBF-SVM and EMP-SVM) and the classical CNN related methods (i.e., CNN, SSRN, and VGG). For RBF-SVM, the radial basis function is adopted as the kernel, and a grid search method is used for finding the best value of C and γ, which are in the exponentially growing sequence $\{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$. The best parameters C and γ are obtained using five-fold cross-validation. EMP-SVM combines EMP with SVM; for EMP, a disk-shape structure element with an increasing size from two to eight is designed for the opening and closing operations in EMP to extract features. The architectures of CNN and SSRN are implemented following the settings described in [34,49], respectively.

The experimental results of SST and SST-FA are reported in Tables 2–4. As can be seen, the values of OA, AA, and kappa achieve by the proposed SST-FA are the best, which reach 94.94%, 93.37%, and 88.98% on the Salinas, Pavia, Indian Pines, respectively.

Table 2. Test accuracy with different preprocessing methods on the Salinas dataset.

Method	RBF-SVM	EMP-SVM	CNN	SSRN	VGG	SST	SST-FA
OA(%)	83.09 ± 1.08	87.59 ± 2.39	88.40 ± 2.13	88.73 ± 2.06	89.25 ± 5.20	94.42 ± 1.59	**94.94 ± 1.31**
AA(%)	85.46 ± 2.06	88.65 ± 1.93	92.48 ± 2.79	92.63 ± 1.47	90.16 ± 4.81	**93.11 ± 1.85**	93.05 ± 0.92
$K\times100$	81.07 ± 1.19	86.15 ± 2.75	88.27 ± 2.73	87.51 ± 2.26	88.05 ± 5.82	93.73 ± 1.80	**94.32 ± 1.48**
Brocoli_green_weeds_1	94.15 ± 0.50	95.28 ± 4.21	81.75 ± 5.36	**100.00 ± 0.00**	82.46 ± 17.94	92.58 ± 11.90	92.44 ± 13.63
Brocoli_green_weeds_2	98.57 ± 0.89	98.49 ± 0.28	88.89 ± 7.69	**100.00 ± 0.00**	88.85 ± 2.11	97.50 ± 4.06	97.47 ± 3.37
Fallow	90.56 ± 0.50	70.53 ± 22.58	89.35 ± 3.85	72.85 ± 13.98	**92.04 ± 1.61**	91.47 ± 9.55	88.47 ± 12.43
Fallow_rough_plow	98.93 ± 0.40	**99.80 ± 0.12**	80.01 ± 5.86	99.78 ± 2.15	89.86 ± 0.37	91.98 ± 11.84	92.60 ± 7.79
Fallow_smooth	95.23 ± 0.63	93.34 ± 4.40	88.40 ± 2.57	**98.68 ± 0.54**	98.32 ± 3.18	95.59 ± 8.23	93.09 ± 13.63
Stubble	99.25 ± 0.91	99.36 ± 0.22	90.45 ± 6.38	99.85 ± 0.89	**99.97 ± 1.53**	99.53 ± 0.84	99.46 ± 0.90
Celery	98.82 ± 0.33	97.48 ± 1.84	97.96 ± 2.79	**99.43 ± 0.43**	96.98 ± 3.36	98.65 ± 2.14	97.89 ± 3.43
Grapes_untrained	78.50 ± 0.57	90.09 ± 7.83	69.63 ± 9.62	62.90 ± 14.95	75.66 ± 11.52	**95.22 ± 2.90**	95.09 ± 2.78
Soil_vinyard_develop	94.11 ± 0.50	98.89 ± 0.08	89.33 ± 9.79	97.87 ± 3.96	98.44 ± 0.13	99.46 ± 1.12	**99.70 ± 0.44**
Corn_senesced_green_weeds	85.56 ± 0.36	90.74 ± 2.23	85.75 ± 9.08	88.08 ± 5.20	96.76 ± 13.29	98.28 ± 1.88	**99.69 ± 0.60**
Lettuce_romaine_4wk	90.63 ± 0.78	93.06 ± 2.02	88.92 ± 9.72	86.54 ± 7.17	95.72 ± 28.25	**99.87 ± 0.15**	98.47 ± 2.80
Lettuce_romaine_5wk	99.48 ± 0.03	**99.98 ± 0.04**	82.07 ± 9.37	99.41 ± 5.75	97.41 ± 1.01	89.80 ± 10.95	96.08 ± 3.15
Lettuce_romaine_6wk	20.08 ± 2.47	77.91 ± 39.03	82.65 ± 8.30	84.74 ± 7.69	96.82 ± 37.37	97.32 ± 4.98	**97.47 ± 3.02**
Lettuce_romaine_7wk	66.29 ± 1.67	98.34 ± 0.99	85.41 ± 9.25	**99.33 ± 1.18**	98.23 ± 1.40	96.37 ± 3.82	93.81 ± 2.07
Vinyard_untrained	59.14 ± 1.06	35.73 ± 30.40	76.80 ± 18.48	**93.40 ± 12.17**	84.13 ± 35.13	81.42 ± 16.37	86.72 ± 15.08
Vinyard_vertical_trellis	66.96 ± 0.78	79.35 ± 6.22	56.06 ± 12.02	**98.71 ± 2.58**	50.94 ± 41.64	65.07 ± 14.69	60.33 ± 21.01

Table 3. Test accuracy with different preprocessing methods on the Pavia dataset.

Method	RBF-SVM	EMP-SVM	CNN	SSRN	VGG	SST	SST-FA
OA(%)	80.06 ± 1.52	89.43 ± 1.20	91.41 ± 1.44	91.59 ± 3.57	91.72 ± 2.12	92.74 ± 1.08	**93.37 ± 1.96**
AA(%)	69.48 ± 3.03	80.37 ± 3.60	81.03 ± 4.99	**87.56 ± 3.57**	84.13 ± 5.41	83.60 ± 2.35	85.01 ± 3.78
$K\times100$	74.19 ± 2.02	85.81 ± 1.64	89.12 ± 1.83	88.96 ± 1.56	89.53 ± 2.70	90.80 ± 1.41	**91.65 ± 2.49**
Asphalt	88.75 ± 2.70	91.84 ± 1.75	92.36 ± 5.62	**99.62 ± 2.57**	89.85 ± 2.05	94.38 ± 5.08	98.36 ± 5.16
Meadows	94.94 ± 1.47	98.02 ± 0.68	98.85 ± 1.04	98.85 ± 4.91	98.68 ± 1.23	**99.9 ± 0.27**	97.49 ± 0.94
Gravel	34.89 ± 15.46	66.63 ± 14.10	42.94 ± 22.88	**90.99 ± 25.41**	68.99 ± 11.70	72.86 ± 6.29	48.88 ± 14.58
Trees	64.11 ± 9.76	92.10 ± 5.14	93.52 ± 3.53	**94.13 ± 8.55**	86.72 ± 2.18	77.73 ± 9.44	89.7 ± 9.26
Metal sheets	89.35 ± 8.20	86.26 ± 21.92	99.33 ± 0.72	**99.63 ± 2.74**	99.71 ± 0.21	79.02 ± 7.64	83.21 ± 9.83
Bare soil	66.54 ± 6.15	78.66 ± 7.12	98.62 ± 1.54	77.72 ± 8.91	91.36 ± 3.23	**99.88 ± 3.93**	98.32 ± 1.52
Bitumen	55.77 ± 22.79	74.43 ± 10.12	45.56 ± 27.30	59.63 ± 15.34	75.24 ± 22.87	59.78 ± 15.77	**84.97 ± 13.92**
Bricks	73.39 ± 7.31	90.92 ± 3.70	90.60 ± 5.68	67.52 ± 23.28	97.24 ± 1.59	97.02 ± 1.61	**98.43 ± 2.03**
Shadow	57.61 ± 21.51	74.11 ± 22.21	67.52 ± 33.51	**99.90 ± 2.89**	49.41 ± 26.69	71.84 ± 14.85	65.78 ± 8.79

All the experimental results demonstrate that SST-FA reaches the best performance on all the HSI datasets, which has advantages in alleviating the overfitting. For the proposed SST, take the Salinas dataset as an example, compared to the traditional methods, the OA is 11.33% and 6.83% points higher than that of RBF-SVM and EMP-SVM, respectively; the AA is 7.65% and 4.46% points better, respectively, and the kappa is 12.66% and 7.58% points higher. Besides, compared to CNN, OA of SST is improved by 6.02%, 1.33%, and 1.96% on the Salinas, Pavia, and Indian Pines datasets, respectively. In addition, compared to CNN-based methods including SSRN and VGG, for the Indian Pines dataset, the accuracy of the proposed SST achieves 88.77%, which increases of 5.56% and 1.97%, respectively. SST also offers improvement on the Salinas and Pavia datasets. Figure 9 shows learning curves of SST including loss, accuracy of training, and validate samples on the three datasets. The experimental results demonstrate that the proposed SST has the advantages in extracting sequential information of HSI.

Table 4. Test accuracy with different preprocessing methods on the Indian Pines dataset.

Method	RBF-SVM	EMP-SVM	CNN	SSRN	VGG	SST	SST-FA
OA(%)	79.75 ± 1.89	81.53 ± 2.39	86.81 ± 1.28	83.21 ± 1.25	86.80 ± 1.08	88.77 ± 1.07	**88.98 ± 0.66**
AA(%)	60.11 ± 2.56	**72.43 ± 6.35**	68.30 ± 4.82	62.88 ± 7.12	63.18 ± 1.98	66.75 ± 6.44	68.15 ± 1.06
$K\times100$	75.82 ± 2.35	78.87 ± 2.78	84.20 ± 1.57	80.86 ± 1.39	84.10 ± 1.29	86.48 ± 1.27	**86.70 ± 0.78**
Alfalfa	38.67 ± 26.66	50.27 ± 41.37	38.70 ± 32.39	25.00 ± 35.08	35.56 ± 9.07	35.40 ± 33.24	**62.91 ± 29.44**
Corn-notill	80.01 ± 9.17	72.43 ± 7.51	84.64 ± 6.07	87.16 ± 12.44	90.74 ± 1.98	90.52 ± 3.51	**91.56 ± 3.23**
Corn-min	52.50 ± 11.83	80.77 ± 5.42	63.15 ± 22.10	**95.61 ± 20.25**	38.80 ± 20.66	50.81 ± 15.66	63.28 ± 13.48
Corn	23.85 ± 22.71	**74.44 ± 19.28**	55.37 ± 45.79	20.60 ± 17.82	55.32 ± 27.30	22.13 ± 36.06	13.83 ± 27.66
Grass-pasture	61.00 ± 16.10	77.04 ± 8.10	56.41 ± 10.40	**84.93 ± 21.91**	51.24 ± 11.62	66.00 ± 10.46	64.64 ± 10.76
Grass-trees	91.76 ± 4.49	95.27 ± 3.54	96.90 ± 3.94	**99.71 ± 6.22**	94.21 ± 2.60	93.50 ± 2.83	94.40 ± 1.18
Grass-pasture-mowed	16.58 ± 23.12	49.63 ± 49.64	42.54 ± 38.16	28.70 ± 48.04	13.58 ± 19.21	**42.96 ± 43.87**	29.63 ± 22.83
Hay-windrowed	87.89 ± 8.56	97.43 ± 2.90	85.84 ± 27.28	**97.65 ± 2.15**	73.44 ± 18.42	82.15 ± 17.78	70.66 ± 21.10
Oats	**42.21 ± 37.64**	28.95 ± 39.68	12.75 ± 18.33	9.28 ± 43.22	24.07 ± 34.05	**5.88 ± 13.15**	8.82 ± 17.65
Soybean-notill	74.57 ± 8.69	74.92 ± 7.72	90.33 ± 6.48	80.77 ± 8.61	85.78 ± 4.08	91.54 ± 3.91	**92.57 ± 2.70**
Soybean-mintill	92.74 ± 3.23	85.30 ± 4.85	94.21 ± 3.32	83.09 ± 6.57	95.70 ± 1.82	97.29 ± 1.82	**97.37 ± 2.38**
Soybean-clean	48.60 ± 13.61	60.49 ± 10.01	58.07 ± 12.47	**70.77 ± 21.72**	69.35 ± 8.91	70.19 ± 11.47	65.73 ± 8.25
Wheat	61.59 ± 25.50	75.54 ± 37.88	85.99 ± 14.62	**98.48 ± 12.96**	96.75 ± 2.72	81.24 ± 17.71	58.94 ± 22.58
Woods	94.26 ± 2.15	96.56 ± 3.24	96.63 ± 2.53	96.99 ± 3.86	98.02 ± 1.93	97.16 ± 3.36	**98.88 ± 1.80**
Buildings-Grass-Trees	52.78 ± 28.23	73.77 ± 9.55	67.38 ± 28.38	38.76 ± 18.17	58.04 ± 42.38	80.15 ± 26.42	**98.55 ± 2.20**
Stone-Steel-Towers	42.78 ± 34.42	66.09 ± 43.03	63.96 ± 33.59	53.85 ± 44.51	30.22 ± 12.73	61.08 ± 37.67	**78.69 ± 38.57**

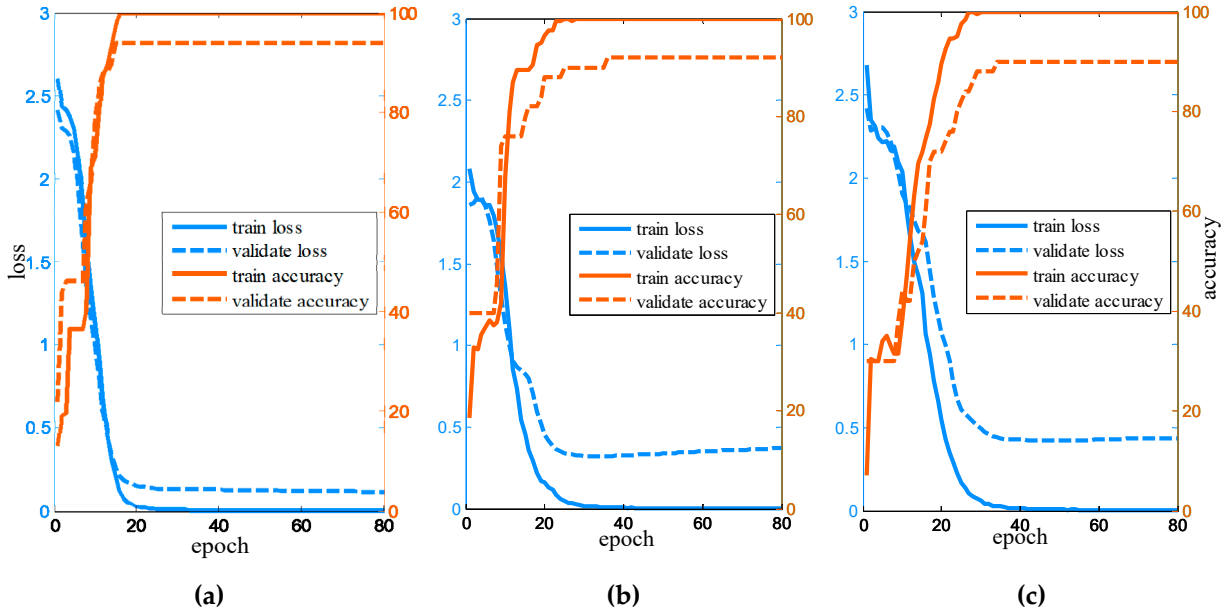

Figure 9. Learning curves of SST on the three HSI datasets. (**a**) Salinas; (**b**) Pavia; (**c**) Indian Pines.

4.5. The Classification Results of the Proposed T-SST and T-SST-L for HSI Classification

In this section, the experimental results of the proposed T-SST and T-SST-L are presented to test the performance of HSI classification, which use the pre-trained VGGNet on a large dataset as the initialized weights of VGGNet. To further verify that the proposed T-SST and T-SST-L are superior to other methods for HSI classification, EMP-random forest (RF), EMP-CNN, VGG, and T-CNN are selected for comparison. For EMP-RF, the detailed information about EMP is the same as the previous settings. Then, the features extracted by EMP are fed into the RF classifier with 200 decision trees [50]. Additionally, EMP-CNN combines EMP with CNN and is implemented for spectral-spatial classification. Specifically, the architecture design of EMP-CNN is similar to CNN [49]. Moreover, we adopt a comparison method named VGG, whose architecture follows all the convolutional layers

in VGGNet; after that, a fully connected layer is added for HSI classification. In addition, to demonstrate the proposed Transformer method with transfer learning is effective, CNN with transfer learning (T-CNN) is also utilized. Three bands are randomly selected from all bands, then all the VGGNet weights of the first seven convolutional layers are used to initialize T-CNN to finish HSI classification task.

The results of the proposed T-SST and T-SST-L are reported in Tables 5–7. The proposed T-SST-L achieves competitive results as compared to state-of-the-art well-designed networks. The OA of the proposed T-SST-L reaches 96.83%, 93.73%, and 91.20% on Salinas, Pavia, and Indian Pines datasets, respectively. In addition, it can be observed that the proposed T-SST is superior to other existing methods on all the datasets. Specifically, the T-SST outperforms the EMP-RF by 4.16%, 5.08%, and 4.96% in terms of OA on Salinas, Pavia, and Indian Pines datasets, respectively. Additionally, the accuracy obtained by the proposed T-SST on the Salinas dataset is 3.09% better than that of the EMP-CNN, while, for the Indian Pines dataset, it is 2.84%. In addition, compared to T-CNN, the proposed T-SST achieves about 2% improvements on the Salinas dataset in terms of OA and K. Furthermore, compared to the proposed T-SST, the proposed T-SST-L increases accuracies by 1.03% and 1.14% on Salinas and Indian Pines datasets, respectively. It demonstrates that label smoothing is an effective method to prevent the overfitting problem.

Table 5. Test accuracy with different preprocessing methods on the Salinas dataset.

Method	EMP-RF	EMP-CNN	VGG	T-CNN	T-SST	T-SST-L
OA(%)	91.64 ± 0.82	92.71 ± 0.66	89.25 ± 5.20	93.17 ± 1.46	95.80 ± 0.90	**96.83 ± 0.97**
AA(%)	92.05 ± 2.14	93.22 ± 0.62	90.16 ± 4.81	90.34 ± 2.79	94.20 ± 1.38	**96.08 ± 0.83**
$K\times100$	90.68 ± 0.92	91.83 ± 0.74	88.05 ± 5.82	92.16 ± 1.79	95.28 ± 1.01	**96.43 ± 1.09**
Brocoli_green_weeds_1	**99.39 ± 0.48**	93.91 ± 6.37	82.46 ± 17.94	90.25 ± 11.95	95.00 ± 10.13	98.47 ± 2.83
Brocoli_green_weeds_2	98.64 ± 2.55	85.93 ± 8.49	88.85 ± 2.11	**99.28 ± 1.45**	91.55 ± 4.95	93.51 ± 6.67
Fallow	86.65 ± 7.69	93.31 ± 11.47	92.04 ± 1.61	71.84 ± 23.63	91.97 ± 10.59	93.19 ± 9.53
Fallow_rough_plow	98.91 ± 1.40	97.94 ± 1.57	89.86 ± 0.37	93.22 ± 6.48	92.75 ± 4.05	**97.38 ± 1.91**
Fallow_smooth	91.71 ± 13.54	94.90 ± 4.25	98.32 ± 3.18	95.69 ± 4.86	97.39 ± 2.17	**98.57 ± 1.46**
Stubble	97.71 ± 1.66	98.91 ± 1.54	99.97 ± 1.53	96.14 ± 3.22	99.91 ± 0.18	**99.99 ± 0.03**
Celery	99.21 ± 0.58	97.42 ± 1.66	96.98 ± 3.36	**100.00 ± 0.00**	97.44 ± 3.36	98.71 ± 1.05
Grapes_untrained	89.18 ± 2.53	93.71 ± 2.40	75.66 ± 11.52	94.24 ± 3.98	96.61 ± 2.23	**96.85 ± 2.33**
Soil_vinyard_develop	99.56 ± 0.36	98.41 ± 0.86	98.44 ± 0.13	**100.00 ± 0.00**	99.96 ± 0.06	99.68 ± 0.60
Corn_senesced_green_weeds	92.34 ± 6.93	95.93 ± 3.25	96.76 ± 13.29	97.86 ± 1.48	97.64 ± 3.67	**98.25 ± 2.30**
Lettuce_romaine_4wk	82.34 ± 17.84	93.25 ± 7.93	95.72 ± 28.25	93.15 ± 10.85	95.92 ± 4.15	**97.69 ± 1.65**
Lettuce_romaine_5wk	92.30 ± 9.95	92.98 ± 9.99	97.41 ± 1.01	**98.78 ± 1.49**	97.06 ± 3.74	97.25 ± 2.89
Lettuce_romaine_6wk	93.16 ± 12.10	91.48 ± 11.01	**96.82 ± 37.37**	76.05 ± 33.94	93.80 ± 4.74	95.21 ± 2.88
Lettuce_romaine_7wk	85.10 ± 28.67	98.07 ± 2.78	**98.23 ± 1.40**	95.83 ± 5.27	94.53 ± 4.84	96.56 ± 3.30
Vinyard_untrained	78.36 ± 8.13	78.57 ± 5.36	84.13 ± 35.13	81.07 ± 16.68	91.53 ± 3.80	**93.13 ± 5.15**
Vinyard_vertical_trellis	**88.30 ± 6.08**	86.80 ± 7.39	50.94 ± 41.64	62.03 ± 23.75	74.11 ± 12.41	82.86 ± 13.46

Table 6. Test accuracy with different preprocessing methods on the Pavia dataset.

Method	EMP-RF	EMP-CNN	VGG	T-CNN	T-SST	T-SST-L
OA(%)	88.29 ± 2.24	91.65 ± 1.22	91.72 ± 2.12	92.55 ± 2.06	93.37 ± 1.60	**93.73 ± 1.73**
AA(%)	81.87 ± 5.41	83.13 ± 3.57	84.13 ± 5.41	84.84 ± 2.43	85.30 ± 2.55	**89.45 ± 3.66**
$K\times100$	84.36 ± 3.09	89.42 ± 1.56	89.53 ± 2.70	90.54 ± 2.64	61.61 ± 2.03	**92.10 ± 2.19**
Asphalt	94.22 ± 2.45	91.99 ± 2.81	89.85 ± 2.05	92.59 ± 4.48	90.88 ± 6.09	86.63 ± 3.32
Meadows	96.21 ± 1.71	97.95 ± 1.32	98.68 ± 1.23	99.66 ± 0.19	99.60 ± 0.52	98.77 ± 0.41
Gravel	70.14 ± 10.05	49.02 ± 10.07	68.99 ± 11.70	70.42 ± 12.74	**98.51 ± 11.94**	86.16 ± 9.81
Trees	85.59 ± 10.22	**89.25 ± 4.96**	86.72 ± 2.18	72.63 ± 11.76	82.02 ± 2.03	86.70 ± 3.28
Metal sheets	85.26 ± 24.14	95.31 ± 5.13	**99.71 ± 0.21**	95.84 ± 3.54	82.39 ± 7.64	**100 ± 0.00**
Bare soil	65.94 ± 10.99	98.45 ± 1.47	91.36 ± 3.23	**98.62 ± 2.29**	99.61 ± 3.54	98.48 ± 5.38
Bitumen	52.93 ± 22.35	73.50 ± 13.36	75.24 ± 22.87	**82.02 ± 12.91**	64.63 ± 11.87	**86.58 ± 11.03**
Bricks	94.29 ± 3.25	93.24 ± 3.25	97.24 ± 1.59	95.35 ± 4.84	**98.85 ± 1.35**	96.51 ± 2.85
Shadow	**92.24 ± 16.31**	59.49 ± 21.87	49.41 ± 26.69	56.41 ± 27.04	51.17 ± 23.38	65.24 ± 28.69

Table 7. Test accuracy with different preprocessing methods on the Indian Pines dataset.

Method	EMP-RF	EMP-CNN	VGG	T-CNN	T-SST	T-SST-L
OA(%)	85.10 ± 1.59	87.22 ± 1.37	86.80 ± 1.08	87.83 ± 0.78	90.06 ± 0.68	**91.20 ± 1.35**
AA(%)	74.85 ± 2.35	74.77 ± 5.50	63.18 ± 1.98	65.32 ± 2.77	80.12 ± 4.49	**83.05 ± 5.94**
$K\times100$	82.96 ± 1.84	85.03 ± 1.62	84.10 ± 1.29	85.35 ± 1.00	88.05 ± 0.82	**89.85 ± 1.17**
Alfalfa	60.88 ± 31.15	44.81 ± 30.09	35.56 ± 9.07	37.78 ± 36.95	84.64 ± 6.09	**92.36 ± 8.59**
Corn-notill	78.10 ± 7.99	87.81 ± 7.01	**90.74 ± 1.98**	93.03 ± 3.61	85.86 ± 3.84	88.96 ± 6.83
Corn-min	**83.43 ± 11.56**	73.58 ± 11.10	38.80 ± 20.66	45.46 ± 17.23	68.91 ± 22.92	71.79 ± 21.13
Corn	59.23 ± 16.02	64.95 ± 37.09	55.32 ± 27.30	34.79 ± 33.53	47.86 ± 40.14	54.96 ± 45.01
Grass-pasture	74.90 ± 10.75	71.72 ± 21.07	51.24 ± 11.62	**82.07 ± 10.06**	67.90 ± 21.08	72.32 ± 16.20
Grass-trees	94.34 ± 6.50	88.79 ± 5.09	94.21 ± 2.60	94.02 ± 4.19	91.08 ± 6.22	**93.55 ± 5.45**
Grass-pasture-mowed	32.15 ± 39.56	63.95 ± 42.30	13.58 ± 19.21	38.52 ± 32.84	77.78 ± 39.13	**79.26 ± 39.66**
Hay-windrowed	99.85 ± 0.17	96.50 ± 3.02	73.44 ± 18.42	70.01 ± 33.41	98.36 ± 2.11	**99.90 ± 0.20**
Oats	5.26 ± 15.79	20.00 ± 40.00	24.07 ± 34.05	13.33 ± 26.67	48.17 ± 39.89	**50.53 ± 42.96**
Soybean-notill	74.18 ± 10.67	88.60 ± 4.97	85.78 ± 4.08	89.20 ± 6.52	**91.24 ± 3.77**	90.85 ± 2.44
Soybean-mintill	90.84 ± 4.55	92.43 ± 3.42	95.70 ± 1.82	94.45 ± 2.11	**98.23 ± 0.32**	97.08 ± 0.67
Soybean-clean	71.13 ± 15.48	71.39 ± 17.31	69.35 ± 8.91	77.95 ± 21.29	**76.98 ± 4.45**	76.85 ± 5.79
Wheat	**99.50 ± 0.90**	91.08 ± 7.05	96.75 ± 2.72	92.61 ± 8.42	85.56 ± 13.47	93.33 ± 10.81
Woods	97.29 ± 1.35	95.60 ± 4.29	**98.02 ± 1.93**	93.05 ± 4.13	95.04 ± 1.44	95.12 ± 1.11
Buildings-Grass-Trees	82.12 ± 17.62	74.28 ± 14.20	58.04 ± 42.38	36.49 ± 28.41	80.56 ± 18.16	**87.02 ± 16.96**
Stone-Steel-Towers	**94.46 ± 2.12**	70.88 ± 27.37	30.22 ± 12.73	52.31 ± 30.74	83.74 ± 21.95	84.95 ± 20.63

4.6. Analysis of Transformer Encoder Representation of the Proposed T-SST

In this subsection, to see how the proposed T-SST captures long-distance dependency relations, we analyze the Transformer encoder representation by visualizing the normalized attention weights. Figures 10–12 display normalized attention weights between bands on the three hyperspectral datasets.

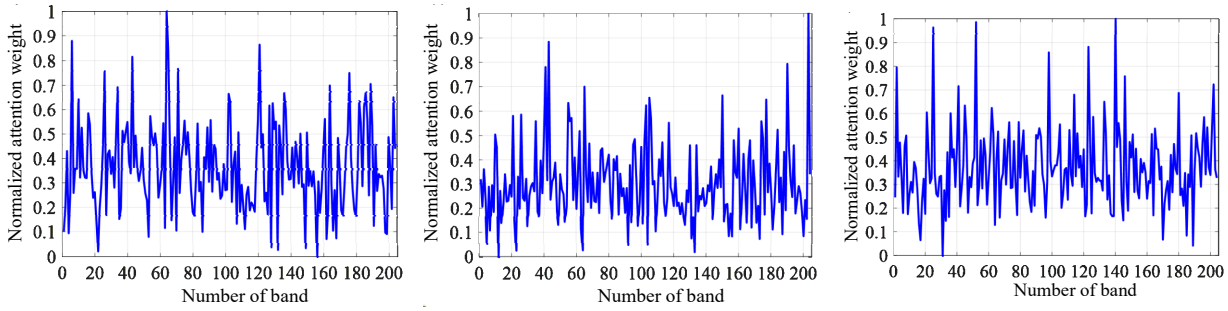

Figure 10. Visualization of attention weights for the head 2, layer 2 of T-SST on the Salinas dataset. (**a**) Dependency relations between the band 1 and another band; (**b**) dependency relations between the band 100 and another band; (**c**) dependency relations between the band 204 and another band.

Since there are hundreds of bands in HSI, it is difficult to make complete visualization for each of them. Therefore, the first band, the middle band, and the last band are chosen to illustrate the long-range dependency relations between the chosen band and another band. Specially, band 1, band 100, and band 204 are chosen on the Salinas dataset; band 1, band 50, and band 103 are chosen on the Pavia dataset; and band 1, band 100, and band 200 are chosen on the Indian Pines dataset.

As shown in Figures 10–12, the values of attention weight strongly fluctuate on the three hyperspectral datasets, and the value of attention weight can be high even if the two bands have a long distance.

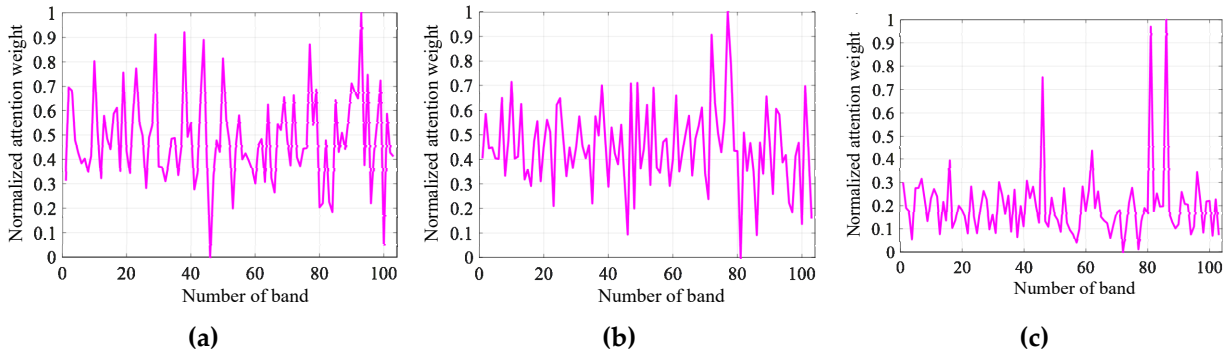

Figure 11. Visualization of attention weights for the head 2, layer 2 of T-SST on the Pavia dataset. (**a**) Dependency relations between the band 1 and another band; (**b**) dependency relations between the band 50 and another band; (**c**) dependency relations between the band 103 and another band.

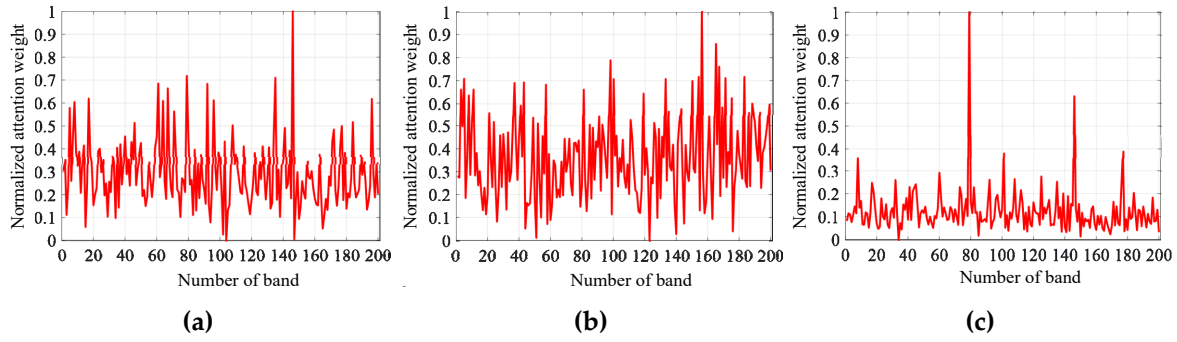

Figure 12. Visualization of attention weights for the head 2, layer 2 of T-SST on the Indian Pines dataset. (**a**) Dependency relations between the band 1 and another band; (**b**) dependency relations between the band 100 and another band; (**c**) dependency relations between the band 200 and another band.

Take the Salinas dataset as an example: in Figure 10a, the value of normalized attention weight between band 50 and band 203 is very high, although distance of the two bands is far. The results demonstrate that T-SST tends to capture long-range dependency relations.

4.7. Classification Maps

Here, to fully evaluate the classification results, Figures 13–15 display classification maps of different methods from a visual perspective on all the datasets; the methods include EMP-SVM, CNN, SSRN, VGG, and our proposed methods (i.e., SST-FA and T-SST-L). Through comparison, it can be observed that classification maps of EMP-SVM produce more errors for all the datasets, while for the proposed SST and T-SST, there exist fewer noise points. In addition, in Figure 15, compared to other CNN-based methods, for example, in comparison of CNN, SSRN, and VGG (see Figure 15b–d), many pixels are misclassified on the boundary among different classes on the Indian Pines dataset, while the proposed methods are able to classify more classes correctly (i.e., Soybean-clean) and have a clearer distinction. Obviously, SST-FA and T-SST-L produce classification maps with the highest quality compared to other approaches, which demonstrates that the proposed SST-FA and T-SST-L are effective in enhancing the performance of model, respectively.

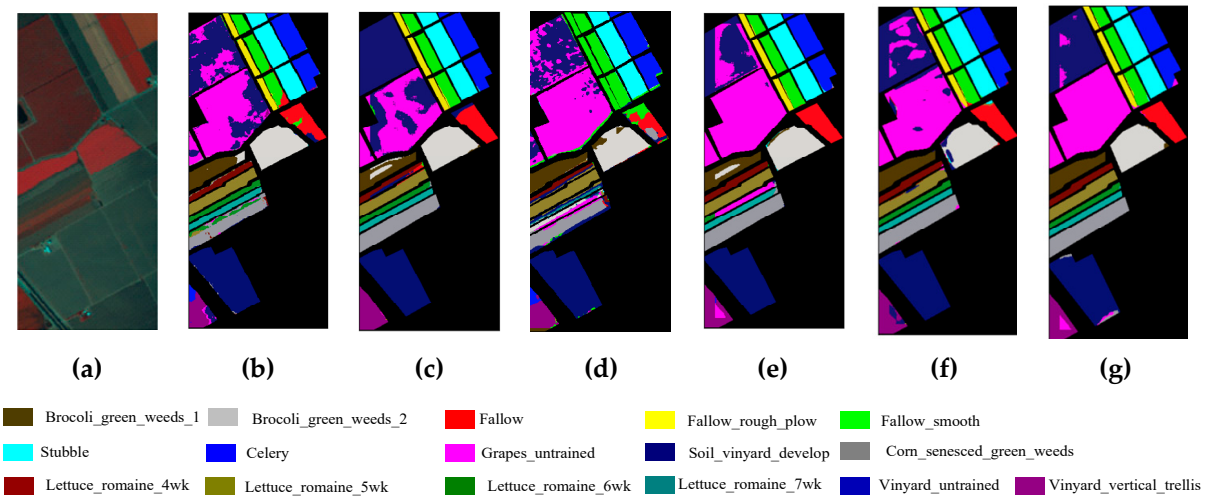

Figure 13. Salinas. (**a**) False-color composite image. The classification maps using (**b**) EMP-SVM; (**c**) CNN; (**d**) SSRN; (**e**) VGG; (**f**) SST-FA; (**g**) T-SST-L.

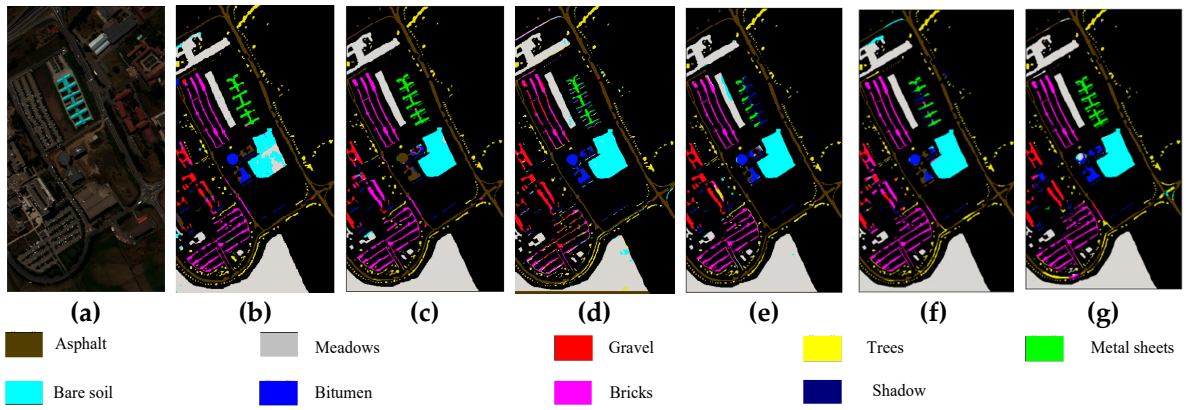

Figure 14. Pavia. (**a**) False-color composite image. The classification maps using (**b**) EMP-SVM; (**c**) CNN; (**d**) SSRN; (**e**) VGG; (**f**) SST-FA; (**g**) T-SST-L.

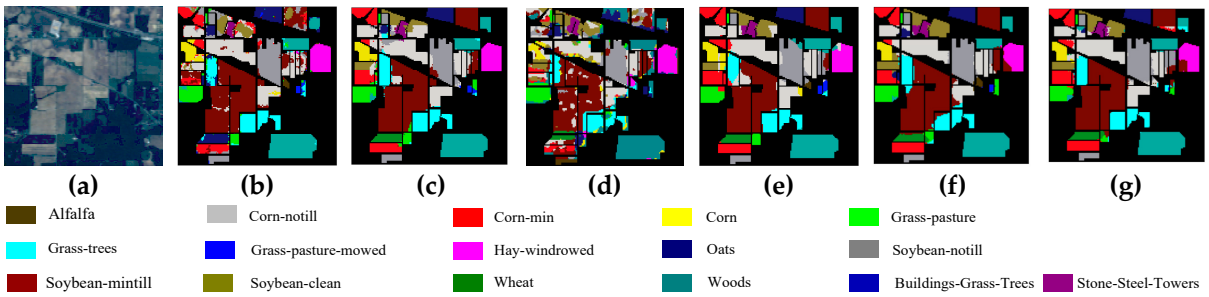

Figure 15. Indian Pines. (**a**) False-color composite image. The classification maps using (**b**) EMP-SVM; (**c**) CNN; (**d**) SSRN; (**e**) VGG; (**f**) SST-FA; (**g**) T-SST-L.

4.8. Time Consumption

The execution time of different methods for the three HSI datasets with 200 training samples is reported in Table 8. All the experiments are conducted on a computer with an Intel Core i7-10700F processor with 2.9 GHz, 64 GB of DDR4 RAM, an NVIDIA GeForce RTX 3070 graphical processing unit (GPU). For the traditional methods including RBF-SVM, EMP-SVM, and EMP-RF, the processing time is short, but these methods achieve poor performance. In addition, for the CNN and T-CNN, since CNN includes fewer parameters

than other competitive deep-learning-based methods and T-CNN only contains three bands for transfer learning, the running time is short. Additionally, compared to CNN, SSRN and VGG take longer time, because SSRN needs more epochs to train the network and VGG contains many 3×3 convolutional kernels. For the proposed methods (i.e., SST, T-SST, and T-SST-L), since the proposed methods consider the model of Transformer, the processing times are long.

Table 8. Running time (min.) of different methods on the three HSI datasets with 200 training samples.

Datasets \ Methods	RBF-SVM	EMP-SVM	EMP-RF	EMP-CNN	CNN	T-CNN	SSRN	VGG	SST	SST-FA	T-SST	T-SST-L
Salinas	0.83	0.06	0.06	1.60	1.86	0.22	3.47	2.53	22.42	22.46	28.86	28.89
Pavia	0.71	0.06	0.07	0.64	0.79	0.24	1.62	0.95	16.69	16.70	18.01	18.05
Indian Pines	0.47	0.03	0.03	0.35	0.55	0.17	2.27	0.71	21.43	21.44	26.91	26.91

5. Conclusions

In this study, Transformer is investigated for HSI classification. Specifically, Dense-Transformer is proposed, which uses dense connection to alleviate the vanishing-gradient problem in the training of a Transformer.

Moreover, two classification frameworks (i.e., SST and T-SST) have been proposed to handle the task of HSI classification. The proposed methods obtained superior performance in terms of classification accuracy on the three popular HSI datasets.

For the proposed SST-based HSI classification method, it took full advantage of CNN to capture spatial features of a 2D patch and made best of DenseTransformer to capture relationships in a long distance in spectral domain. The used self-attention mechanism considered the intrinsic sequential data structure of a pixel vector of his, and the combination of CNN and DenseTransformer obtained the spectral-spatial discriminate features, which are useful for the following HSI classification.

In addition, DenseTransformer combined with dynamic feature augmentation (i.e., SST-FA) is proposed for alleviating the overfitting problem, and thus it enhances the accuracy of the model in a simple form.

Furthermore, the effectiveness of T-SST has been tested. The proposed T-SST combined transfer learning and SST to further improve the classification performance. To use the pre-trained model on the ImageNet dataset, a heterogeneous mapping layer was designed, which was used to map the model from the source domain (i.e., ImageNet dataset) to target domain (i.e., HSI). The obtained experimental results have shown the usefulness of T-SST for HSI classification.

At last, label smoothing has been proved as a useful regularization technique in Transformer-based HSI classification. The proposed T-SST-L led to high performance compared to SST, SST-FA, T-SST, and other methods.

The proposed SST and T-SST have shown the potential of the proposed DenseTransformer for HSI classification. However, it is in the early stage of Transformer-based HSI classification. In our future work, various improvements of Transformer can be used to open a new widow for HSI accurate classification.

Author Contributions: Conceptualization: Y.C.; methodology: X.H., Y.C., and Z.L.; writing—original draft preparation: X.H., Y.C., and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of China under the Grant 61971164.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: [http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes].

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

References

1. Bioucas-Dias, J.; 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. Della, C.; Bekit, A.; Lampe, B.; Chang, C.-I. Hyperspectral image classification via compressive sensing. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 8290–8303. [CrossRef]
3. Datt, B.; McVicar, T.; Van Niel, T.; Jupp, D.; Pearlman, J. Preprocessing EO-1 Hyperion hyperspectral data to support the application of agricultural indexes. *IEEE Trans. Geosci. Remote Sens.* **2003**, *41*, 1246–1259. [CrossRef]
4. Mahlein, A.; Oerke, E.; Steiner, U.; Dehne, H. Recent advances in sensing plant diseases for precision crop protection. *Eur. J. Plant Pathol.* **2012**, *133*, 197–209. [CrossRef]
5. Murphy, R.; Schneider, S.; Monteiro, S. Consistency of measurements of wavelength position from hyperspectral imagery: Use of the ferric iron crystal field absorption at ∼900 nm as an indicator of mineralogy. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 2843–2857. [CrossRef]
6. Lu, G.; Fei, B. Medical hyperspectral imaging: A review. *J. Biomed. Opt.* **2014**, *19*, 010901. [CrossRef] [PubMed]
7. Hestir, E.; Brando, V.; Bresciani, M.; Giardino, C.; Matta, E.; Villa, P.; Dekker, A. Measuring freshwater aquatic ecosystems: The need for a hyperspectral global mapping satellite mission. *Remote Sens. Environ.* **2015**, *167*, 181–195. [CrossRef]
8. Camps-Valls, G.; Tuia, D.; Bruzzone, L.; Benediktsson, J.A. Advances in Hyperspectral image classification: Earth monitoring with statistical learning methods. *IEEE Signal Process. Mag.* **2014**, *31*, 45–54. [CrossRef]
9. Ghamisi, P.; Maggiori, E.; Li, S.; Souza, R.; Tarablaka, Y.; Moser, G.; Giorgi, A.D.; Fang, L.; Chen, Y.; Chi, M.; et al. New frontiers in spectral-spatial hyperspectral image classification: The latest advances based on mathematical morphology, Markov random fields, segmentation, sparse representation, and deep learning. *IEEE Geosci. Remote Sens. Mag.* **2018**, *6*, 10–43. [CrossRef]
10. Rasti, B.; Hong, D.; Hang, R.; Ghamisi, P.; Kang, X.; Chanussot, J.; Benediktsson, J.A. Feature extraction for hyperspectral imagery: The evolution from shallow to deep: Overview and toolbox. *IEEE Geosci. Remote Sens. Mag.* **2020**, *8*, 60–88. [CrossRef]
11. Audebert, N.; Le, B.; Lefevre, S. Deep learning for classification of hyperspectral data: A comparative review. *IEEE Geosci. Remote Sens. Mag.* **2019**, *7*, 159–173. [CrossRef]
12. Melgani, F.; Lorenzo, B. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Trans. Geosci. Remote Sens.* **2004**, *42*, 1778–1790. [CrossRef]
13. Gualtieri, J.; Chettri, S. Support vector machines for classification of hyperspectral data. In Proceedings of the IGARSS, Honolulu, HI, USA, 24–28 July 2000; pp. 813–815.
14. Ghamisi, P.; Yokoya, N.; Li, J.; Liao, W.; Liu, S.; Plaza, J.; Rasti, B.; Plaza, A. Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art. *IEEE Geosci. Remote Sens. Mag.* **2017**, *5*, 37–78. [CrossRef]
15. 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]
16. 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]
17. Mura, M.D.; 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.* **2011**, *8*, 542–546. [CrossRef]
18. Fang, L.; He, N.; Li, S.; Ghamisi, P.; Benediktsson, J.A. Extinction profiles fusion for hyperspectral images classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 1803–1815. [CrossRef]
19. Ma, X.; Wang, H.; Geng, J. Spectral-spatial classification of hyperspectral image based on deep auto-encoder. *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **2016**, *9*, 4073–4085. [CrossRef]
20. Zhong, P.; Gong, Z.; Li, S.; Schönlieb, C.-B. Learning to diversify deep belief networks for hyperspectral image classification. *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **2017**, *55*, 3516–3530. [CrossRef]
21. 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]
22. Gong, Z.; Zhong, P.; Yu, Y.; Hu, W.; Li, S. A CNN with multiscale convolution and diversified metric for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 3599–3618. [CrossRef]
23. Mou, L.; Ghamisi, P.; Zhu, X.X. Deep recurrent neural networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3639–3655. [CrossRef]
24. Zhang, X.; Sun, Y.; Jiang, K.; Li, C.; Jiao, L.; Zhou, H. Spatial sequential recurrent neural networks for hyperspectral image classification. *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **2018**, *11*, 4141–4155. [CrossRef]
25. Feng, J.; Yu, H.; Wang, L.; Zhang, X.; Jiao, L. Classification of hyperspectral images based on multiclass spatial–spectral generative adversarial networks. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 5329–5343. [CrossRef]
26. Zhu, L.; Chen, Y.; Ghamisi, P.; Benediktsson, J.A. Generative adversarial networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 5046–5063. [CrossRef]

27. Zhu, K.; Chen, Y.; Ghamisi, P.; Jia, X.; Benediktsson, J.A. Deep convolutional capsule network for hyperspectral image spectral and spectral-spatial classification. *Remote Sens.* **2019**, *11*, 223. [CrossRef]
28. Paoletti, M.; Haut, J.; Fernandez-Beltran, R.; Plaza, J.; Plaza, A.; Li, J.; Pla, F. Capsule networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 2145–2160. [CrossRef]
29. Hu, W.; Huang, Y.; Li, W.; Zhang, F.; Li, H. Deep convolutional neural networks for hyperspectral image classification. *J. Sens.* **2015**, *2015*, 258619. [CrossRef]
30. Li, W.; Wu, G.; Zhang, F.; Du, Q. Hyperspectral image classification using deep pixel-pair features. *IEEE Trans. Geosci. Remote Sens.* **2016**, *55*, 844–853. [CrossRef]
31. Haut, J.; Paoletti, M.; Plaza, J.; Plaza, A.; Li, J. Visual attention-driven hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 8065–8080. [CrossRef]
32. Li, Y.; Zhang, H.; Shen, Q. Spectral-spatial classification of hyperspectral imagery with 3D convolutional neural network. *Remote Sens.* **2017**, *9*, 67. [CrossRef]
33. He, M.; Li, B.; Chen, H. Multi-scale 3D deep convolutional neural network for hyperspectral image classification. In Proceedings of the IEEE International Conference on Image Processing, Beijing, China, 17–20 September 2017; pp. 3904–3908.
34. Zhong, Z.; Li, J.; Luo, Z.; Chapman, M. Spectral–spatial residual network for hyperspectral image classification: A 3D deep learning framework. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 847–858. [CrossRef]
35. Tal, L.; Emmanuel, D.; Yoav, G. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 521–535.
36. Hang, R.; Liu, Q.; Hong, D.; Ghamisi, P. Cascaded recurrent neural networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 5384–5394. [CrossRef]
37. Tang, G.; Mathias, M.; Rio, A.; Sennrich, R. Why self-attention? A targeted evaluation of neural machine translation architectures. *arXiv* **2018**, arXiv:1808.08946v3. Available online: https://arxiv.org/pdf/1808.08946v3.pdf (accessed on 11 November 2020).
38. Lin, Z.; Feng, M.; Santos, C.; Yu, M.; Xiang, B.; Zhou, B.; Bengio, Y. A structured self-attentive sentence embedding. *arXiv* **2017**, arXiv:1703.03130. Available online: https://arxiv.org/pdf/1703.03130.pdf (accessed on 9 March 2017).
39. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
40. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
41. Windrim, L.; Melkumyan, A.; Murphy, R.J.; Chlingaryan, A.; Ramakrishnan, R. Pretraining for hyperspectral convolutional neural network classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 2798–2810. [CrossRef]
42. Hendrycks, D.; Gimpel, K. Gaussian error linear units (GELUs). *arXiv* **2020**, arXiv:1606.08415. Available online: https://arxiv.org/pdf/1606.08415.pdf (accessed on 8 July 2020).
43. Ba, J.; Kiros, J.; Hinton, G. Layer normalization. *arXiv* **2016**, arXiv:1607.06450. Available online: https://arxiv.org/pdf/1607.06450v1.pdf (accessed on 21 July 2016).
44. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In Proceedings of the Advance in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 3320–3328.
45. Oquab, M.; Bottou, L.; Laptev, I.; Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 20–23 June 2014; pp. 1717–1724.
46. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531. Available online: https://arxiv.org/abs/1503.02531 (accessed on 9 March 2015).
47. Muller, R.; Kornblith, S.; Hinton, G. When does label smoothing help? In Proceedings of the NeurIPS, Vancouver, BC, Canada, 8–14 December 2019; pp. 4696–4705.
48. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2017**, arXiv:1412.6980. Available online: https://arxiv.org/abs/1412.6980 (accessed on 30 January 2017).
49. Chen, Y.; Zhu, L.; Ghamisi, P.; Jia, X.; Li, G.; Tang, L. Hyperspectral images classification with gabor filtering and convolutional neural network. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 2355–2359. [CrossRef]
50. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]