remote sensing

MDPI

Article

A High-Performance Spectral-Spatial Residual Network for Hyperspectral Image Classification with Small Training Data

Wijayanti Nurul Khotimah [1,*], Mohammed Bennamoun [1], Farid Boussaid [2], Ferdous Sohel [3] and David Edwards [4]

1 Department of Computer Science and Software Engineering, The University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia; mohammed.bennamoun@uwa.edu.au
2 Department of Electrical, Electronic and Computer Engineering, The University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia; farid.boussaid@uwa.edu.au
3 Information Technology, Mathematics & Statistics, Murdoch University, 90 South Street, Murdoch, Perth, WA 6150, Australia; F.Sohel@murdoch.edu.au
4 School of Plant Biology and The UWA Institute of Agriculture, The University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia; dave.edwards@uwa.edu.au
* Correspondence: wijayantinurul.khotimah@research.uwa.edu.au

Received: 13 August 2020; Accepted: 21 September 2020; Published: 24 September 2020

Abstract: In this paper, we propose a high performance Two-Stream spectral-spatial Residual Network (TSRN) for hyperspectral image classification. The first spectral residual network (sRN) stream is used to extract spectral characteristics, and the second spatial residual network (saRN) stream is concurrently used to extract spatial features. The sRN uses 1D convolutional layers to fit the spectral data structure, while the saRN uses 2D convolutional layers to match the hyperspectral spatial data structure. Furthermore, each convolutional layer is preceded by a Batch Normalization (BN) layer that works as a regularizer to speed up the training process and to improve the accuracy. We conducted experiments on three well-known hyperspectral datasets, and we compare our results with five contemporary methods across various sizes of training samples. The experimental results show that the proposed architecture can be trained with small size datasets and outperforms the state-of-the-art methods in terms of the Overall Accuracy, Average Accuracy, Kappa Value, and training time.

Keywords: hyperspectral image classification; two stream residual network; deep learning; Batch Normalization

1. Introduction

Hyperspectral imaging has received much attention in recent years due to its ability to capture spectral information that is not detected by the naked human eye [1]. Hyperspectral imaging provides rich cues for numerous computer vision tasks [2] and a wide range of application areas, including medical [1], military [3], forestry [4], food processing [5], and agriculture [6].

One of the main challenges when analyzing Hyperspectral Images (HSIs) lies in extracting features, which is challenging due to the complex characteristics, i.e., the large size and the large spatial variability of HSIs [7]. Furthermore, HSI is composed of hundreds of spectral bands, in which wavelengths are very close, resulting in high redundancies [7,8]. Traditional machine learning methods are less suitable for HSI analysis because they heavily depend on hand-crafted features, which are commonly designed for a specific task, and are thus not generalizable [9]. In contrast, deep learning

techniques can capture characteristic features automatically [9,10], thus constituting a promising avenue for HSI analysis.

Several deep learning architectures have been proposed to classify HSIs. Many architectures, such as one-dimensional convolutional neural network (1D-CNN) [11,12], one-dimensional generative adversarial network (1D-GAN) [13,14], and recurrent neural network (RNN) [15,16], have been proposed to learn spectral features. Other works, e.g., Reference [17–19], have shown that adding spatial features can improve the classification performance. Numerous spectral-spatial network architectures have been proposed for HSIs [19–28].

A number of methods argue that extracting the spectral and spatial features in two separate streams can produce more discriminative features [25,29,30]. Examples of such methods include stacked denoising autoencoder (SdAE) and 2D-CNN [30], plain 1D-CNN and 2D-CNN [25], spectral-spatial long short-term memory (SSLSTMs) [27], and a spectral-spatial unified network (SSUN) [23]. In terms of the spectral stream, the work of Reference [27] used a LSTM, which considers the spectral values of the different channels as a sequence. However, using LSTM on hundreds of a sequence of channels is complex; thus, [23] tried to simplify the sequence by grouping them. One of the grouping strategies is dividing the adjacent band into the same sequence following the spectral orders. The work in Reference [30] considered spectral values as a vector with noise and used a denoising technique, SdAE, to encode the spectral features. These networks, based on LSTM and SdAE, are all shallow. To increase the accuracy, Reference [25] tried to make a deeper network by employing a simpler layer, based on 1D convolution. The work in Reference [31] considered that the HSI bands have a different variance and correlation. Hence, they cluster the bands into some groups based on their similarity, then extracted the spectral features of each cluster using 1D convolution. Different from Reference [31], the study in Reference [32] considered that different objects have different spectral reflection profiles; hence, they used 2D convolution with a kernel size of 1x1 to extract the spectral features. For the spatial stream, Reference [27] also used LSTM, and due to its complexity, thus used a shallow network. Other approaches [23,25,30] used 2D convolution with a plain network, which could be made deeper, while Reference [31,32] used 2D convolution with a multi-scale input to extract multi-scale spatial features.

Other works claim that extracting spectral and spatial features directly using a single stream network can be more beneficial as it leverages the joint spectral-spatial features [28,33,34]. Most that adopt this approach utilize 3D convolutional layers [12,19,34,35] because they are naturally suited to the 3D cube data structure of HSIs. Reported experiments show that deep 3D-CNN produces better performance compared with 2D-CNN [18]. However, 3D-CNN requires large memory size and expensive computation cost [36]. Moreover, 3D-CNN faces over-smoothing phenomena because it fails to take the full advantage of spectral information, which results in misclassification for small objects and boundaries [23]. In addition, the labeling process of HSIs is labor-intensive, time-consuming, difficult, and thus expensive [37]. Using a complex deep learning architecture, in which parameters are in the millions, to learn from a small labeled dataset may also lead to over-fitting [38]. Moreover, adjusting millions of parameters during the deep-learning training process consumes a lot of time. Devising a deep learning architecture, which can work well on complex data of HSI, in which labeled datasets are small, is desirable.

Another issue with HSI classification based on deep learning is the depth of the network. The deeper the layer is, the richer the features will be, where the first layer of the deep network extracts general characteristics, and the deeper layers extract more specific features [39,40]. However, such deep networks are prone to the vanishing/exploding gradient problem, which occurs when the layers are deeper [41,42]. To solve this problem, Reference [40] reformulated the layers as learning residual functions with reference to the layer inputs. This approach is called a residual network (ResNet), which has become popular because of its remarkable performance on image classification [43]. For HSI classification, a single stream ResNet has been used by Reference [19,44–46].

Another problem related to HSI feature extraction is that the spectral values are prone to noise [47]. However, most of the previous research, which focus on the extraction of spectral features with deep-networks, have not taken noise into account. They usually use a pixel vector along the spectral dimension directly as their spectral network input [23,25,27,29,30], without considering that noise can worsen the classification performance.

Considering the aforementioned challenges and the limitations of existing network architectures, associated with HSI feature extraction and classification, we propose an efficient yet high performance two-stream spectral-spatial residual network. The spectral residual network (sRN) stream uses 1D convolutional layers to fit the spectral data structure, and the spatial residual network (saRN) uses 2D convolutional layers to fit the spatial data structure. The residual connection in the sRN and saRN can solve the vanishing/exploding gradient problem. Since proceeding the convolutional layer with Batch Normalization (BN) layer and full pre-activation rectified linear unit (ReLU) generalizes better than the original ResNet [48], in each of our residual unit, we use BN layer and ReLU layer before the convolutional layer. We then combine our sRN and saRN in a parallel pipeline. As shown in Figure 1, given a spectral input cube X_{ij}^{s} of a pixel x_{ij}, the sRN extracts its spectral features. Concurrently, given a spatial input cube X_{ij}^{sa} of a pixel x_{ij}, the saRN will extract its spatial characteristics. Since the sRN and the saRN use different input sizes and different types of convolution layers, they produce different sizes of feature maps. The gap between the number of spectral feature maps and the number of spatial feature maps can worsen the classification accuracy. To make the number of feature maps in each network proportional, we add an identical fully connected layer at the end of each network. Subsequently, we employ a dense layer to fuse the spectral features and the spatial features. Finally, we classify the joint spectral-spatial features using a softmax layer (Figure 1).

In summary, the main contributions of this research are:

- We propose TSRN, a Two-Stream Spectral-Spatial network with residual connections, to extract spectral and spatial features for HSI classification. The identity shortcut in the residual-unit is parameter-free, thus adding shortcut connections into a residual-unit does not increase the number of parameters. Furthermore, the use of 1D convolutional layers in the sRN and 2D convolutional layers in the saRN results in few trainable parameters. We can, therefore, construct a deeper and wider network with fewer parameters, making it particularly suitable for HSI classification when the amount of available labeled data is small.
- We achieve the state-of-the-art performance on HSI classification with various sizes of training samples (4%, 6%, 8%, 10%, and 30%). Moreover, compared to networks based on 3D convolutional layers, our proposed architecture is faster.

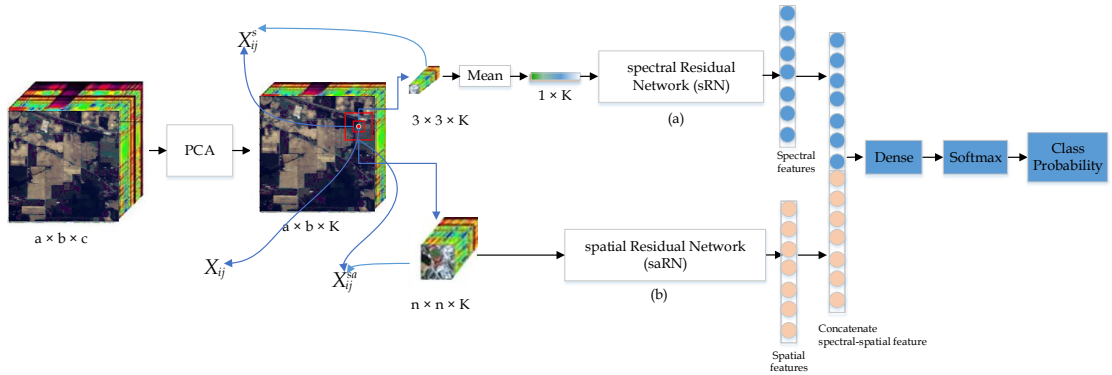


Figure 1. Proposed Two-Stream Spectral-Spatial Residual Network (TSRN) architecture. The details of spectral residual network (sRN) and spatial residual network (saRN) sub-networks are shown in Figure 2.

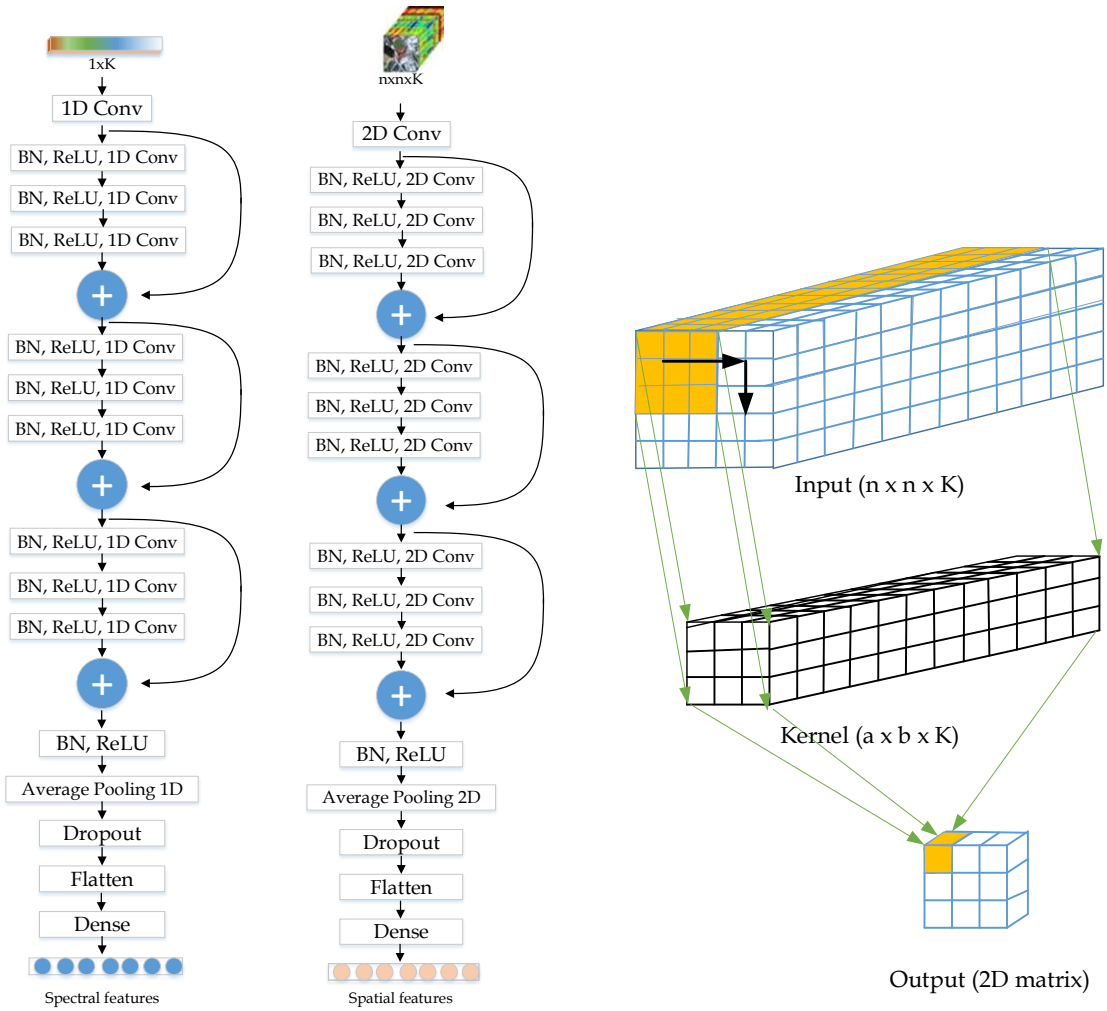


Figure 2. The detailed network of the (**a**) sRN, (**b**) saRN, and (**c**) the detail process of 2D convolution on 3D input.

2. Technical Preliminaries

2.1. CNN

Convolutional Neural Networks (CNNs) have been increasingly used for HSI analysis. A number of works aimed at improving the performance of Deep CNNs (DCNNs) have focused on different aspects, e.g., the network architecture, the type of nonlinear activation function, supervision methods, regularization mechanisms, and optimization techniques [4,49,50]. Based on the network architecture, specifically on the convolutional layers, there are different types of CNNs, namely 1D-CNN, 2D-CNN, and 3D-CNN. The 1D-CNN has one-dimensional filters in its convolutional layers which are naturally fit to the spectral data structure. Consider the case when the size of the input is $K \times 1$, and the kernel size is $B \times 1$, with K representing the number of HSI bands, B is the kernel size, and $B << K$. Wei Hu et al. [11] used 1D convolutional layers to extract the spectral features of HSI. Their network input is an HSI pixel vector, with size $(K, 1)$. This research initiated the use of multiple convolutional layers for HSI classification. Compared to 2D convolution and 3D convolution, the process of 1D convolution, which is shown in Equation (1), is the simplest one.

A 2D-CNN has two-dimensional filters in its convolutional layers. It has been widely used to solve several computer vision problems, such as object detection [51], scene recognition [52], and image classification [51], because of its ability to extract features from a raw image directly. For the HSI classification problem, 2D convolutions have been used to extract spatial features [25,30]. In contrast to RGB images, HSI has a much larger number of channels. Applying 2D convolutions along the hundreds of channels results in more learned parameters [18]; hence, several studies on HSIs, which employ 2D convolutions, do not use all of the channels. Most of them use a dimensionality reduction technique as a preprocessing step with their network [53–55] or use the average of all the images over the spectral bands of the HSI [25].

A 3D-CNN employs 3D convolutions in its convolutional layers. 3D-CNNs are popular for video classification [36], 3D object reconstruction [56], and action recognition [57]. For the case of HSIs, the form of the 3D filter suits the data structure of the HSI cube. Some research papers on HSI classification use 3D convolutional layers to extract the spectral-spatial features directly [18,33]. Their research shows that 3D-CNN outperforms both 1D-CNN and 2D-CNN. However, as shown in Equation (3), the process of 3D convolution requires more parameters, more memory, and requires a higher computational time and complexity compared to 1D convolution in Equation (1) and 2D convolution in Equation (2).

$$v_{ij}^{z} = f\left(\sum_{m} \sum_{b=0}^{B_i-1} k_{ijm}^{b} v_{(i-1)m}^{z+b} + r_{ij}\right), \tag{1}$$

$$v_{ij}^{xy} = f\left(\sum_{m} \sum_{h=0}^{H_i-1} \sum_{w=0}^{W_i-1} k_{ijm}^{wh} v_{(i-1)m}^{(x+h)(y+w)} + r_{ij}\right), \tag{2}$$

$$v_{ij}^{xyz} = f\left(\sum_{m} \sum_{b=0}^{B_i-1} \sum_{h=0}^{H_i-1} \sum_{w=0}^{W_i-1} k_{ijm}^{whb} v_{(i-1)m}^{(x+h)(y+w)(z+b)} + r_{ij}\right), \tag{3}$$

where: i is the layer under consideration, m is the index of feature map, z is the index that corresponds to the spectral dimension, v_{ij}^{z} is the output of the ith layer and the jth feature map at position z , k_{ijm}^{b} is the kernel value at index b on the layer i and feature map j, r_{ij} is the bias at layer i and feature map j. For the 1D convolution in Equation (1), B_i is the size of the 1D filter in layer i, while, for the 3D convolution in Equation (3), B_i is the depth of 3D kernel. W_i and H_i are the width and height of the kernel, respectively, for both 2D and 3D convolutions.

The expensive computational cost and memory demand of 3D convolution has led studies to investigate alternative network architectures based on (2D + 1D) convolutions. For instance, in the case of action recognition, a study in Reference [58] proposed to replace 3D convolution with m parallel streams of n 2D and one 1D convolution. This study empirically showed that their network, which is based on (2D + 1D) convolution, achieves around 40% reduction in model size and yields a drastic reduction in the number of learning parameters compared to another network with 3D convolution. In Reference [36], a simplified 3D convolution was implemented using 2D and 1D convolutions in three different blocks: 2D followed by 1D, 2D and 1D in parallel, and 2D followed by 1D with skip connections. These blocks were subsequently interleaved using a sequence network. The proposed architecture has a depth of 199 and a model size of 261 MB, which is much lighter compared to the 3D-CNN, in which model size is 321 MB when the depth is 11. The architecture was also shown to be faster than its 3D-CNN counterpart [36]. Using (2D + 1D) convolutions instead of 3D convolutions allows the network to be deeper without significantly increasing the number of parameters. Such deep networks can extract richer features and have been shown to outperform 3D-CNN architectures [36,58,59]. Because the model size and the number of parameters grow dramatically as the network becomes deeper, the training of deep 3D-CNNs is extremely difficult with the risk of overfitting.

2.2. Residual Network

Deeper CNN can extract richer features [39]. In some cases, when the networks are deeper, their accuracy degrades because of the vanishing/exploding gradients problem [60]. Hence, He et al. [40] proposed to use a shortcut connection to perform identity mapping without adding extra parameters or extra computational time. The shortcut connection outputs are added to the output of the stacked layers, and a ReLU is applied as the activation function. This network is named ResNet. It has achieved outstanding classification accuracy on some image benchmark datasets, such as ImageNet, ILSVRC 2015, and CIFAR-10.

He et al. [48] followed up their work on ResNet by analyzing the propagation formulation behind the residual unit. Their analysis has shown that a "clean" information path in the skip connection results in the lowest training loss compared to those with scaling, gating, and convolution. Regarding the position of the ReLU activation function in the residual building blocks, they proved that putting ReLU and BN before the add function (full pre-activation) generalizes better than the original ResNet [40], which used ReLU after the add function (post-activation). The difference between pre-activation and post-activation in the residual building blocks is shown in Figure 3.

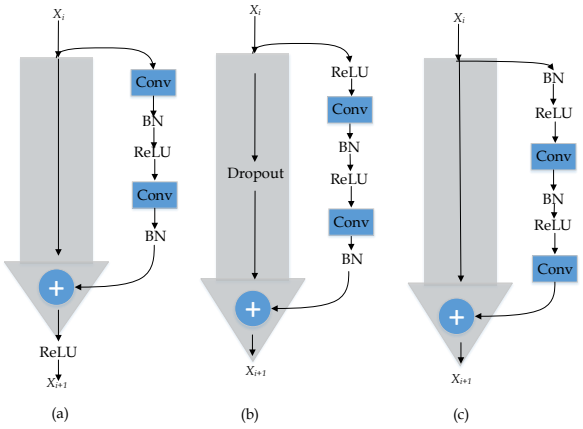


Figure 3. (**a**) Original residual unit with clear short-cut connection. (**b**) rectified linear unit (ReLU)-only pre-activation with dropout short-cut connection. (**c**) Full pre-activation with clear short-cut connection.

For the use of ResNet for HSI classification, Zhong et al. [44] proved that with the same size of convolutional layers, ResNet achieves better recognition accuracy than a plain CNN. Then, they explored ResNet by applying more various kernel sizes to sequentially extract the spectral features and the spatial features [19]. Roy et al. [61] used 3D convolutional layers followed by 2D convolutional layers in their residual network. Their network achieved high performance with 30% training samples. Meanwhile, Reference [45] explored ResNet by implementing a variable number of kernels in each convolutional layer. The kernel number was set to increase gradually in all convolutional layers like a pyramid to increase the diversity of the spectral-spatial features. In contrast to Reference [19,45,61], which focus on exploring the convolutional layer, Reference [46] improved the ResNet architecture by combining it with a dense convolutional network, which helps the ResNet to explore new features. These various works all improve ResNet performance by using a single network to extract both spectral and spatial features. Our proposed architecture extracts the spectral and spatial features from two separate stream networks to produce distinctive spectral and spatial features.

3. Proposed TSRN Network

The flowchart of the proposed TSRN is displayed in Figure 1. From the diagram, we can see that TSRN has two important residual network streams: a sRN and a saRN. Since the number of bands in the spectral dimension is very large (hundreds of channels), and thus comprises much redundancy,

we first apply PCA to extract the first K principal components. Then, for each pixel, we take a 3×3 cube alongside the spectral direction, which is centered at that pixel, as the input of the sRN stream to learn the spectral features. Similarly, we take an $n \times n \times K$ cube and feed it to the saRN streams to extract the spatial features. In this method, we propose to use the same number of spectral and spatial features. Hence, we need to ensure that their feature map sizes at the output of the sRN and the saRN networks are the same. To this end, we have applied a dense layer with the same number of units in each stream. Then, we apply a fully connected layer to fuse the spectral features and the spatial features. Finally, we use a softmax layer to classify the features. In the next subsection, we will explain in more detail both spectral and spatial networks.

3.1. Spectral Residual Network

Although we have minimized the data redundancy of the HSI by using PCA, the spectral values can still be noisy. In the sRN, we propose to compute the mean of the reflectance in each spectral band before inputting the spectral cube into the sRN to minimize the effects of the noise. Given a pixel x_{ij}, we choose a spectral cube $X_{ij}^{s} \in R^{3\times3\times K}$, which is centered at x_{ij} and K is the number of PCA components. Then, we compute the mean reflectance of each band by using Equation (4), where $k \in K$, $\overline{X}_{ij}^{s} = \{\overline{x}_{ij1}^{s}, \overline{x}_{ij2}^{s}, ..., \overline{x}_{ijK}^{s}\}$, and $\overline{X}_{ij}^{s} \in R^{1xK}$.

$$\overline{x}_{ijk}^{s} = \frac{\sum_{h=j-1}^{h=j+1} \sum_{g=i-1}^{g=i+1} x_{g,h,k}^{s}}{9}. \tag{4}$$

Then, we input the mean of the spectral value ($\overline{X}_{ij}^{s}$) into the sRN. In this proposed architecture, we use three full pre-activation with clear skip connection residual units inspired by Reference [48]. It has been shown that these residual units are better than the traditional residual units. These residual units consist of three different layers.

- One-dimensional convolutional layers, which perform a dot product between every small window of input data ($1 \times B_i$) and the kernel's weights and biases (see Equation (1)).
- BN layer that normalizes the layer inputs of each training mini-batch to overcome the internal covariate shift problem [62]. The internal covariate shift problem is a condition which occurs when the distribution of each layer's inputs in deep-network changes due to the change of the previous layer's parameters. This situation slows down the training. Normalizing the layer's inputs stabilizes its distribution and thus speeds up the training process.
- ReLU is an activation function, which learns the non-linear representations of each feature map's components [63].

In the full pre-activation residual unit, the BN layer and the ReLU activation layer are established before the convolution layer, as shown in Figure 2a. From that figure, we can see that an Average Pooling layer, a Dropout layer, and a Dense layer have been applied at the end of the sRN. The Average Pooling layer and the Dropout layer are used as a regularizer to minimize the over-fitting problem due to the small number of training samples, while the Dense layer is used to perform the high-level reasoning to produce 128 spectral features.

3.2. Spatial Residual Network

The saRN is devised to extract the spatial features of a pixel x_{ij}. The input is an $n \times n \times K$ cube, centered at pixel x_{ij}. Then, the input is processed by the full pre-activation residual unit. As shown in Figure 2b, the rest of the layers architecture of this saRN are similar to those of sRN. The main difference between them is that the saRN uses 2D convolutional layers, while the sRN uses 1D convolutional layers. In the end of this network, 128 spatial features are extracted.

Figure 2c illustrates the 2D convolution process with a spatial cube in which size is $n \times n \times K$, where K is the number of channels or the input depth. Since the input is 3D then the kernel is also 3D,

i.e., the kernel depth must be the same as the input depth. Hence, in this case, the kernel depth must be K. So, we can only select the kernel width and height. The convolution process is performed along the x and y-direction and produces a 2D matrix as output [64].

Given a pixel x_{ij}, the spectral features F_{ij}^{s} produced by sRN and the spatial features F_{ij}^{sa} produced by saRN are given by Equations (5) and (6), respectively.

$$F_{ij}^{s} = sRN(\overline{X}_{ij}^{s}), \tag{5}$$

$$F_{ij}^{sa} = saRN(X_{ij}^{sa}). \tag{6}$$

Using F_{ij}^{s} and F_{ij}^{sa}, we implement a fully connected layer to obtain the joint spectral-spatial features F_{ij}^{ssa} using the formula in Equation (7), where W^{fcl} and b^{fcl} are the weight vector and the bias of the fully connected layer, respectively, and $\oplus$ is the concatenation operation.

$$F_{ij}^{ssa} = f(W^{fcl} \cdot \{F_{ij}^{s} \oplus F_{ij}^{sa}\} + b^{fcl}). \tag{7}$$

After obtaining the joint spectral-spatial feature F_{ij}^{ssa}, we use a softmax regression layer (SRL) to predict the class probability distribution of the pixel x_{ij} by using Equation (8). Here, N is the number of classes, and $P(x_{ij})$ is a vector consisting of the probability distribution of each class on pixel x_{ij}. In the end, the label of the pixel x_{ij} is decided using Equation (9).

$$P(x_{ij}) = \frac{1}{\sum_{n=1}^{N} e^{w_n^{srl}.F_{ij}^{ssa}}} \begin{bmatrix} e^{w_1^{srl}.F_{ij}^{ssa}} \\ e^{w_2^{srl}.F_{ij}^{ssa}} \\ \vdots \\ e^{w_N^{srl}.F_{ij}^{ssa}} \end{bmatrix}, \tag{8}$$

$$label(x_{ij}) = argmaxP(x_{ij}). \tag{9}$$

4. Experiments

4.1. Experimental Datasets

We evaluated the proposed architecture on three publicly available HSI datasets, which are frequently used for pixel-wise HSI classification. These datasets are Indian Pine (IP), Pavia University (PU), and Kennedy Space Center (KSC). These datasets were captured by different sensors, thus having different spatial resolutions and a different number of bands. Each dataset has a different category (class), and each class has a different number of instances. The details of the datasets are provided below:

1. IP Dataset: IP dataset was acquired by Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) hyperspectral sensor data on 12 June 1992, over Purdue University Agronomy farm and nearby area, in Northwestern Indiana, USA. The major portions of the area are Indian Creek and Pine Creek watershed; thus, the dataset is known as the Indian Pine dataset. The captured scene contains 145 × 145 pixels with a spatial resolution of 20 meters per pixel. In other words, the dataset has 21,025 pixels. However, not all of the pixels have ground-truth information. Only 10,249 pixels are categorized between 1 to 16 (as shown in Table 1a), and the remaining pixels remain unknown (labeled with zero in the ground truth). Regarding the spectral information, the entire spectral band of this dataset is 224, with wavelengths ranging from 400 to 2500 nm. Since some of the bands cover the region of water absorption, (104–108), (150–163), and 220, they are removed, so only 200 bands remain Reference [65].
2. KSC Dataset: Same as the IP dataset, the KSC dataset was also collected by AVIRIS sensor in 1996 over the Kennedy Space Center, Florida, USA. Its size is 512 × 614; thus, this dataset consists of

314,368 pixels, but only 5122 pixels have ground-truth information (as shown in Table 1b). The dataset's spatial resolution is 18 meters per pixel, and its band number is 174.

3. PU Dataset: The PU dataset was gathered during a flight campaign over the campus in Pavia, Northern Italy, using a Reflective Optics System Imaging Spectrometer (ROSIS) hyperspectral sensor. The dataset consists of 610 × 610 pixels, with a spatial resolution 1.3 meters per pixel. Hence 207,400 pixels are available in this dataset. However, only 20% of these pixels have ground-truth information, which are labeled into nine different classes, as shown in Table 1c. The number of its spectral bands is 103, ranging from 430 to 860 nm.

Table 1. Detailed categories and number of instances of Indian Pines dataset (The colours represent the colour labels that are used in the figures of Section 4.3).

Label	Category Name	# Pixel
(a) Indian Pines Dataset		
C1	Alfafa	46
C2	Corn-notil	1428
C3	Corn-mintill	830
C4	Corn	237
C5	Grass-pasture	483
C6	Grass-trees	730
C7	Grass-pasture-mowed	28
C8	Hay-windrowed	478
C9	Oats	20
C10	Soybean-notil	972
C11	Soybean-mintill	2455
C12	Soybean-clean	593
C13	Wheat	205
C14	Woods	1265
C15	Building-Grass-Tress	386
C16	Stone-Steel-Towers	93
(b) KSC Dataset		
C1	Scrub	761
C2	Willow swamp	253
C3	Cabbage palm hammock	256
C4	Cabbage palm	252
C5	Slash pine	161
C6	Oak	229
C7	Hardwood swamp	105
C8	Graminoid marsh	431
C9	Spartina marsh	520
C10	Cattail marsh	404
C11	Salt marsh	419
C12	Mud flats	503
C13	Water	927
(c) Pavia University Dataset		
C1	Asphalt	6631
C2	Meadows	18,649
C3	Gravel	2099
C4	Trees	3064
C5	Painted metal sheets	1345
C6	Bare soil	5029
C7	Bitumen	1330
C8	Self-Blocking Bricks	3682
C9	Shadows	947

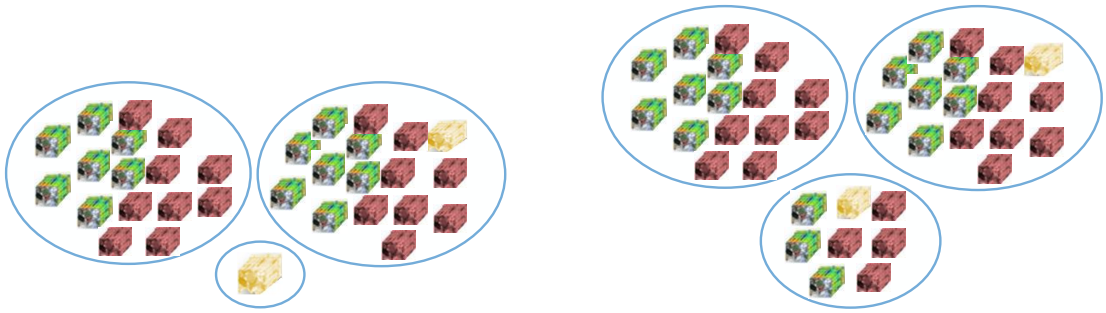
4.2. Experimental Configuration

In our proposed model, we do standardization (a technique to rescale the data to have a mean of 0 and a standard deviation of 1) in advance, before dividing the data into training and

testing. Hyperparameters are initialized based on previous research or optimized during experiments. We initialized the convolution kernel by using the "He normal optimizer" [66] and applied l2 (0.0001) for the kernel regularizer. We use 1D convolutional kernels of size 5 in the sRN sub-network and 2D convolutional kernels of size 3×3 in the saRN sub-network. For the number of filters, we use the same size of filters in each convolution layer, 24. We apply 1D average pooling layer with pool size 2 and 2D average pooling layer with pool size 5×5 in the sRN and saRN, respectively. Furthermore, a 50% dropout is applied in both sub-networks. Then, we trained our model using Adam optimizer with a learning rate of 0.0003 [67].

Regarding batch-size, a constant batch-size sometimes results in a tiny mini-batch (see Figure 4a). Meanwhile, in a network with BN layers, there is dependency between the mini-batch elements because BN uses mini-batch statistics to normalize the activations during the learning process [68]. This dependency may decrease the performance if the mini-batch is too small [68,69]. Some approaches can be applied to overcome this problem. The first is to ignore the samples in the last mini-batch. This approach is not viable for the IP dataset because the number of training samples in a category can be very small; for example, with 10% training samples, we only have two training samples in Oats category. Performance will be badly affected if the removed samples are from this category (see Figure 4a). The second approach is by copying other samples from the previous mini-batch. This technique will make some samples appear twice in the training process, and these samples will have more weight. Another approach is by dividing the training size over the intended batch number. For example, if we intend to have three batches so the batch size = training size/3. However, when the training sample is too large, the batch size will be large and thus prone to an out of memory problem. If the training size is too small, the batch size will also be small, having a tiny batch size can decrease the performance. Therefore, in our experiment, we used Equation (10) to compute the batch-size prior to the training process to prevent the occurrence of a tiny mini-batch, where s_b is the standard batch, tr is the training size, and th is the threshold (the allowed smallest mini-batch, see Figure 4b). We used $s_b = 256$ and $th = 64$ in this paper.

$$batch_{size} = \begin{cases} s_b, & \text{if } tr \bmod s_b > th. \\ s_b + \frac{tr \bmod s_b}{int(\frac{tr}{s_b})}, & \text{otherwise.} \end{cases} \tag{10}$$



(**a**) (**b**)

Figure 4. Example condition when the batch size cannot divide the training size evenly: (**a**) the latest mini-batch size is one, and (**b**) the latest mini-batch size is more than threshold (if the threshold is seven).

In the sRN, we used a 3×3 spectral cube and computed its mean instead of using a pixel vector directly to minimize the effect of spectral noise. In contrast to sRN, saRN focus is to get the spatial features; hence, the region size of the input cube gives an impact on the spatial feature representation.

In this research, in order to find the optimum saRN input region size, n, we experiment on a variable set of n {21, 23, 25, and 27} with the number of PCA components set to 30 by using 10% random training samples and repeat the experiment 10 times. Table 2 shows the results of the Overall Accuracy mean and standard deviation, and from this table, we can conclude that each dataset has a different optimum number n. For the PU, IP, and KSC dataset, the optimum n is 21, 25, and 27, respectively.

We then use the optimum value of n to find the optimum number of PCA components, K. We experiment with different size of K {25, 30, 35, and 40}. The Overall Accuracy (OA)-mean with different values of K and 10% training samples are shown in Table 3. The table shows that the optimum K of KSC dataset is 25, while, for the IP dataset and PU dataset, the optimum K is 35.

Table 2. Overall Accuracy of each dataset based on various patch sizes (SoP). The number in bold is the best Overall Accuracy.

SoP	21	23	25	27
IP	98.73 ± 0.22	98.66 ± 0.29	**98.77 ± 0.32**	98.75 ± 0.16
KSC	97.73 ± 0.47	97.95 ± 1.12	96.74 ± 1.43	**98.51 ± 0.31**
PU	**99.87 ± 0.06**	99.46 ± 0.02	99.65 ± 1.28	99.82 ± 0.39

Table 3. Overall Accuracy based on PCA number. The number in bold is the best Overall Accuracy.

nPCA	25	30	35	40
IP	98.74 ± 0.24	98.77 ± 0.32	**98.82 ± 0.38**	98.80 ± 0.15
KSC	**99.15 ± 0.18**	98.51 ± 0.31	98.29 ± 0.82	98.10 ± 0.88
PU	99.72 ± 0.50	99.87 ± 0.06	**99.91 ± 0.02**	99.72 ± 0.47

Given the optimal parameters for our proposed method, we perform two experiments to understand the impact of each module of our proposed architecture. The first is an experiment to discover the effect of using the mean in the sRN sub-network. Second, we perform an experiment to evaluate the performance of sRN, saRN, and our proposed architecture.

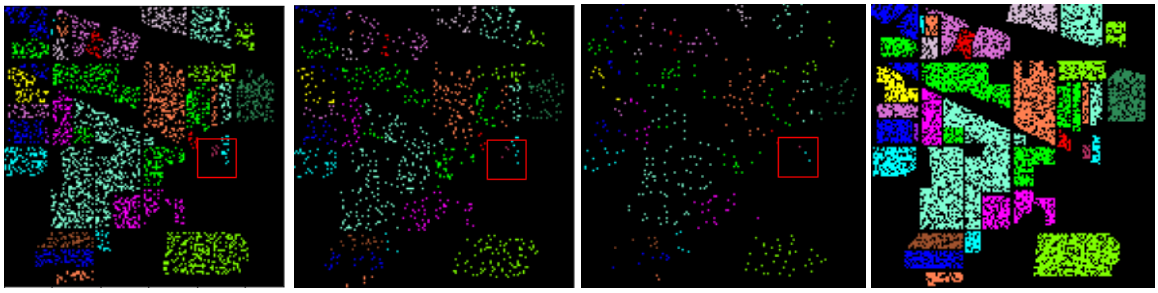
To demonstrate the effectiveness of our proposed method, we compare our method with the state-of-the-art architectures, which focus on exploring the spectral-spatial features of HSI, namely 3D-CNN [34], SSLSTMs [27], SSUN [23], spectral-spatial residual network (SSRN) [19], and hybrid spectral convolutional neural network (HybridSN) [61]. The SSLSTMs and the SSUN explore the spectral and the spatial features using two different streams, while the 3D-CNN, the SSRN, and the HybridSN extract features using a single stream network based on 3D convolutional layer. The implementation codes of the 3D-CNN (https://github.com/nshaud/DeepHyperX), the SSUN (https://github.com/YonghaoXu/SSUN), the SSRN (https://github.com/zilongzhong/SSRN), and the HybridSN (https://github.com/gokriznastic/HybridSN) are publicly available, letting us execute the codes to produce the classification results with all datasets. For the SSLSTMs, even though the implementation code is not accessible, we wrote the code based on their paper architecture and parameters. To confirm that our implemented code is correct, we tested it on 10% of the training dataset and verified our results with the work of Reference [27]. All experiments except the 3D-CNN were conducted on X299 UD4 Pro desktop computer with the GeForce RTX 2080 Ti Graphical Processing Unit (GPU). The experiment of the 3D-CNN was conducted on Google Colab server because 3D-CNN used the Pytorch framework.

To validate the performance of the proposed model with respect to the training size of each compared model, we performed three different experiments. In all of these experiments, we used 10-fold cross-validation. To guarantee that all of the techniques use the same training indices and testing indices, we created a module to generate the training indices and testing indices by using StratifiedShuffleSplit function available in Keras. The input of this function is the training size percentage and the number of the fold/group (k). The output is k fold training indices and testing indices, where each fold is made by preserving the percentage of samples for each class. We then

saved the training indices and testing indices of each fold in a file. Those files were read by each method during the experiment. Following the protocol in Reference [19], we use the same number of training epoch, 200, for all of the experiments. Regarding the hyperparameters, we used the optimum parameter of each model that has been provided in their respective paper. For the hyperparameters of this proposed approach, we used the optimum settings that have been optimized on 10% training samples, which were provided by Tables 2 and 3.

In conclusion, we divided the experiments into two groups. The first group (experiments 1 and 2) is an ablation analysis to understand the impact of using the mean, and concatenating sRN and saRN in the proposed method with respect to the overall performance accuracy. The second group (experiments 3, 4, and 5) are experiments to understand the effectiveness of the proposed method compared to other previous studies. The details of these experiments are as follows:

1. To evaluate the effect of the mean operation in the sRN sub-network input, we performed experiments on our proposed architecture with two case scenarios. First, the sRN input is the mean of a 3×3 spectral cube. Second, the sRN input is a spectral vector of a pixel x_{ij} without the mean operation. We performed experiments with 4%, 6%, 8%, 10%, and 30% training samples for IP dataset, PU dataset, and KSC dataset. We use the rest of the data that is not used in training for testing. In each run, we use the same data points for the training of both "with mean" and "without mean" setups.
2. To discover the concatenation effect of the sRN sub-network and the saRN sub-network on the performance accuracy, we performed experiments on three different architectures, the sRN network only, the saRN network only, and our proposed method with 30%, 10%, and 4% training samples. Here, we divided the data into a training set (30%) and a testing set (70%). Then, from the training set, we used all, one-third, and one-seventh point five for training. For testing, in all experiments in this part, we used all data in the testing set. The examples of train and test split on IP dataset with various percentage training samples are shown in Figure 5.



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

Figure 5. (**a**–**c**) The train split with 30%, 10%, and 4% training size on Indian Pine (IP) dataset (**d**) the test split.

3. In our third experiment, this proposed approach is compared to 3D-CNN, SSLSTMs, SSUN, SSRN, and HybridSN by using 10% training samples. We chose 10% training samples because the SSLSTMs and other experiments on SSUN, SSRN, and HybridSN have also been conducted using 10% training samples.
4. In our fourth experiment, we compared all of those methods on the smaller training samples, 4%, 6%, and 8%. Besides, because 3D-CNN has been tested using 4% training samples, the use of small labeled samples during training can be used to investigate over-fitting issues.
5. In the last experiment, we compared all of those methods on large training samples, 30%. Not only because HybridSN [61] had been tested on 30% training samples but also to investigate under-fitting issues with large training samples.

4.3. Experimental Results

Experiment 1: Table 4 shows the OA-mean and standard deviation of this proposed architecture in two different cases. In the first case, the sRN input of our network is a 3×3 cube followed by mean operations (with mean), and the second case, the sRN input of our network is a spectral vector, which was not followed by mean operations (without mean). From the table, we can see that, in 11 cases out of 15, the "with mean" slightly outperform the "without mean". We also found that, in 10 cases out of 15, the "with mean" is more stable than "without mean".

Table 4. Comparison between with mean and without mean in our proposed network (Bold represents the best results in the experiment setup).

Training Percentage	Indian Pines		Pavia University		KSC	
	with Mean	w\o Mean	with Mean	w\o Mean	with Mean	w\o Mean
4%	**95.40 ± 0.79**	95.07 ± 0.81	**99.85 ± 0.06**	99.62 ± 0.55	**96.97 ± 0.86**	95.32 ± 1.16
6%	**97.38 ± 0.58**	97.37 ± 0.63	99.44 ± 1.54	**99.93 ± 0.03**	**98.04 ± 0.65**	96.62 ± 1.46
8%	**98.24 ± 0.50**	98.14 ± 0.43	99.78 ± 0.55	**99.94 ± 0.05**	**99.36 ± 0.31**	99.02 ± 0.28
10%	98.70 ± 0.26	**98.81 ± 0.24**	**99.86 ± 0.27**	99.67 ± 0.74	**99.48 ± 0.34**	99.20 ± 0.42
30%	99.70 ± 0.10	**99.75 ± 0.15**	**99.89 ± 0.20**	99.03 ± 3.04	**99.96 ± 0.02**	99.93 ± 0.06

Experiment 2: Table 5 displays the OA-mean and standard deviation of sRN, saRN, and TSRN with various training samples, where the best performance is shown in bold. The table shows that with 30% training data, saRN's performance is slightly better than others. With 10% training samples, TSRN's performance starts to exceed saRN's performance. TSRN's superiority is clearly shown in 4% training samples. When the training size is large (30%), and the train and test sets are sampled randomly over the whole image, the possibility of the training samples become the testing samples' neighbor is high. Other spatial features, such as line and shape, are clear, too. See Figure 5a, suppose the center of the red window is the testing sample, we can easily predict its label by seeing its spatial features. However, with 10% training samples, predicting the pixel's label only by using its spatial features is slightly difficult (see Figure 5b). The prediction problems are more complicated when the training size is 4%. Figure 5c shows that the spatial features (e.g., neighborhood, shape, line) alone cannot perform well. Therefore, with 4% training samples, the TSRN, which also use spectral features, produces much better performance then saRN. Meanwhile, the low performance of sRN on IP dataset and KSC dataset probably because IP and KSC dataset have significantly low spatial resolution 20 m per pixel and 18 m per pixel, respectively. For example, in IP dataset, where most classes are vegetation, one pixel corresponds to the average reflectance of vegetation in 400 m^2, which results in a mixture of ground materials. As a consequence, classifying the objects based on spectral information only is difficult.

Table 5. Comparison between sRN, saRN, and Proposed (TSRN) with 30%, 10%, and 4% training samples (Bold represents the best results in the experiment setup).

Training	30%			10%			4%		
Dataset	sRN	saRN	TSRN	sRN	saRN	TSRN	sRN	saRN	TSRN
IP	92.16 ± 0.66	**99.75 ± 0.18**	99.69 ± 0.15	86.61 ± 0.86	98.44 ± 0.26	**99.03 ± 0.24**	79.94 ± 1.54	94.20 ± 0.43	**95.50 ± 0.87**
PU	98.96 ± 0.08	**99.97 ± 0.02**	99.95 ± 0.13	98.05 ± 0.19	99.84 ± 0.05	**99.93 ± 0.10**	96.68 ± 0.14	99.56 ± 0.11	**99.82 ± 0.10**
KSC	95.55 ± 0.77	**100 ± 0.02**	99.95 ± 0.05	92.52 ± 0.86	**99.55 ± 0.2**	99.52 ± 0.29	82.42 ± 3.48	94.60 ± 1.25	**95.62 ± 1.27**

Experiment 3: Tables 6–8 show the quantitative evaluations of those compared models with 10% training samples. The tables present three generally used quantitative metrics, i.e., Overall Accuracy (OA), Average Accuracy (AA), Kappa coefficient (K), and the classification accuracy of each class. The first three rows show the OA, AA, and K of each method. The following rows show the classification accuracy of each class. The numbers indicate the mean, followed by the standard deviation of each evaluation with a 10-fold cross-validation. The bold, the underlined, and the

italic numbers represent the first-best performance, the second-best, and the third-best performance, respectively. Subsequently, Figures 6–8 display the false-color image, the ground-truth image, and the classification map of each method on Indian Pine, Pavia University and KSC datasets.

Table 6. Overall Accuracy, Average Accuracy, Kappa Value, and Class Wise Accuracy of our proposed method versus other methods on IP dataset when using 10% training samples. The best performance is in bold, the second-best performance is underlined, and the third-best is in italic.

Label	3D-CNN [34]	SSLSTMs [27]	SSUN [23]	SSRN [19]	HybridSN [61]	Proposed
OA	85.29 ± 7.24	94.65 ± 0.72	96.79 ± 0.36	98.24 ± 0.29	*97.36 ± 0.82*	**98.70 ± 0.25**
AA	81.11 ± 0.12	94.47 ± 1.77	95.78 ± 2.97	91.69 ± 3.46	*95.70 ± 1.08*	**98.71 ± 0.61**
K × 100%	83.20 ± 0.08	93.89 ± 0.82	96.33 ± 0.41	97.99 ± 0.32	*96.99 ± 0.93*	**98.52 ± 0.28**
C1	68.70 ± 0.28	*99.32 ± 2.05*	99.51 ± 0.98	**100 ± 0**	97.80 ± 3.53	98.72 ± 3.83
C2	84.30 ± 0.17	93.09 ± 1.71	94.65 ± 1.33	**98.63 ± 0.82**	*96.76 ± 1.44*	98.02 ± 1.08
C3	75.50 ± 0.11	86.37 ± 1.81	96.09 ± 1.87	96.82 ± 0.70	*96.27 ± 2.58*	**97.08 ± 1.72**
C4	74.50 ± 0.09	89.35 ± 4.41	94.65 ± 4.90	**99.20 ± 1.37**	*96.67 ± 2.44*	99.16 ± 1.22
C5	88.80 ± 0.07	93.69 ± 3.10	*94.89 ± 2.09*	96.95 ± 2.27	94.57 ± 3.93	**99.41 ± 0.36**
C6	96.00 ± 0.02	95.46 ± 0.92	99.06 ± 0.78	98.19 ± 1.17	*98.48 ± 0.70*	**99.76 ± 0.26**
C7	61.90 ± 0.27	**99.22 ± 1.57**	*93.20 ± 4.75*	70 ± 45.83	88.80 ± 13.24	99.15 ± 1.71
C8	96.80 ± 0.02	98.16 ± 1.73	99.81 ± 0.49	*99.15 ± 2.13*	99.81 ± 0.39	**99.98 ± 0.07**
C9	60.00 ± 0.24	90.70 ± 19.01	*86.67 ± 13.19*	20.00 ± 40.00	86.11 ± 14.33	**98.50 ± 4.50**
C10	82.70 ± 0.08	96.50 ± 1.42	95.18 ± 1.92	97.35 ± 1.55	*97.34 ± 1.29*	**98.19 ± 1.10**
C11	87.30 ± 0.07	96.73 ± 0.85	98.13 ± 0.38	98.65 ± 0.40	*98.22 ± 0.89*	**99.32 ± 0.46**
C12	77.40 ± 0.11	88.92 ± 1.58	92.85 ± 4.39	95.51 ± 1.28	*93.82 ± 2.73*	**97.87 ± 1.82**
C13	97.70 ± 0.02	93.83 ± 4.27	**99.57 ± 0.22**	98.21 ± 2.38	99.02 ± 0.94	*98.76 ± 2.15*
C14	95.30 ± 0.03	98.07 ± 1.03	98.66 ± 0.54	**99.57 ± 0.42**	99.39 ± 0.40	*98.98 ± 0.91*
C15	69.30 ± 0.05	96.63 ± 3.98	*97.32 ± 2.91*	**99.34 ± 0.82**	95.48 ± 2.81	98.44 ± 1.74
C16	81.50 ± 0.28	*95.48 ± 4.10*	92.26 ± 6.74	**99.48 ± 0.85**	92.74 ± 4.95	98.11 ± 1.20

Table 7. Overall Accuracy, Average Accuracy, Kappa Value, and Class Wise Accuracy of our proposed method versus other methods on Pavia University (PU) dataset with 10% training samples. The best performance is in bold, the second-best performance is underlined, and the third-best is in italic.

Label	3D-CNN [34]	SSLSTMs [27]	SSUN [23]	SSRN [19]	HybridSN [61]	Proposed
OA	94.07 ± 0.86	98.58±0.23	99.53 ± 0.09	*99.59 ± 0.72*	99.73 ± 0.11	**99.86 ± 0.26**
AA	96.54 ± 0.01	98.65 ± 0.16	99.18 ± 29.9	*99.31 ± 1.48*	99.43 ± 0.23	**99.77 ± 0.54**
K × 100%	92.30 ± 0.01	98.11 ± 0.31	99.38 ± 0.12	*99.46 ± 0.95*	99.65 ± 0.15	**99.82 ± 0.34**
C1	96.50 ± 0.01	97.47 ± 0.46	99.29 ± 0.22	*99.85 ± 0.17*	99.91 ± 0.18	**99.94 ± 0.07**
C2	95.20 ± 0.01	98.95 ± 0.34	99.91 ± 0.04	*99.93 ± 0.07*	**100 ± 0.01**	99.94 ± 0.06
C3	92.20 ± 0.03	*98.80 ± 0.59*	97.67 ± 0.74	**99.56 ± 0.56**	99.22 ± 0.81	98.56 ± 4.09
C4	97.20 ± 0.01	98.43 ± 0.46	*99.36 ± 0.29*	99.66 ± 0.33	98.4 ± 0.96	**99.99 ± 0.03**
C5	99.90 ± 0	99.87 ± 0.15	99.93 ± 0.07	*99.91 ± 0.08*	**99.97 ± 0.06**	99.84 ± 0.10
C6	97.60 ± 0.02	98.92 ± 0.51	*99.89 ± 0.12*	99.99 ± 0.05	99.99 ± 0.01	**100 ± 0**
C7	95.50 ± 0.02	*98.36 ± 1.03*	97.87 ± 1.64	95.67 ± 12.94	**100 ± 0**	99.73 ± 0.75
C8	95.40 ± 0.02	97.67 ± 0.60	99.19 ± 0.36	*99.28 ± 1.11*	99.41 ± 0.45	**99.92 ± 0.08**
C9	99.40 ± 0.01	99.44 ± 0.50	*99.54 ± 0.35*	99.92 ± 0.12	97.98 ± 0.95	**100 ± 0**

Table 8. Overall Accuracy, Average Accuracy, Kappa Value, and Class Wise Accuracy of our proposed method versus other methods on Kennedy Space Center (KSC) dataset with 10% training samples. The best performance is in bold, the second-best performance is underlined, and the third-best is in italic.

Label	3D-CNN [34]	SSLSTMs [27]	SSUN [23]	SSRN [19]	HybridSN [61]	Proposed
OA	82.21 ± 2.96	*97.51 ± 0.63*	96.22 ± 0.86	98.77 ± 0.75	91.72 ± 1.52	**99.48 ± 0.32**
AA	71.68 ± 0.12	*97.36 ± 0.69*	94.65 ± 2.87	98.10 ± 1.13	88.94± 1.53	**99.04 ± 0.46**
K × 100%	80.20 ± 0.03	*97.22 ± 0.71*	95.79 ± 0.96	98.63 ± 0.83	90.77 ± 1.69	**99.42 ± 0.36**
C1	92.40 ± 0.03	*96.65 ± 1.73*	97.12 ± 0.98	**100 ± 0**	95.68 ± 4.17	**100 ± 0**
C2	84.20 ± 0.08	97.57 ± 2.39	94.25 ± 4.04	*96.90 ± 7.10*	76.99 ± 5.48	**99.77 ± 0.43**
C3	43.00 ± 0.27	*96.75 ± 2.76*	95.05 ± 3.41	**100 ± 0**	90.35 ± 3.60	97.17 ± 4.41
C4	33.50 ± 0.16	**98.41 ± 1.38**	*89.08 ± 5.15*	88.86 ± 11.12	70.40 ± 5.30	98.39 ± 2.69
C5	34.70 +- 0.19	97.55 ± 2.52	89.93 ± 8.78	96.66 ± 5.93	*97.24 ± 3.26*	**97.91 ± 4.81**
C6	40.90 ± 0.22	*97.82 ± 2.84*	79.56 ± 3.851	**99.90 ± 0.30**	81.89 ± 6.60	99.19 ± 1.42
C7	59.30 ± 0.31	96.34 ± 4.93	**98.19 ± 1.43**	93.56 ± 10.79	82.23 ± 6.73	*96.02 ± 4.26*
C8	75.60 ± 0.08	*95.27 ± 2.83*	93.61 ± 2.73	99.54 ± 0.76	93.40 ± 3.81	**99.62 ± 0.35**
C9	84.10 ± 0.09	96.93 ± 2.34	*98.25 ± 2.60*	**100 ± 0**	88.29 ± 4.40	99.62 ± 0.42
C10	94.40 ± 0.04	*97.28 ± 3.23*	96.95 ± 2.45	**100 ± 0**	90.08 ± 4.19	99.89 ± 0.33
C11	97.70 ± 0.02	98.21 ± 1.35	*98.86 ± 1.11*	**99.92 ± 0.24**	97.19 ± 2.55	99.90 ± 0.32
C12	92.10 ± 0.02	97.06 ± 1.52	*99.56 ± 0.83*	99.93 ± 0.20	92.50 ± 3.78	**100 ± 0**
C13	99.90 ± 0	*99.88 ± 0.14*	**100 ± 0**	**100 ± 0**	**100 ± 0**	**100 ± 0**

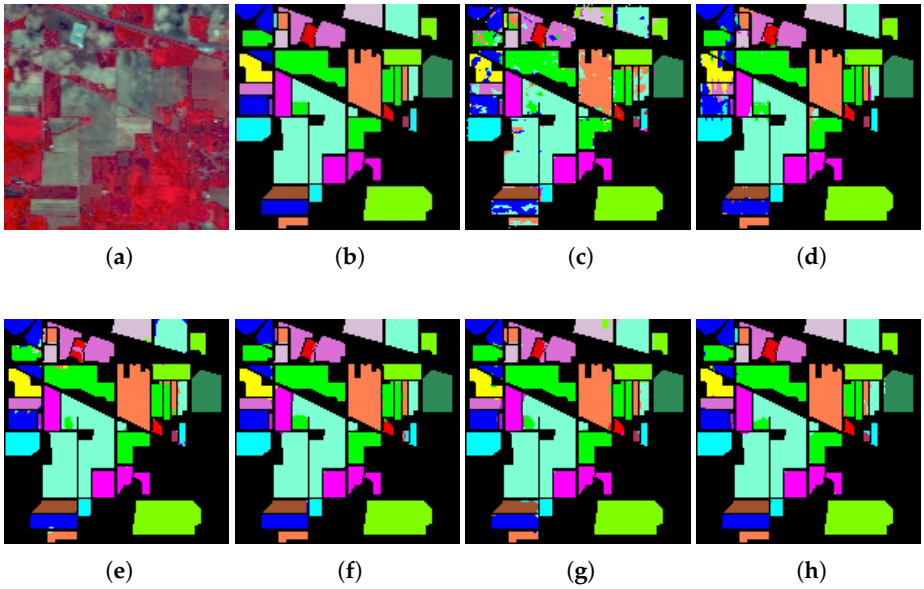


Figure 6. The classification map of IP dataset. (**a**) False color image, (**b**) Ground truth, and (**c**–**h**) Prediction classification maps of 3D-Convolutional Neural Network (CNN) (85.29%), spectral-spatial long short-term memory (SSLSTMs) (95%), spectral-spatial unified network (SSUN) (97.24%), SSRN (98.29%), HybridSN (97.38%), and our proposed architecture (98.69%).

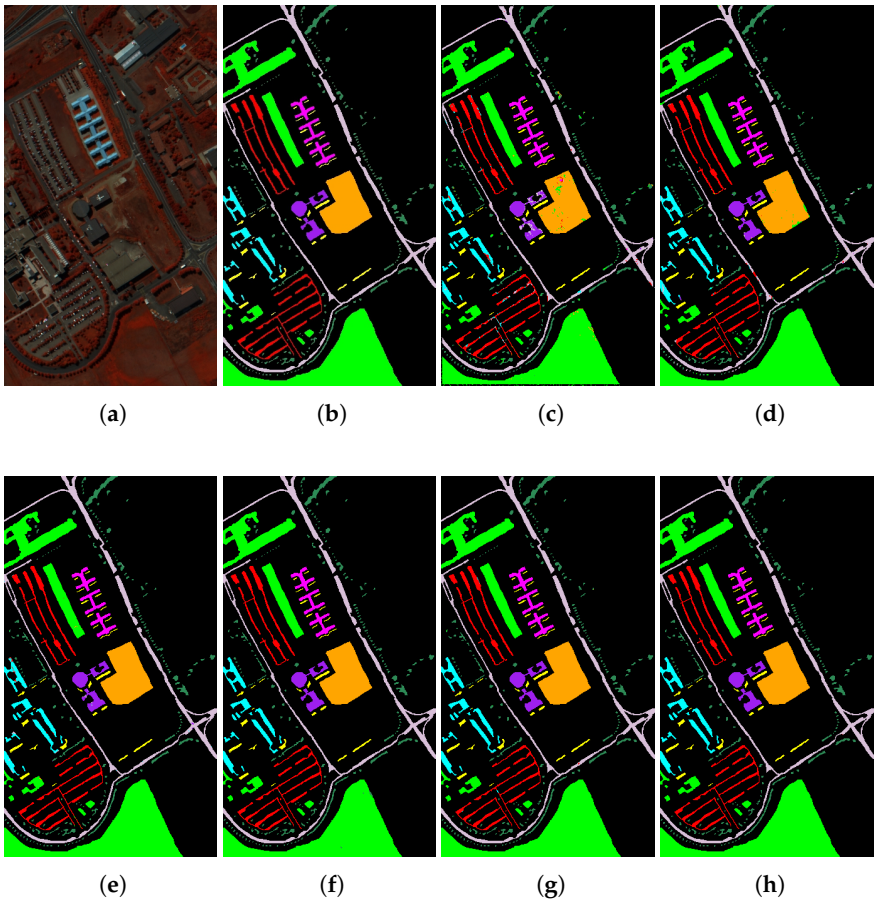


Figure 7. The classification map of Pavia University dataset. (**a**) False color image, (**b**) Ground truth, (**c**–**h**) Prediction classification maps of 3D-CNN (94.07%), SSLSTMs (98.50%), SSUN (99.52%), SSRN (99.88%), HybridSN (99.85%), and our proposed architecture (99.94%).

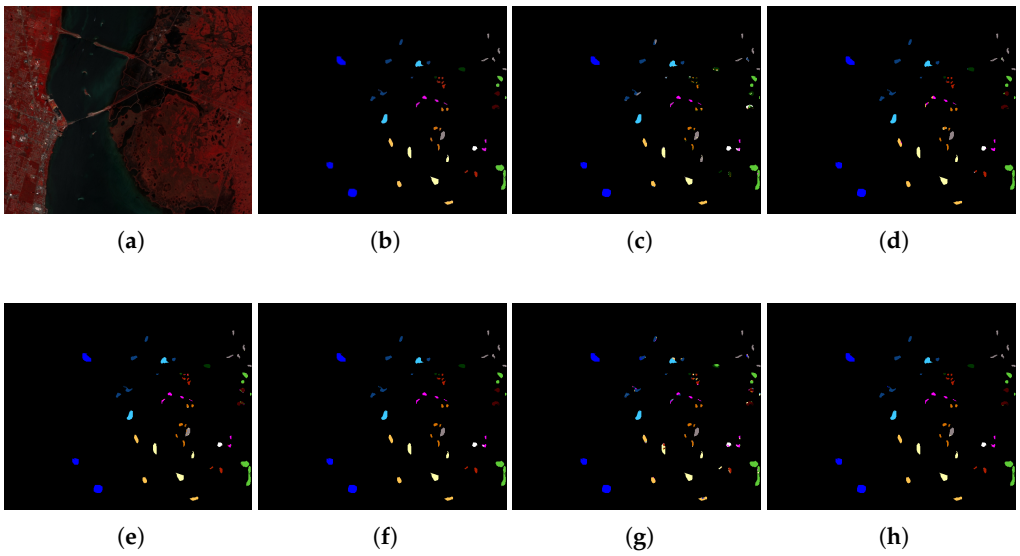
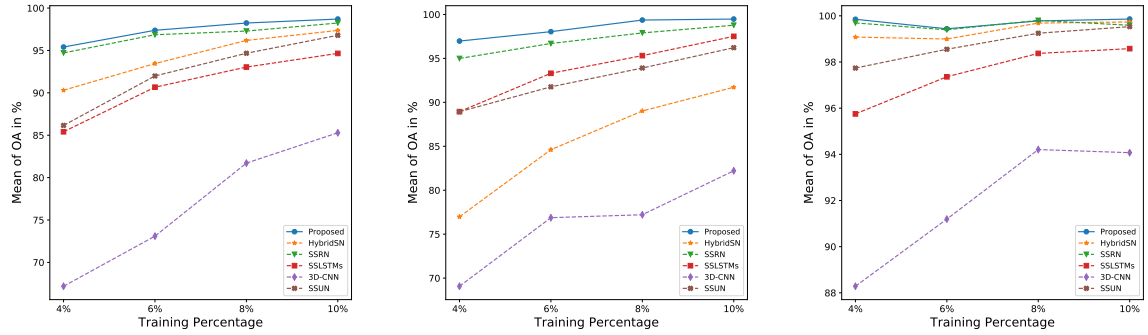


Figure 8. The classification map of KSC dataset. (**a**) False color image, (**b**) Ground truth, (**c**–**h**) Prediction classification maps of 3D-CNN (82.21%), SSLSTMs (97%), SSUN (97.10%), SSRN (99.27%), HybridSN (87.46%), and our proposed architecture (99.61%).

Experiment 4: Figure 9 presents the graphic of AO-mean obtained from our fifth experiment, where all of those methods are trained on smaller training samples 4%, 6%, and 8%. In the figure, we include the results of our first experiment, where those methods are trained on the 10% samples. The performances of all of the compared methods are displayed using a dotted line, while our proposed method is displayed with a solid line.



(**a**) (**b**) (**c**)

Figure 9. Overall accuracy of each method for different training data sizes of: (**a**) Indian Pine dataset, (**b**) KSC dataset, and (**c**) Pavia University dataset.

Experiment 5: Tables 9–11 show the OA, AA, and K of each method with 30% training samples. On large training samples, almost all of the compared methods produce a high accuracy. The difference is small. Hence, in the table, we report the comparison on each fold for a more detailed comparison. The bold numbers are the best accuracies produced by these methods.

Table 9. Fold Overall Accuracy, Average Accuracy, and Kappa Value on IP dataset with 30% training data. The best performance is in bold, the second-best performance is underlined, and the third-best is in italic.

Fold	1	2	3	4	5	6	7	8	9	10	OA-Mean and Std
						OA					
Proposed	**99.80**	99.71	**99.61**	**99.72**	99.57	*99.54*	99.69	**99.85**	99.68	**99.79**	**99.70 ± 0.10**
HybridSN	*99.53*	99.71	99.57	99.61	**99.68**	**99.75**	**99.83**	*99.53*	*99.67*	99.64	99.65 ± 0.09
SSRN	98.89	**99.79**	99.53	99.26	99.50	99.68	99.04	99.33	99.46	99.48	99.40 ± 0.26
SSUN	99.60	99.48	99.57	*99.46*	99.57	99.46	99.67	99.61	**99.74**	*99.55*	*99.57 ± 0.09*
SSLSTMs	99.16	99.19	99.07	99.15	98.94	99.23	97.09	98.91	99.4	99.32	98.95 ± 0.64
3D-CNN	95.15	95.89	92.42	95.47	94.51	95.55	95.86	95.26	95.41	95.33	95.09 ± 0.96
						AA					
Proposed	**99.89**	**99.66**	**99.68**	**99.73**	99.61	99.58	**99.77**	**99.82**	**99.65**	**99.58**	**99.70 ± 0.10**
HybridSN	*99.04*	*99.50*	*98.82*	99.44	**99.62**	**99.69**	99.71	*99.03*	99.61	99.07	*99.35 ± 0.31*
SSRN	80.29	99.64	92.98	91.55	93.18	93.20	86.22	91.96	92.79	97.99	91.98 ± 5.19
SSUN	99.60	99.14	99.47	99.26	*99.34*	*99.50*	*99.24*	99.61	99.50	99.53	99.42 ± 0.15
SSLSTMs	*99.04*	99.12	98.26	*99.34*	98.77	97.10	97.66	98.65	*99.55*	*99.31*	98.68 ± 0.75
3D-CNN	96.57	97.33	94.04	95.42	95.20	96.75	97.02	96.81	96.57	96.01	96.17 ± 0.96
						K × 100%					
Proposed	**99.78**	99.67	**99.56**	**99.68**	99.51	*99.48*	99.65	**99.83**	99.63	**99.76**	**99.66 ± 0.11**
HybridSN	*99.46*	99.67	99.51	99.56	**99.63**	**99.71**	**99.81**	*99.46*	*99.62*	99.59	99.60 ± 0.11
SSRN	98.73	**99.76**	99.46	99.16	99.43	99.63	98.90	99.24	99.38	99.41	99.31 ± 0.30
SSUN	99.54	99.41	99.51	*99.38*	99.51	99.38	*99.62*	99.56	**99.70**	*99.49*	*99.51 ± 0.10*
SSLSTMs	99.05	99.08	98.94	99.03	98.79	99.13	96.68	98.76	99.32	99.22	98.80 ± 0.73
3D-CNN	94.49	95.33	91.40	94.85	93.75	94.95	95.29	94.61	94.78	94.69	94.41 ± 1.09

Table 10. Fold Overall Accuracy, Average Accuracy, and Kappa Value on PU dataset with 30% training data. The best performance is in bold, the second-best performance is underlined, and the third-best is in italic.

Fold	1	2	3	4	5	6	7	8	9	10	OA-Mean and Std
						OA					
Proposed	99.96	**99.96**	**99.98**	**99.98**	**99.99**	99.84	99.34	**99.99**	99.92	*99.95*	*99.89 ± 0.19*
HybridSN	**99.97**	**99.96**	99.97	99.96	**99.99**	**100**	**99.96**	99.97	99.92	**99.98**	**99.97 ± 0.02**
SSRN	99.93	99.83	99.83	*99.92*	99.81	*99.89*	*99.82*	99.86	97.44	99.58	99.59 ± 0.72
SSUN	*99.94*	99.95	*99.92*	99.91	99.91	99.94	99.9	*99.91*	**99.94**	99.96	99.93 ± 0.02
SSLSTMs	99.85	99.80	99.83	99.76	99.72	99.79	99.68	99.86	99.70	99.82	99.78 ± 0.06
3D-CNN	95.76	95.80	95.80	95.30	95.75	95.76	94.75	95.94	95.92	94.06	95.48 ± 0.58
						AA					
Proposed	99.96	**99.94**	99.94	**99.97**	**100**	99.77	98.70	**99.99**	**99.95**	**99.97**	*99.82 ± 0.38*
HybridSN	**99.97**	99.93	**99.95**	99.94	99.97	**100**	**99.93**	99.95	*99.81*	99.96	**99.94 ± 0.05**
SSRN	**99.93**	99.74	*99.85*	*99.86*	99.81	99.92	99.84	99.74	94.88	99.48	99.31 ± 1.48
SSUN	99.90	*99.91*	99.81	99.84	*99.82*	99.92	*99.82*	99.79	99.87	*99.94*	99.86 ± 0.05
SSLSTMs	99.88	99.74	99.81	99.79	99.68	99.73	99.66	*99.86*	99.67	99.85	99.77 ± 0.08
3D-CNN	97.46	97.69	97.54	97.00	97.68	97.54	96.91	97.77	97.79	94.80	97.22 ± 0.86
						K × 100%					
Proposed	99.95	**99.94**	**99.97**	**99.98**	**99.99**	99.78	99.12	**99.98**	99.90	*99.93*	*99.85 ± 0.25*
HybridSN	**99.96**	**99.94**	99.96	99.95	99.98	**100**	**99.94**	99.96	99.90	**99.97**	**99.96 ± 0.03**
SSRN	99.91	99.78	99.77	*99.90*	99.75	*99.85*	*99.77*	99.81	96.62	99.45	99.46 ± 0.95
SSUN	*99.92*	99.93	*99.89*	99.88	*99.88*	99.92	99.87	*99.88*	**99.92**	99.95	99.90 ± 0.03
SSLSTMs	99.80	99.74	99.77	99.68	99.63	99.72	99.58	99.82	99.60	99.77	99.71 ± 0.08
3D-CNN	94.46	94.50	94.51	93.87	94.45	94.47	93.19	94.69	94.66	92.25	94.11 ± 0.75

Table 11. Fold Overall Accuracy, Average Accuracy, and Kappa Value on KSC dataset with 30% training data. The best performance is in bold, the second-best performance is underlined, and the third-best is in italic.

Fold	1	2	3	4	5	6	7	8	9	10	OA-Mean and Std
						OA					
Proposed	**99.95**	99.97	**100**	99.95	**99.97**	**99.97**	**99.92**	99.95	99.92	**99.95**	**99.96 ± 0.02**
HybridSN	99.01	99.37	99.23	99.48	98.96	98.93	98.85	98.96	*99.59*	99.26	99.16 ± 0.24
SSRN	*99.67*	**100**	**100**	**100**	*99.84*	99.37	98.27	**100**	**99.97**	*99.64*	99.68 ± 0.51
SSUN	99.10	99.31	99.67	98.68	99.56	*99.31*	*99.29*	99.29	99.01	99.40	99.26 ± 0.27
SSLSTMs	99.78	*99.95*	*99.56*	*99.73*	99.95	99.10	99.78	*99.42*	99.26	99.78	*99.63 ± 0.27*
3D-CNN	94.60	95.39	94.76	95.34	91.89	94.24	94.57	95.45	94.46	93.15	94.39 ± 1.05
						AA					
Proposed	**99.95**	99.97	**100**	99.96	**99.97**	**99.97**	**99.90**	99.95	99.92	**99.96**	**99.96 ± 0.03**
HybridSN	98.36	99.18	98.63	99.17	98.33	98.31	98.38	98.42	*99.35*	98.92	98.71 ± 0.39
SSRN	*99.51*	**100**	**100**	**100**	*99.75*	99.09	97.97	**100**	**99.96**	*99.44*	*99.57 ± 0.61*
SSUN	98.25	98.73	*99.27*	97.59	99.28	98.61	*98.43*	98.98	98.26	99.03	98.64 ± 0.50
SSLSTMs	99.74	*99.85*	99.44	*99.62*	99.95	*98.95*	99.66	*99.49*	99.34	99.78	99.58 ± 0.28
3D-CNN	92.57	93.37	92.01	92.70	87.23	91.71	91.82	93.09	91.58	89.46	91.55 ± 1.77
						K × 100%					
Proposed	**99.94**	99.97	**100**	99.94	**99.97**	**99.97**	**99.91**	99.94	99.91	**99.94**	**99.95 ± 0.03**
HybridSN	98.90	99.30	99.15	99.42	98.84	98.81	98.72	98.84	99.54	99.18	99.07 ± 0.27
SSRN	*99.63*	**100**	**100**	**100**	*99.82*	99.30	98.08	**100**	**99.97**	*99.60*	99.64 ± 0.57
SSUN	98.99	99.24	99.63	98.53	99.51	*99.24*	*99.21*	99.21	98.90	99.33	99.18 ± 0.30
SSLSTMs	99.76	*99.94*	*99.51*	*99.69*	99.94	98.99	99.76	*99.36*	99.18	99.76	*99.59 ± 0.30*
3D-CNN	93.99	94.88	94.18	94.81	90.98	93.60	93.97	94.94	93.85	92.39	93.76 ± 1.17

4.4. Discussion

According to the highlighted results in Section 4.3, one of the first apparent points is that the proposed method is able to produce a high performance for all ranges of training sizes (4%, 6%, 8%, 10%, and 30% training samples). Its OA, AA, and K values are higher compared to 3D-CNN, SSLSTMs, SSUN, SSRN, and HybridSN. The differences get higher as the training sample size is

reduced. With large training samples, e.g., 30% training samples, the performances of these methods are similar.

The quantitative evaluation of those models with 10% training samples are reported in Tables 6–8. These tables show three standard quantitative metrics, i.e., OA, AA, K; and the classification accuracy of each class. More specifically, on Indian Pines dataset (see Table 6), in which class sizes are imbalanced, our proposed method produces the highest OA, AA, and K value, and the proposed approach yields OA 0.46% higher than the second-best method, SSRN. Considering the AA, the difference between the proposed architecture and SSRN is much higher, more than 7%. From Table 6, we can see that TSRN tries to optimize the recognition of each class even though the number of instances in the class is tiny. Hence, it achieves a high accuracy compared to the other methods when classifying C9 (Oats), in which number of instances is 20, which means C9 training samples is 2. For more detailed classification accuracy of each class, TSRN yields the best recognition on eight classes out of 16 classes. Its recognition for the other five classes and two classes are the second- and third-best, respectively.

The results are consistent on the Pavia University dataset, in which characteristics are different from the Indian Pine dataset. In the PU dataset, the number of data for each class is large, with the minimum number of instances on Shadows category equal to 947. As shown in Table 7, our proposed method attains the best OA, AA, and K compared to the other architectures, albeit insignificant disparity. The small gap between TSRN and the second-best method, HybridSN, shows that those methods are very competitive for large training samples. For class recognition, the proposed method achieves the highest accuracy on five out of nine classes in the PU dataset, with an improvement of less than 1%.

In contrast to the IP and PU datasets, the total number of instances of KSC dataset is relatively small. From Table 8, we can see that our proposed approach achieves the best performance. Its OA, AA, and K is $\pm$ 0.71, $\pm$ 0.94, and $\pm$ 0.79 higher compared to the second-best method, SSRN. In contrast, HybridSN yields performance that is not as good as its performance on IP and PU dataset.

The comparison between the proposed architecture and other methods on smaller training samples for IP, KSC, and PU dataset is demonstrated in Figure 9a–c, respectively. These figures reveal that the proposed method achieves the best accuracy even with smaller training samples. The accuracy gap between our method and the second-best method is high on KSC dataset. With 4% training data, our method achieves OA $\pm$ 2% higher than the second-best method, SSRN. The difference is smaller on IP dataset and is extremely small on PU dataset. The reason is that the size of the KSC dataset is the smallest compared to other datasets. Four percent training samples in the experiment correspond to 208 training instances on KSC dataset, 410 instances on IP dataset, and 1711 samples on PU dataset.

The performance of TSRN and the other methods with larger training samples, 30%, is shown in Tables 9–11 for IP dataset, PU dataset, and KSC dataset, respectively. In IP dataset, out of ten folds, the proposed method achieves the best OA and K on five-folds, and the best AA on eight folds. Our method also outperforms HybridSN, which presents the best OA on three folds. In PU dataset (see Table 10), the HybridSN shows a slightly better OA than the proposed architecture. HybridSN produces the best OA on six-folds while TSRN produces the best performance within five-folds. In the 2nd fold and 5th fold, their OA is precisely the same. Regarding AA, the proposed method achieves the best AA on six-folds when HybridSN achieves the best AA on four-folds. In terms of K, those two methods yield the best value on five-folds. The same with the result from IP dataset, with KSC dataset, our proposed approach also produces the best performance or the second-best performance in each fold (see Table 11). From these results, we can conclude that on large training samples, those approaches, i.e., TSRN, SSUN, HybridSN, SSRN, SSLSTMs, are very competitive.

Table 12 presents the number of parameters, the model size, the depth, the training time, and the testing time of the other methods with 10% training samples from IP dataset. We do not report the time complexity of 3D-CNN because 3D-CNN was tested on a different machine. However, Reference [61] has shown that 3D-CNN is less efficient compared to HybridSN. Moreover, from Table 12, we perceive that the proposed method is more efficient than HybridSN. In other words, we can conclude that TSRN

is much more efficient than 3D-CNN. Compared to 3D convolution-based SSRN [19] and (3D + 2D) convolution-based HybridSN [61], our proposed network has a faster training time and fewer learning parameters (Table 12). Our network model size, in which depth is 24, is 3.3 MB. The size is smaller compared to 61.5 MB of a 7-layer HybridSN and more effective compared to 2.9 MB of a 9-layer SSRN. Note that an increase in the network depth results in a model size increase. From Table 12, we can see that our network, which uses (2D+1D) convolution, can be deeper without increasing the number of parameters by a large number. Such a deeper network can extract richer features. On the other hand, for 3D convolution, the model size and the number of parameters will grow dramatically as the network becomes deeper. As a result, training on very deep 3D-CNN becomes challenging with the risk of overfitting. Our network yields a smaller number of learnable parameters, making it less prone to the overfitting problem especially when small samples are used for training.

Table 12. Number of parameters, model size, depth, training time, and testing time on IP dataset on different methods with 10% training samples.

Method	SSLSTMs	SSUN	SSRN	HybridSN	Proposed
# parameters	343,072	949,648	346,784	5,122,176	239,672
Model Size	6.7 MB	9.6 MB	2.9 MB	61.5 MB	3.3 MB
Depth	4	10	9	7	24
Training Time	300 s	66 s	150 s	120 s	60 s
Testing Time	5.3 s	3.03 s	4.1 s	2.57 s	3.1 s

5. Conclusions

The paper presents a novel two-streams residual network architecture for the classification of hyperspectral data. Our network improves the spectral and the spatial feature extraction by applying a full pre-activation sRN and saRN separately. These two networks are similar in their structure but use a different type of convolutional layer. The convolutional layer of sRN is based on 1D convolution, which best fits the spectral data structure, while the saRN is based on 2D convolution, which best fits the spatial data structure of HSI.

Our experiments were conducted on three well-known hyperspectral datasets, versus five different methods, as well as various sizes of training samples. One of the main conclusions that arises from our experiments is that our proposed method can provide a higher performance versus state-of-the-art classification methods, even with various training samples proportion from 4% training samples up to 30% training samples. The high accuracy of our proposed method on small training samples, 4%, shows that this method does not overfit. Otherwise, the competitive accuracy of our proposed method with large training samples, 30%, explains that this architecture is not under-fitting either.

Author Contributions: W.N.K. proposed the methodology, implemented it, did experiments and analysis, and wrote the original draft manuscript. M.B., F.B., and F.S. supervised the study, directly contributed to the problem formulation, experimental design and technical discussions, reviewed the writing, and gave insightful suggestions for the manuscript. D.E. supervised the study, reviewed the writing, gave insightful suggestions for the manuscript and provided resources. All authors have read and agreed to the published version of the manuscript.

Funding: W.N. Khotimah is supported by a scholarship from Indonesian Endowment Fund for Education, Ministry of Finance, Indonesia. This research is also supported by Australia Research Council Grants DP150100294, DP150104251, and Grains Research and Development Corporation Grant UWA2002-003RTX.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Lu, G.; Fei, B. Medical hyperspectral imaging: a review. *J. Biomed. Opt.* **2014**, *19*, 010901. [CrossRef] [PubMed]
2. Wang, L.; Sun, C.; Fu, Y.; Kim, M.H.; Huang, H. Hyperspectral Image Reconstruction Using a Deep Spatial-Spectral Prior. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
3. Shimoni, M.; Haelterman, R.; Perneel, C. Hypersectral Imaging for Military and Security Applications: Combining Myriad Processing and Sensing Techniques. *IEEE Geosci. Remote Sens. Mag.* **2019**, *7*, 101–117. [CrossRef]
4. Stuart, M.B.; McGonigle, A.J.S.; Willmott, J.R. Hyperspectral Imaging in Environmental Monitoring: A Review of Recent Developments and Technological Advances in Compact Field Deployable Systems. *Sensors* **2019**, *19*, 3071. [CrossRef] [PubMed]
5. Ma, J.; Sun, D.W.; Pu, H.; Cheng, J.H.; Wei, Q. Advanced Techniques for Hyperspectral Imaging in the Food Industry: Principles and Recent Applications. *Annu. Rev. Food Sci. Technol.* **2019**, *10*, 197–220. [CrossRef] [PubMed]
6. Mahlein, A.K.; Kuska, M.; Behmann, J.; Polder, G.; Walter, A. Hyperspectral Sensors and Imaging Technologies in Phytopathology: State of the Art. *Annu. Rev. Phytopathol.* **2018**, *56*, 535–558. [CrossRef] [PubMed]
7. Li, S.; Wu, H.; Wan, D.; Zhu, J. An effective feature selection method for hyperspectral image classification based on genetic algorithm and support vector machine. *Knowl. Based Syst.* **2011**, *24*, 40–48. [CrossRef]
8. Cao, F.; Yang, Z.; Ren, J.; Ling, W.K.; Zhao, H.; Sun, M.; Benediktsson, J.A. Sparse representation-based augmented multinomial logistic extreme learning machine with weighted composite features for spectral–spatial classification of hyperspectral images. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 6263–6279. [CrossRef]
9. Li, S.; Song, W.; Fang, L.; Chen, Y.; Ghamisi, P.; Benediktsson, J.A. Deep Learning for Hyperspectral Image Classification: An Overview. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 6690–6709. [CrossRef]
10. Shah, S.A.A.; Bennamoun, M.; Boussaïd, F. Iterative deep learning for image set based face and object recognition. *Neurocomputing* **2016**, *174*, 866–874. [CrossRef]
11. Hu, W.; Huang, Y.; Wei, L.; Zhang, F.; Li, H. Deep Convolutional Neural Networks for Hyperspectral Image Classification. *J. Sens.* **2015**, *2015*, 1–12. [CrossRef]
12. Haut, J.M.; Paoletti, M.E.; Plaza, J.; Li, J.; Plaza, A. Active learning with convolutional neural networks for hyperspectral image classification using a new bayesian approach. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 6440–6461. [CrossRef]
13. 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]
14. Zhan, Y.; Hu, D.; Wang, Y.; Yu, X. Semisupervised Hyperspectral Image Classification Based on Generative Adversarial Networks. *IEEE Geosci. Remote Sens. Lett.* **2018**, *15*, 212–216. [CrossRef]
15. Angelopoulou, T.; Tziolas, N.; Balafoutis, A.; Zalidis, G.; Bochtis, D. Remote Sensing Techniques for Soil Organic Carbon Estimation: A Review. *Remote Sens.* **2019**, *11*, 676. [CrossRef]
16. Yang, X.; Ye, Y.; Li, X.; Lau, R.Y.; Zhang, X.; Huang, X. Hyperspectral image classification with deep learning models. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 5408–5423. [CrossRef]
17. Liu, B.; Gao, G.; Gu, Y. Unsupervised Multitemporal Domain Adaptation With Source Labels Learning. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 1477–1481. [CrossRef]
18. Li, Y.; Zhang, H.; Shen, Q. Spectral–Spatial Classification of Hyperspectral Imagery with 3D Convolutional Neural Network. *Remote Sens.* **2017**, *9*, 67. [CrossRef]
19. 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.* **2018**, *56*, 847–858. [CrossRef]
20. Wang, W.; Dou, S.; Jiang, Z.; Sun, L. A Fast Dense Spectral–Spatial Convolution Network Framework for Hyperspectral Images Classification. *Remote Sens.* **2018**, *10*, 1068. [CrossRef]
21. PV, A.; Buddhiraju, K.M.; Porwal, A. Capsulenet-Based Spatial–Spectral Classifier for Hyperspectral Images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *12*, 1849–1865. [CrossRef]

22. Liu, Q.; Zhou, F.; Hang, R.; Yuan, X. Bidirectional-Convolutional LSTM Based Spectral-Spatial Feature Learning for Hyperspectral Image Classification. *Remote Sens.* **2017**, *9*, 1330. [CrossRef]
23. Xu, Y.; Zhang, L.; Du, B.; Zhang, F. Spectral-Spatial Unified Networks for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 5893–5909. [CrossRef]
24. Koda, S.; Melgani, F.; Nishii, R. Unsupervised Spectral-Spatial Feature Extraction With Generalized Autoencoder for Hyperspectral Imagery. *IEEE Geosci. Remote Sens. Lett.* **2019**, *17*, 469–473. [CrossRef]
25. Yang, J.; Zhao, Y.Q.; Chan, J.C.W. Learning and Transferring Deep Joint Spectral-Spatial Features for Hyperspectral Classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 4729–4742. [CrossRef]
26. Hu, W.S.; Li, H.C.; Pan, L.; Li, W.; Tao, R.; Du, Q. Feature Extraction and Classification Based on Spatial-Spectral ConvLSTM Neural Network for Hyperspectral Images. *arXiv* **2019**, arXiv:1905.03577.
27. Zhou, F.; Hang, R.; Liu, Q.; Yuan, X. Hyperspectral image classification using spectral-spatial LSTMs. *Neurocomputing* **2019**, *328*, 39–47. [CrossRef]
28. Liu, B.; Yu, X.; Zhang, P.; Tan, X.; Wang, R.; Zhi, L. Spectral–spatial classification of hyperspectral image using three-dimensional convolution network. *J. Appl. Remote Sens.* **2018**, *12*, 016005. [CrossRef]
29. Pan, B.; Shi, Z.; Zhang, N.; Xie, S. Hyperspectral Image Classification Based on Nonlinear Spectral-Spatial Network. *IEEE Geosci. Remote Sens. Lett.* **2016**, *13*, 1782–1786. [CrossRef]
30. Hao, S.; Wang, W.; Ye, Y.; Nie, T.; Bruzzone, L. Two-stream deep architecture for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 2349–2361. [CrossRef]
31. Sun, G.; Zhang, X.; Jia, X.; Ren, J.; Zhang, A.; Yao, Y.; Zhao, H. Deep Fusion of Localized Spectral Features and Multi-scale Spatial Features for Effective Classification of Hyperspectral Images. *Int. J. Appl. Earth Obs. Geoinf.* **2020**, *91*, 102157. [CrossRef]
32. Gao, L.; Gu, D.; Zhuang, L.; Ren, J.; Yang, D.; Zhang, B. Combining t-Distributed Stochastic Neighbor Embedding With Convolutional Neural Networks for Hyperspectral Image Classification. *IEEE Geosci. Remote Sens. Lett.* **2019**, *17*, 1368–1372. [CrossRef]
33. He, M.; Li, B.; Chen, H. Multi-scale 3D deep convolutional neural network for hyperspectral image classification. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3904–3908. [CrossRef]
34. Ben Hamida, A.; Benoit, A.; Lambert, P.; Ben Amar, C. 3-D deep learning approach for remote sensing image classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 4420–4434. [CrossRef]
35. Paoletti, M.E.; Haut, J.M.; 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]
36. Qiu, Z.; Yao, T.; Mei, T. Learning spatio-temporal representation with pseudo-3d residual networks. In Proceedings of the IEEE International Conference on Computer Vision, Honolulu, HI, USA, 21–26 July 2017; pp. 5533–5541.
37. Lin, J.; Zhao, L.; Li, S.; Ward, R.; Wang, Z.J. Active-Learning-Incorporated Deep Transfer Learning for Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 4048–4062. [CrossRef]
38. Alam, F.I.; Zhou, J.; Liew, A.W.C.; Jia, X.; Chanussot, J.; Gao, Y. Conditional Random Field and Deep Feature Learning for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 1612–1628. [CrossRef]
39. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 818–833.
40. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June 26–1 July 2016; pp. 770–778.
41. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef] [PubMed]
42. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
43. Hanif, M.S.; Bilal, M. Competitive residual neural network for image classification. *ICT Express* **2020**, *6*, 28–37. [CrossRef]

44. Zhong, Z.; Li, J.; Ma, L.; Jiang, H.; Zhao, H. Deep residual networks for hyperspectral image classification. In Proceedings of the 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA, 23–28 July 2017; pp. 1824–1827.
45. Paoletti, M.E.; Haut, J.M.; Fernandez-Beltran, R.; Plaza, J.; Plaza, A.J.; Pla, F. Deep pyramidal residual networks for spectral–spatial hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 740–754. [CrossRef]
46. Kang, X.; Zhuo, B.; Duan, P. Dual-path network-based hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 447–451. [CrossRef]
47. Aggarwal, H.K.; Majumdar, A. Hyperspectral Unmixing in the Presence of Mixed Noise Using Joint-Sparsity and Total Variation. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 4257–4266. [CrossRef]
48. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity mappings in deep residual networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 630–645.
49. Rawat, W.; Wang, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Comput.* **2017**, *29*, 2352–2449. [CrossRef]
50. Khan, S.H.; Rahmani, H.; Shah, S.A.A.; Bennamoun, M. *A Guide to Convolutional Neural Networks for Computer Vision*; Morgan & Claypool: San Rafael, CA, USA, 2018.
51. Kim, J.; Song, J.; Lee, J.K. Recognizing and Classifying Unknown Object in BIM Using 2D CNN. In Proceedings of the International Conference on Computer-Aided Architectural Design Futures, Daejeon, Korea, 26–28 June 2019; pp. 47–57.
52. Parmar, P.; Morris, B. HalluciNet-ing Spatiotemporal Representations Using 2D-CNN. *arXiv* **2019**, arXiv:1912.04430.
53. 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]
54. He, N.; Paoletti, M.E.; Haut, J.M.; Fang, L.; Li, S.; Plaza, A.; Plaza, J. Feature extraction with multiscale covariance maps for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 755–769. [CrossRef]
55. Fang, L.; Liu, G.; Li, S.; Ghamisi, P.; Benediktsson, J.A. Hyperspectral Image Classification With Squeeze Multibias Network. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 1291–1301. [CrossRef]
56. Han, X.F.; Laga, H.; Bennamoun, M. Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**. [CrossRef]
57. Hara, K.; Kataoka, H.; Satoh, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6546–6555.
58. Gonda, F.; Wei, D.; Parag, T.; Pfister, H. Parallel separable 3D convolution for video and volumetric data understanding. *arXiv* **2018**, arXiv:1809.04096.
59. Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; Paluri, M. A closer look at spatiotemporal convolutions for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6450–6459.
60. He, K.; Sun, J. Convolutional neural networks at constrained time cost. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 5353–5360.
61. 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]
62. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 37, pp. 448–456.
63. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
64. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 4489–4497.
65. Baumgardner, M.F.; Biehl, L.L.; Landgrebe, D.A. 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3. 2015. Available online: https://doi.org/doi:10.4231/R7RX991C (accessed on 20 January 2020).

66. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
67. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
68. Singh, S.; Krishnan, S. Filter Response Normalization Layer: Eliminating Batch Dependence in the Training of Deep Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020.
69. Lian, X.; Liu, J. Revisit Batch Normalization: New Understanding and Refinement via Composition Optimization. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, Okinawa, Japan, 16–18 April 2019.

© 2020 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 (http://creativecommons.org/licenses/by/4.0/).