Hyperspectral image classification using a hybrid 3D-2D convolutional neural networks

Saeed Ghaderizadeh, Dariush Abbasi-Moghadam, Alireza Sharifi, Na Zhao, Aqil Tariq

Abstract— **Due to the unique feature of the three-dimensional convolution neural network, it is used in image classification. For There are some problems such as noise, lack of labeled samples, the tendency to overfitting, a lack of extraction of spectral and spatial features, which has challenged the classification. Among the mentioned problems, the lack of experimental samples is the main problem that has been used to solve the methods in recent years. Among them, convolutional neural network-based algorithms have been proposed as a popular option for hyperspectral image analysis due to their ability to extract useful features and high performance. The traditional CNN-based methods mainly use the 2D-CNN for feature extraction, which makes the interband correlations of HSIs underutilized. The 3D-CNN extracts the joint spectral–spatial information representation, but it depends on a more complex model. To address these issues, the report uses a 3D fast learning block (depthwise separable convolution block and a fast convolution block) followed by a 2D convolutional neural network was introduced to extract spectral-spatial features. Using a hybrid CNN reduces the complexity of the model compared to using 3D-CNN alone and can also perform well against noise and a limited number of training samples. In addition, a series of optimization methods including batch normalization, dropout, exponential decay learning rate, and L2 regularization are adopted to alleviate the problem of overfitting and improve the classification results. To test the performance of this hybrid method, it is performed on the Salinas, University Pavia and Indian Pines datasets, and the results are compared with 2D-CNN and 3D-CNN deep learning models with the same number of layers.**

Keywords — **convolutional neural network (CNN), deep learning, hyperspectral image (HSI) classification, spectral-spatial features.**

I. INTRODUCTION

Information and data related to spectral images have been available since the late 1980s, showing the reflective wavelengths in the range of 400 to 2400 nm. These images are used for distance measurement studies in research fields such as agricultural, forestry and environmental monitoring and land surface mapping. Using the classification method, hyperspectral images can be used to the maximum in the future research [1]. Hyperspectral datasets have rich information both spatially and spectrally. However, spectral and spatial correlations make a lot of such information redundant. One can obtain efficient representations using techniques such as band selection [2], multi-modal learning [3] dimensionality reduction [4]. Also, advanced methods and technologies in the field of machine learning have provided the conditions to benefit from hyperspectral image information in various fields such as natural language processing [5], medical connection prediction [6], remote sensing image processing [4–9], etc. In the early stages, traditional classification methods are based on spectral information, which generally includes two main elements: feature engineering and classifiers [10]. The function of feature engineering is to obtain discriminative features or bands and reduce the dimension of HSIs. Two common methods in feature engineering are feature selection and feature extraction [11]. Feature extraction is aimed at changing high-dimensional space data to low-dimension space data, so that the categories can be easily separated from each other. Typical methods of feature extraction include minimum noise fraction (MNF) [12], linear discriminant analysis (LDA) [13], principal component analysis (PCA) [14], independent component analysis (ICA) [11], and so on. Whereas the function of feature selection is to retain the spectral information of the most representative bands from the raw HSIs and discard the bands that contribute less to the classification. Common methods of feature selection include Jeffries-Matusita distance [15], spectral angle mapper (SAM) [16], etc. Features generated by feature engineering are used as the input of the classifier. Representative classifiers include k-nearest neighbor (KNN) [17], random forest (RF) [18], support vector machine (SVM) [19], etc. However, the traditional classification methods based on spectral information do not make full use of the spatial information of HSIs. Nevertheless, the traditional classification methods of HSIs, rely on hand-crafted features with limited representation ability, which cannot fit the classification task well. On the other hand, researchers have exploited spectral-spatial contextual information and developed a variety of classification algorithms. These methods assume that neighboring pixels share similar spectral signatures and thus belong to the same landcover types. Based on this assumption, spectral-spatial feature extraction methods, such as Gabor filtering [20], wavelet transformation [21] are proposed to improve the discrimination of classes.

Most recently, deep learning has emerged as the state-of-the-art machine learning technique with great potential for HSI classification [22]. Instead of depending on shallow manually-engineered features, deep learning techniques can automatically learn hierarchical features (from low-level to high-level) from raw input data. Such learned features have achieved tremendous success in many machine vision tasks. Nowadays, the existing classification methods based on the CNN framework provide rich solutions for HSI classification tasks [18-31]. In general, there are three categories of the convolution operation in the

existed CNN HSI classification frameworks including 1D CNN, 2D CNN, and 3D CNN respectively.

The network architecture of 1D CNN is designed to use the pixel vector along the radiometric dimension as a training sample to extract deep feature, which is called spectral-based classification approach conceptually. In [23], it was the first time to employ CNN with multiple layers for HSI classification directly in the spectral domain. A novel RNN model [24] was proposed to effectively analyze hyperspectral pixels as sequential data to capture the intrinsic feature, which designed a new activation function to train the network without the risk of divergence.

The 2D CNN model for HSI classification is called spatial-based classification approaches tried to learn spatial features [20–22] by utilizing the similar approaches for traditional images with the colorful pattern of RGB, which brought out an inevitable drawback caused by the ignore the united spectral-spatial attributes of the specific hypercube. In [27], Hao and Wang designed the SRCL model for HSI classification, which explored a super-resolution-aided way to construct a spatially enhanced image. In [26], a CNN-MRF model was proposed to integrate spectral and spatial information in a unified Bayesian framework by learning the posterior class distributions. Li and Xie [25] introduced a CNN model to reconstruct an enhanced image cube by bands selection with a new spatial feature-based strategy.

Since the hyperspectral image is originally 3D hypercube with the spectral and spatial continuity, the HSI classification methods integrated both spectral and spatial information have gained more popularity [23-31]. Handling the hyperspectral CNN classification with 3D convolutions is a straightforward way, which is also called the spectral-spatial classification approach. In this way, 3D regions with joint spatial-spectral information can be processed simultaneously. Chen et al. [28] used several convolutional and pooling layers of 3D CNN for HSI classification. In 2017, a three-dimensional convolutional neural network (3D-CNN) was introduced for the classification of hyperspectral images [29]. The proposed HSI cube data method extracts spectral-spatial properties without relying on any preprocessing or post-processing. It also requires fewer parameters than other deep learning methods, which are lighter in model and easier to teach. This is the main motivating factor in the current work. Feng and Yu [30] proposed a multiclass spatial-spectral GAN (MSGAN) method to utilize generators for the samples production and the discriminator for the joint spatial-spectral feature extraction. In [31], a semi-supervised 3D convolutional neural network for the spectral-spatial HSIC is proposed by engaging adaptive dimensionality reduction (ADR) to deal with the problem of the curse of dimensionality. In [32] the authors propose a semi-supervised approach to exploit multimodal data for better inference. With GCNs, label information is allowed to flow from labeled nodes to unlabeled nodes. A novel version of GCN called miniGCNs is proposed, where regular patches of the original HSI are used for training the GCN model, yielding lower computation cost. Recently, a series of popular deep learning-based methods have been exploited for spatial-spectral classification. In [33], Pauletti et al. proposed a deep and dense 3D-CNN for full use of HSIs information. In [34], the input data is fed to CNN in two different architectures, and several features are taught to better predict the class label for an HSI pixel. Roy et al. Proposed HybridSN, which extracts features using CNN 3D-2D layers [35]. In [36], a cascaded RNN model was designed to explore the redundant and complementary information of HSIs by utilizing two RNN layers. In [37], the multi-scale hierarchical recurrent neural networks (MHRNNs) was proven to be efficient in hyperspectral image classification, which learns the multi-scale local feature by 3D CNNs and learns the spatial dependency of non-adjacent image patches in the spatial domain by RNN. However, there are two main limitations revealed with the 3D convolution model. On one hand, with the increasing number of the 3D kernels, the complexity and time cost get higher, on the other hand, more training examples are needed to train a deeper 3D CNN model which is not practical as the public hyperspectral image datasets are rather small.

The CNN model has an advantage over other learning models due to its high ability to identify spatial and spectral features and is a significant model in the field of HSI classification, but this model also has weaknesses, for example, during the process of gradient descent, it is easy to make the results converge to the local minimum, and the pooling layer will lose a lot of useful information, as it is known, the preprocessing stage plays a vitally important role in the accuracy of classification models, and PCA is known as a preprocessing method among HSI classification models and this preprocessing eliminates the non-linear features of the image. the 2D CNN alone is not able to extract good discriminating feature maps from the spectral dimensions. Similarly, a deep 3D CNN is more computationally complex and this alone seems to perform worse for classes having similar textures over many spectral bands. The use of 2D-CNN and 3D-CNN together leads to maximum accuracy so that they make full use of spectral as well as spatial feature maps.

The proposed model presents the following characteristics which make it different from the models mentioned above:

- It uses a 3D fast learning block which makes the model more robust and efficient by introducing 3D depthwise separable convolution block and the fast convolution block.
- The network parameters are fewer compared to the existing methods which reduce the overfitting.
- It uses many algorithms for optimization, including dropout, batch normalization, exponential decay learning rate, and L2 regularization, so as to make the network more robust and generalized.

II. MATERIAL AND METHODS

A. *Convolutional Neural Networks*

Recently, the use of deep learning techniques in the classification process has received much attention. CNN is the same multilayer neural network that consists of different layers such as: convolution layer, pooling layer, and fully connected layer. The convolution operation is performed on the input data in the convolution layer as the primary layer of the CNN model. Convolution is a dot product operation between two matrices, namely receptive field and kernel (learnable parameters). Generally, the kernel is smaller than the size of the input data, and the kernel slides are located in the receptive field and the feature map will be created according to the input data and

available features. the pooling layer is effective in reducing the spatial dimension of the feature map. The fully connected layer, which consists of neurons and the nature of perceptron, is multilayered, and in which all neurons connected to every succeeding layer neurons, and the output features are used in the mapping. This layer is used to map features into the output. The individual neuron's output for inputs x is calculated as:

$$z = f(w * z + b) \tag{1}$$

Here w is the filter weight, and b is biased. f (.) stands for the nonlinear activation applied on a weighted sum of input.

$$f(x) = \max(0.x) \tag{2}$$

f (.) is a nonlinear function known as the activation function. In this section, the ReLU function is used [38]. This function, if the value of x is greater than zero, is the output of x, and if the value of x is less than or equal to zero, the output is zero. The main advantage of using the ReLU nonlinear function is that it has a fixed derivative for all inputs greater than zero. This fixed derivative accelerates network learning. The main purpose of layers is to extract useful features to be used in later layers to perform the classification process. But the 2D-CNN model consists of three main steps: patch extraction, feature extraction, and label identification. According to a hyperspectral image, we first extract a small patch with the center of each pixel as input. Then, an in-depth learning model is developed to acquire the feature maps of these patches. Finally, the label for each pixel is categorized based on the corresponding patch feature map. For all three models, we remove the pooling layers to preserve as much information as possible from a single pixel. The three-step processing of the 2D-CNN model is shown below. Suppose a hyperspectral image is the size N×M×D, where N and M are the number of rows and columns in the image, and D represents the number of spectral bands. Our goal is to predict the label of each pixel of the image. the first step in processing the S×S×B patch extraction model is for each pixel. Specifically, each patch (e.g., spatial context) is built around a pixel, the central point of the patch. For pixels near the edge of the image, there may not be enough information to make a patch of the expected size. Accordingly, we provide a spatial background for these pixels by using a mirror padding operation. For the second stage of processing, each extracted patch with several channels is treated as an image separately. Thus, a deep CNN model with 2D convolution layers is applied to extract feature maps for patches. The 2D-CNN operation formula in each layer can be shown as follows:

$$z_{x.y}^{l.r} = f\left(\sum_{m}\sum_{i=0}^{I_l-1}\sum_{j=0}^{J_l-1} w_{i.j}^{l.r.m} * z_{x+i.y+j}^{l-1.m} + b^{l.r}\right) \tag{3}$$

l represents the layer to be considered, r is the number of feature maps in layer l, $z_{x,y}^{l,r}$ the output in position (x, y) is the rth feature map in layer i. $b^{l,r}$ is the network bias. f (.) indicates the layer activation function. The m index is a set of feature maps of layer $(l-1)$, which are the inputs of layer l. $w_{i,j}^{l,r,m}$ is a value in position (i, j) where the convolution kernel is related to the rth feature map in the l-th layer, I_l and J_l are the row and column sizes of this kernel. As shown in Fig. 1, the operational details of the 3D-CNN model are quite similar to those of the 2D-CNN model. The main difference is that the 3D-CNN model has an additional step. In this step, we arrange the hyperspectral bands D in ascending order. By doing this, images of similar spectral bands are arranged in sequence, maintaining their correlations in a spectral context. The patch extraction step and the label recognition step of these two models are quite similar. For the feature extraction step, a 3D convolution operator is applied to the 3D-CNN model instead of the 2D convolution operator. The formula for 3D convolution operation is as follows:

$$z_{x.y.d}^{l.r} = f\left(\sum_{m}\sum_{i=0}^{I_l-1}\sum_{j=0}^{J_l-1}\sum_{k=0}^{K_l-1} w_{i.j.k}^{l.r.m} * z_{x+i.y+j.d+k}^{(l-1).m} + b^{l.r}\right) \tag{4}$$

where K_l refers to the size of the 3D kernel along the spectral dimension and k is the number of kernel in layer l. $w_{i,j}^{l,r,m}$ is a value in position (i, j, k) whose convolution kernel is related to the rth feature map in the lth layer. The ReLu function is again shown as the activation function f.

The computational cost of a 3D convolution operation is

$$k \times k \times k \times c_G \times c_F \times l_F \times w_F \times h_F \tag{5}$$

Where $l_F \times w_F \times h_F \times c_F$ and $l_G \times w_G \times h_G \times c_G$ are the input and output size, respectively, with l, w and h representing the length, width and height; and c_F, c_G are the number of channels before and after the convolution. The kernel k here is of the following size: $k \times k \times k \times c_G \times c_F$, where k is the filter side length.

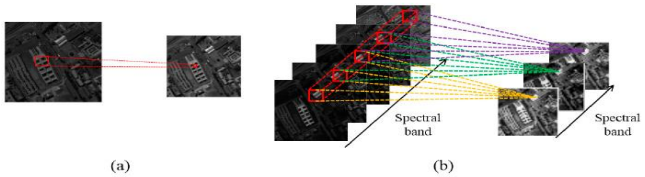


Fig. 1. (a) 2D convolution operation, as per equation (3). (b) 3D convolution operation as per equation (4).

To classify HSI, the three-dimensional convolution operation simultaneously analyzes the input data in both spatial and spectral dimensions, and the 2D complexity operation includes the input data in the spatial dimension. 3D convolution can store the spectral information of the input HSI data, and this is especially important for hyperspectral images containing rich spectral information, whereas if two-dimensional convolution operations are applied to the HSI, significant spectral information from They disappear, but for 2D convolution operation, it is 2D output, regardless of whether it is applied to 2D or 3D data.

B. Proposed Neural Network model

For HSI analysis, researchers demonstrated that the redundancy from inter band correlation is very high. The data structure in the spectral dimension can be reduced without the significant loss of useful information for subsequent utilization. However, an HSI contains hundreds of spectral bands, which increases the pressure on the network model to process data and

also consumes a lot of computing resources. In recent years, many studies on HSI classification use PCA for data preprocessing [33-34]. PCA is the most commonly used linear dimensionality reduction method. Its goal is mapping high-dimensional data to corresponding low-dimensional data through some linear projection that is, maximizing the variance. This method reduces the data dimension while retaining more original data features. The core idea of PCA is calculating the similarity between different data features, extracting the main features according to the strength of the correlation, and completing the information fusion [41]. And hence, PCA is applied to the original HSI for dimensionality reduction in the proposed method.

As in the original paper [42], the 3D depthwise separable convolution is a factorized convolutions which divides the normal 3D convolution into a 3D depthwise convolution and a 1×1×1 convolution called 3D pointwise convolution to combine the output of the 3D depthwise convolution later. This process significantly reduces the computation and the model sizes. Equation (4) becomes

$$v_{x.y.d}^{l.r} = f\left(\sum_{i=0}^{I_l-1}\sum_{j=0}^{J_l-1} w_{i.j.s}^{l.h} * z_{x+i.\ y+j..d+s}^{(l-1).h} + b^{l.r}\right) \quad (6)$$

$$s = ceil\left(\frac{r}{m}\right).h = r - s \times m - 1 \quad (7)$$

$$z_{x.y.d}^{l.r} = f\left(\sum_{m} w^{l.r.m} * v_{x.y.d}^{l.m} + b^{l.r}\right) \quad (8)$$

The computational cost of such decomposition is

$$k \times k \times k \times c_F \times l_F \times w_F \times h_F + c_F \times c_G \times l_F \times w_F \times h_F \quad (9)$$

where $l_F \times w_F \times h_F \times c_F$ and $l_G \times w_G \times h_G \times c_G$ are the input and output size, respectively, with l, w and h representing the length, width and height; and c_F, c_G are the number of channels before and after the convolution. The kernel K here is of size $k \times k \times k \times c_G \times c_F$, where k is the filter side length. Compared with the standard 3D CNN, the computational cost becomes:

$$\frac{k \times k \times k \times c_F \times l_F \times w_F \times h_F + c_F \times c_G \times l_F \times w_F \times h_F}{k \times k \times k \times c_G \times c_F \times l_F \times w_F \times h_F} \quad (10)$$

$$= \frac{k \times k \times k \times c_F + c_F \times c_G}{k \times k \times k \times c_G \times c_F} = \frac{1}{c_G} + \frac{1}{k^3} \quad (11)$$

The amount of calculation is reduced in Equation (11) by about eight to nine times.

The model proposed in this study combines the 3D CNN and 2D CNN to extract good spectral and spatial features maps from the HSI at a cheap cost. The proposed model uses 3D stacked convolution layers (a Conv3D – fast learning block) followed by a reducing dimension block which consist of a Conv3D + reshaping operation + a Conv3D, and then the output features maps from that block is then reshaped and fed to a Conv2D to learn more spatial features. The output of the Conv2D layer is flattened and passed to the first fully connected layer in which a dropout layer was added before the last fully connected layer. The 3D fast learning CNN block of the proposed model is with much less computational cost and faster than the normal 3D CNN block because of the presence of depthwise separable convolution and the fast convolution block in the fast learning block. The architecture of the proposed method is shown in Figure 2.

This report briefly discusses the proposed convolutional neural network architecture. You can see the workflow in Fig. 1. Cubes of spectral and spatial data are denoted by $I \in R^{N\times M\times D}$, so that I is the main image, N is the number of rows, M is the number of columns, and D is the number of spectral bands in the I image. Each HSI pixel in I has a vector labeled $Y = (y_1, y_2, \ldots, y_C) \in R^{(N\times M)\times C}$, where C represents the number of images classes. In a hyperspectral image, there is a high correlation between neighboring bands. Reducing the dimension is therefore a widespread preprocessing step required for effective classification. To eliminate spectral band redundancy, we apply PCA to the primary data of the I spectral images. In addition to reducing the number of spectral bands from D to B, PCA also preserves spatial dimensions (N×M). $X \in \mathcal{R}^{N\times M\times B}$ the hyperspectral image reduced by PCA.

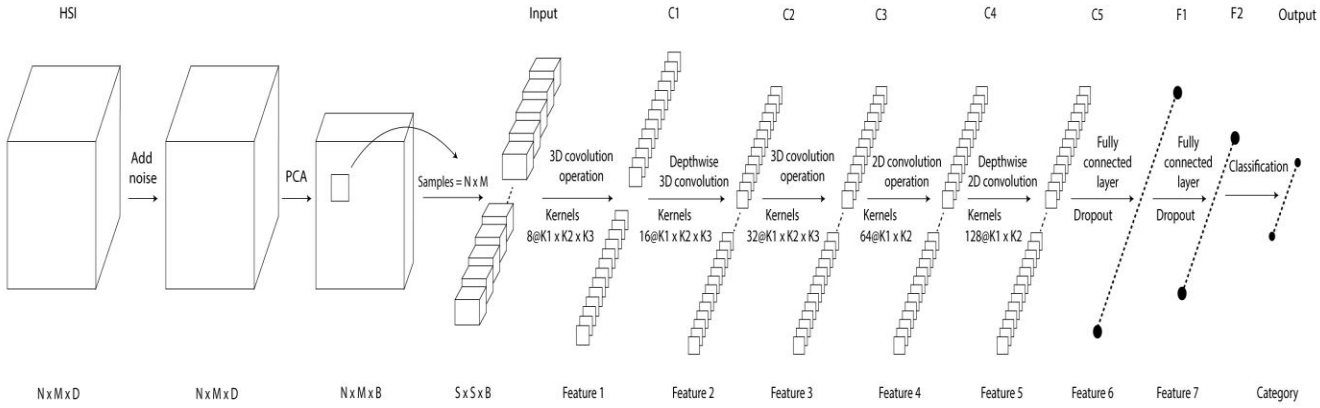


Fig. 2. Hyperspectral image classification architecture using hybrid convolutional neural network.

To use image classification techniques, hyperspectral data cubes for input are divided into small 3D patches $P \in R^{S\times S\times B}$ whose class labels are defined by a central pixel. The number of input data patches is initially the same as the size of the correct N × M labels. But in the correct labels there is a background that after removing the background from the correct labels and patches, we enter the data as input to the network. The convolution layer consists of a sliding kernel on the input image. This kernel contains weights that change during the training process to extract important feature maps from the input. These features are used during the classification process. Patches of hyperspectral data are entered into the network for training, after passing through different layers of the network, ie applying weights and nonlinear functions to the data, a result is obtained at the end of the network. Result z the output of the network will be different from the desired result we expect y. The difference between the network output results and the desired result of these two outputs is a function of the network error or cost. In neural networks, certain functions are usually used. In this report, the cross-entropy error function and mini-batch update are used, which is more suitable for calculating the probabilities of the neural network, and the following relation is obtained:

$$L_0(y.z) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log z_i] \qquad (12)$$

Where y_i is the real label and z_i is the output of our model, and m is the number of batches. After calculating the error by the cost function, we must use the obtained error to modify the network parameters, called the optimization process. The optimization process is the correction and updating of weights to achieve the minimum error. The gradient descent method is one of the optimization methods used. Another method used for this purpose is the Adam technique (adaptive motion estimation) to solve non-convex problems [43]. In this particular method, we consider a different learning rate for each weight amount in the network from the first and second moment of the gradient. After defining the cost function and the optimization function, we come to the central part of network training, weight correction. Weight correction in neural networks is usually done by a method called backpropagation. The chain rule is used to correct the weight, and the backward propagation is done by calculating the output error and returning layer by layer from the output to the input.

In this method, we have two round trips. In the path, the input enters the network and the parameters are applied to it, and output is created and the error is calculated. In the return path, the obtained error is used to correct the parameters, but the parameters are updated from the end to the beginning; That is, it starts from the last layer of the network and corrects the parameters, and then goes back one layer, and the same process continues until the first layer of the network, which is the return path. How the parameters are corrected by the error function is determined by the optimization function. With the new parameters obtained, we calculate the output, and the new output will be a new error. We repeat this round trip process until we get the least error. Since HSI data is a limited training sample, this problem can be somewhat improved with Dropout and network overfitting can be prevented [44]. Dropout technique in practice, neurons are removed with a probability of p or conserved with a chance of 1-p. In each iteration of the process, it randomly selects some neurons and removes them from the network. In general, the dropout rate between 0.2 and 0.5 can be the right choice, which is set at 0.4 in this report. Finally, the classification is done using a possible softmax model. The softmax function takes a next C vector of real numbers such as Z as input and gives values between [0,1] as output whose sum of its components is 1.

$$s(Z)_i = \frac{e^{Z_i}}{\sum_{j=1}^{C} e^{Z_j}} \qquad For\ i = 1.2. \ldots . C \qquad (13)$$

Z_i represents the property extracted with the trained model and C represents the number of classes. Finally, by maximizing the argument, the class label can be predicted.

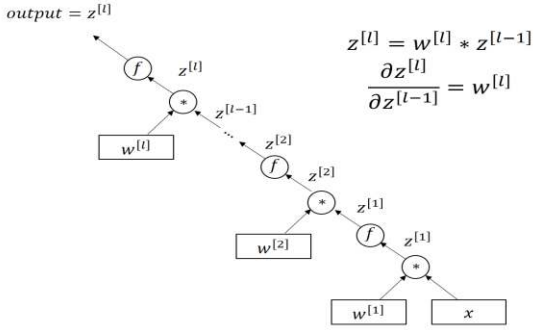


Fig. 3. A simple demonstration to better understand the concept of backward diffusion and to show the relationship between the output of the layers and the weights.

CNN-based HSI classification methods mainly include 2D CNN and 3D CNN. Roy et al [35] used three 3D CNN and a 2D CNN to build the network. Hamida et al [45] mainly utilized 3D CNN in the network. Combining 2D CNN with 3D CNN can improve the performance to a certain extent, but selecting the appropriate number of network layers is not easy to control.

The proposed model consists of seven layers. The first to third layers, a 3D convolution (C1-C3) is used to extract spectral-spatial features, followed by a 2D (C4-C5) convolution in the fourth and fifth layers, and two fully connected layer (F1-F2) in the end of the model. Each neuron in a fully connected layer connects to all the neurons in the previous layer and sends the output value to the classifier. Though the pooling layer (such as the max-pooling layer and the average pooling layer) can reduce the dimensions of feature maps and simplify calculations because the spatial resolution in the image is reduced, the pooling layer is not applied to preserve more information. In the first layer, the dimensions of the 3D convolution kernel are 8×3×3×3×1 (for example, in Fig. 2, $K_1^1 = 3, K_2^1 = 3, K_3^1 = 3$), 16×3×3×3×8 (for example, in Fig. 2, $K_1^2 = 3, K_2^2 = 3, K_3^3 = 3$), 32×3×3×3×16 (for example, in Fig. 2, $K_1^3 = 3, K_2^3 = 3, K_3^3 = 3$), in the subsequent first, second, and third convolution layers, respectively, where 32×3×3×3×16 means 32 3D-kernels of dimension 3×3×3 (i.e., two spatial and one spectral dimension) for all sixteen3-D input feature maps. the output of the first layer contains eight feature maps that are used as the input is used in the next layer. In the fourth layer, to perform 2D convolution operations, its information needs to be a three-dimensional image, by resizing, we prepare the input size for 2D convolution operations (in Table I, you can see the

deformation performed). After preparing the input of the fourth layer, the 2D convolution operation, whose kernel dimensions are 3×3 (for example, $K_1^2 = 3, K_2^2 = 3$ in Fig. 2), is applied to the input of the forth layer and 64 feature maps for the output of the next layer, and 288 is the number of 2D input feature maps. The next layer, the dimension of depthwise 2D convolution kernel is 128×3×3×64 (i.e., $K_1^5 = 3$ and $K_2^5 = 3$ in Fig. 1).

Finally, by flattening the production of the fifth layer, all the neurons are connected to the neurons of the next layer, which is considered to be 256. A summary of the proposed model in terms of layer type, output map dimensions, and the number of parameters is given in Table I. It can be seen that the largest number of parameters is in the first dense layer (fully connected). The number of neurons in the last layer of Dense is 16, which is the same number of classes in the Salinas dataset. Therefore, the total number of parameters in the proposed model depends on the number of classes in a data set. The total number of trainable weight parameters in the proposed model for the Salinas dataset is 1,033,728. All weights are initially initialized randomly; they are then taught using the backpropagation algorithm with the Adam optimizer and Softmax classification. We train the network for 100 epochs of mini-batches with 256 and a learning rate of 0.001 without data augmentation.

TABLE I. SUMMARY OF CNN HYBRID PROPOSED MODEL FOR SA DATA.

Layer (type)	Output Shape	# Parameter
InputLayer_1	(15, 15, 15, 1)	0
Conv3D_1	(13, 13, 13, 8)	224
BatchNormalization_1	(13, 13, 13, 8)	32
Activation	(13, 13, 13, 8)	0
DepthwiseConv3D	(11, 11, 11, 16)	448
BatchNormalization_2	(11, 11, 11, 16)	64
Activation	(11, 11, 11, 16)	0
Conv3D_2	(9, 9, 9, 32)	13856
BatchNormalization_3	(9, 9, 9, 32)	128
Activation	(9, 9, 9, 32)	0
Reshape_1	(9, 9, 288)	0
Conv2D_1	(7,7, 64)	165952
BatchNormalization_4	(7, 7, 64)	256
Activation	(7,7, 64)	0
DepthwiseConv2D	(5,5, 128)	1280
BatchNormalization_5	(5, 5, 128)	256
Activation	(5,5, 128)	0
Flatten_1	(3200)	0
Dense_1	(256)	819200
Dropout_1	(256)	0
Dense_2	(128)	32768
Dropout_2	(128)	0
Dense_3	(16)	2048
Total Trainable Parameters : 1,033,728		

C. Optimization Methods

In the field of the hyperspectral classification, the large amount of noise in the HSIs, the limited number of labeled samples, the complex structure of the model, and the numerous parameters of 3-D CNN all lead to the phenomenon of overfitting. To prevent overfitting and improve the accuracy, a series of optimization methods including batch normalization, dropout, exponential decay learning rate, and L2 regularization are adopted.

1) Batch Normalization

To alleviate the problem of overfitting and accelerate the convergence of the network, the optimization method of BN is used in the paper. Suppose the input of BN is $X = [x_1, x_2, \dots, x_m]$. Where, x_m represents one of the samples, and m represents the batch size. The mean μ_B and variance σ_B^2 of the input data can be calculated by (14) and (15), respectively:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{14}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{n} (x_i - \mu_B)^2 \tag{15}$$

Next, each element of the input is normalized, as show in (19). Where, ε represents a constant.

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{16}$$

Finally, the final output y_i is obtained through scaling and shifting, as shown in the following equation:

$$y_i = \gamma \hat{x}_i + \beta \tag{17}$$

2) L2 Regularization

The basic idea of L2 regularization which can alleviate the problem of overfitting is to add an L2 norm penalty to the loss function as a constraint. The loss function J with L2 regularization is calculated as follows:

$$L = L_0 + \frac{\lambda}{2m}\sum_{w} w^2 \tag{18}$$

where, J0 represents the original loss function, $\frac{\lambda}{2m}\sum_w w^2$ is the L2 norm penalty, λ is the hyperparameter that controls the ratio of the L2 norm penalty, m is the size of the training samples and w represents the weights of the model.

3) Exponential Decay Learning Rate

The setting of learning rate is very important, which determines whether the model converges to the global optimal value and affects the running speed. If the learning rate is too large, the gradient of the model will oscillate back and forth on both sides of the global optimal solution and cannot converge. And if the learning rate is too small, the convergence speed of the algorithm will be very slow and the training time will increase, resulting in the waste of resources. To solve these problems, the exponential decay learning rate is used. The core idea of exponential decay learning rate is to obtain the sub-optimal solution quickly by using a large learning rate at the beginning, and then gradually reduce the learning rate as the iteration continues, so as to make the gradient converge to the optimal value. The equation of exponential decay learning rate η_d is calculated as follows:

$$\eta_d = \eta \times d_r^{\frac{g_s}{d_s}} \tag{19}$$

where, η represents the initial learning rate, d_r represents the decay rate, g_s represents the global step and d_s is the decay step.

4) Dropout

Dropout is adopted to alleviate the problem of overfitting. The basic principle of dropout is that the weights of some neurons in the hidden layer stop updating according to a certain probability in the training process, so as to ease the complex co-adaptation relationship between neurons.

D. Data set

We have used three publicly available HSI data sets, namely, Salinas Scene (SA), Pavia University (PU) and Indian Pines (IP). The SA data set contains the images with 512×217 spatial dimension and 224 spectral bands in the wavelength range of 360–2500 nm. There are 16 classes in this dataset. The SA data set mainly reflects vegetation information and includes a variety of features with a regular distribution. Table II lists 16 challenging land-cover categories and the training and test sets. The PU data set with the spatial dimension of 640 × 310 and 103 spectral bands in the wavelength range of 430–860 nm with a spatial resolution of 1.3 meters. The label is divided into nine urban classes. The class name and the number of training and test sets are detailed in Table III.

TABLE II. SAMPLE SIZE FOR SALINAS.

#	Class	Training	Testing
1	Broccoli green weeds 1	100	1909
2	Broccoli green weeds 2	100	3626
3	Fallow	100	1876
4	Fallow rough plow	100	1294
5	Fallow smooth	100	2578
6	Stubble	100	3859
7	Celery	100	3479
8	Grapes untrained	100	11171
9	Soil vineyard develops	100	6103
10	Corn senesced green weeds	100	3178
11	Lettuce romaine, 4wk	100	968
12	Lettuce romaine, 5wk	100	1827
13	Lettuce romaine, 6wk	100	816
14	Lettuce romaine, 7wk	100	970
15	Vineyard untrained	100	7168
16	Vineyard vertical trellis	100	1707
-	Total	1600	52529

TABLE III. SAMPLE SIZE FOR PAVIA UNIVERSITY.

#	Class	Training	Testing
1	Asphalt	100	6531
2	Meadows	100	18549
3	Gravel	100	1999
4	Trees	100	2964
5	Sheets	100	1245
6	Bare Soil	100	4929
7	Bitumen	100	1230
8	Bricks	100	3582
9	Shadows	100	847
-	Total	900	41876

The PU data sets mainly reflect urban landscape information and include small types of features with an utterly irregular distribution. The IP data set has images with 145×145 spatial dimension and 224 spectral bands in the wavelength range of 400 to 2500 nm, out of which 24 spectral bands covering the region of water absorption have been discarded. The ground truth available is designated into 16 classes of vegetation. Table IV lists 16 main land-cover categories involved in this studied scene, as well as the number of training and testing samples used for the classification task. The experiments are implemented using the Keras framework on Google Colab.

TABLE IV. SAMPLE SIZE FOR INDIAN PINES.

#	Class	Training	Testing
1	Alfalfa	5	41
2	Corn-notill	143	1285
3	Corn-mintill	83	747
4	Corn	24	213
5	Grass-pasture	48	435
6	Grass-trees	73	657
7	Grass-pasture-mowed	3	25
8	Hay-windrowed	48	430
9	Oats	2	18
10	Soybean-notill	97	875
11	Soybean-mintill	245	2210
12	Soybean-clean	59	534
13	Wheat	20	185
14	Woods	126	1139
15	Buildings-Grass-Trees-Drives	39	347
16	Stone-Steel-Towers	9	84
-	Total	1024	9225

III. RESULTS AND DISCUSSION

In this section, we introduced three HSI datasets, described the structure and process of the model, and evaluated the proposed methods using classification criteria such as mean accuracy (AA), kappa coefficient (kappa), and overall accuracy (OA). 1) Overall accuracy (OA): The percentage of correctly classified pixels. 2) Average accuracy (AA): The mean value of the OAs measured over each category. 3) Kappa coefficient (Kappa): A statistic measurement over the inter-rater agreement among qualitative items.

Randomly equal amounts of data from each set were used for training and the remainder for model testing. 10% of the training samples are dedicated to validation set. To obtain a more convincing estimate of the capabilities of these methods, the simulation is repeated 25 times for each data set, finally, the mean accuracy of the report is given.

A. Experiment 1: Comparing accuracy between the proposed model and other classification models

Our first experiment shows a comparison between the proposed method and the three different and well-known classification methods in HSI with the number of training samples of 100 for each class. Table V shows the classification results obtained by different classifiers for the Salinas dataset. Salinas data have a significant order in terms of spatial distribution, and network performance has a structure with acceptable overall accuracy. According to Table V, the best result is related to the combined method, which achieved an overall accuracy of 99.07%, which is 0.66% higher than the second accuracy (98.41%) obtained by the HybridSN model.

TABLE V. CLASSIFICATION ACCURACY ON THE SALINAS DATASET (100 SAMPLES FROM EACH CLASS FOR TRAINING)

Class	SVM	2D CNN	3D CNN	HybridSN	3D-2D CNN
1	99.78±0.14	100.0±0	99.99±0.01	**100.0±0.0**	99.99±0.01
2	99.65±0.17	99.99±0.02	99.97±0.06	100.0±0.0	100.0±0.0
3	93.56±2.25	99.90±0.21	99.91±0.13	99.87±0.17	**99.94±0.15**
4	99.27±0.22	99.61±0.44	99.50±0.62	**99.65±0.48**	99.52±0.57
5	97.67±0.75	98.67±1.16	**99.69±0.21**	99.47±0.49	99.55±0.69
6	99.65±0.12	99.97±0.06	99.99±0.02	99.83±0.23	**99.99±0.01**
7	99.70±0.14	99.96±0.02	**100.0±0**	100.0±0.0	99.99±0.00
8	78.29±3.52	90.43±2.42	92.83±2.74	95.11±2.51	**97.04±1.19**
9	99.70±0.10	99.95±0.05	**100.0±0**	100.0±0.0	99.99±0.00
10	93.70±0.51	98.43±0.96	98.644±1.08	**99.47±0.01**	98.85±0.83
11	93.23±2.18	99.72±0.30	100.0±0	100.0±0.0	100.0±0.0
12	99.88±0.10	99.99±0.03	99.94±0.09	**100.0±0.0**	99.99±0.16
13	99.14±0.31	99.84±0.24	99.98±0.03	99.95±0.06	**100.0±0.0**
14	95.87±1.39	99.84±0.24	**99.89±0.13**	99.50±0.35	99.56±1.00
15	70.77±3.25	92.71±3.63	95.10±2.39	96.77±1.13	**98.75±0.54**
16	98.34±0.70	99.60±0.31	**99.89±0.19**	99.94±0.08	99.49±0.56
OA	90.27±0.57	96.76±0.66	97.68±0.51	98.41±0.57	**99.07±0.19**
Kappa	89.14±0.62	96.39±0.73	97.41±0.57	98.23±0.63	**98.96±0.21**
AA	94.89±0.38	98.66±0.30	99.08±0.20	99.35±0.17	**99.54±0.14**

Classes 8 and 15 are challenging to classify and have lower accuracy than other classes. Fig. 4 shows the best results of the confusion matrix of all models. It can be seen that classes 8 and 15 have a percentage of error due to their high similarity, but the hybrid model, due to its structure, can have less error in classes that are spectrally and spatially complex. Based on the measured values from different classes, the stability of the proposed network will be obtained. According to Table VI, it is shown that the combined method with an overall accuracy of 98.90% achieved the best result, which is 0.53% higher than the second-best accuracy (98.37%) obtained by HybridSN.

The hybrid model has better classification accuracy in most classes compared to other related methods. A comparison between the ambiguity matrices is shown in Fig. 5 to illustrate the performance of the hybrid model better. If the Class 8 hybrid model is used, it has an error of approximately 1.7% with the Class 3, and also for other classes that were misclassified, it has an error percentage of less than 0.9%. But in different models, there are many error classes with an error rate of more than 2%. Of course, the SVM model has abysmal performance compared to other classification models.

The training sample percentage of the Indian Pines Scene is set to 10% randomly, the patch size is fixed as 15×15. According to Table VII the listed value, it can be observed that our proposed 3D-2D CNN model achieves the overall accuracy of 97.14%, and the second (97.09%) OA is implemented by the HybridSN model. The accuracy obtained on this dataset is clearly lower than the accuracies obtained on the other two datasets using the same method. To explore the reason for this, we drew the confusion matrix for methods on this dataset for a quantitative analysis, as shown in Fig. 6. According to Fig 6, the confusion matrix shows the best accuracy in the experiments, the 3D-2D CNN model has achieved higher accuracy for classes 1, 4, 7 and 9 where there are very few training samples available than other methods.

The methods compared in the experiment can be divided into two ways. The SVM method, which belongs to the traditional classification and classifies based on spectral information, does not make full use of the spatial information of hyperspectral images. While 2D CNN, 3D CNN, HybridSN and 3D-2D CNN are all deep learning classification methods. The 2D CNN method is based on spatial information classification, while the 3D CNN, HybridSN and 3D-2D CNN classification methods are based on spectral-spatial information. Deep learning-based classification methods are usually superior to traditional classification methods. Deep learning models have a hierarchical structure, which can automatically learn high-level semantic information from data. Therefore, they are more powerful than conventional methods in extracting features. Spectral-spatial information-based classification methods perform better than spectral-information-based or spatial information-based methods. Because features extracted by spectral-spatial information classification methods include not only spectral information but also spatial information that can contribute to the effective use of features. Hybrid model performance can reach the most advanced level because the model has a hybrid structure and can extract more specific information from hyperspectral images than to other models.

We also make a visual comparison between different classification methods in the form of classification maps, as shown in Figs. 7–9. In general, pixelwise classification models (e.g., SVM) result in salt and pepper noise in the classification maps. As expected, the 3D-2D CNN method obtain smoother and more detailed maps in comparison with other competitors, mainly due to the effective combination of different features that further enhance the HSI representation ability. It should be noted, however, that the batchwise input in CNNs could lead to losing some edge details to some extent (e.g., 2D-CNN and 3D-CNN).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTARS.2021.3099118, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing

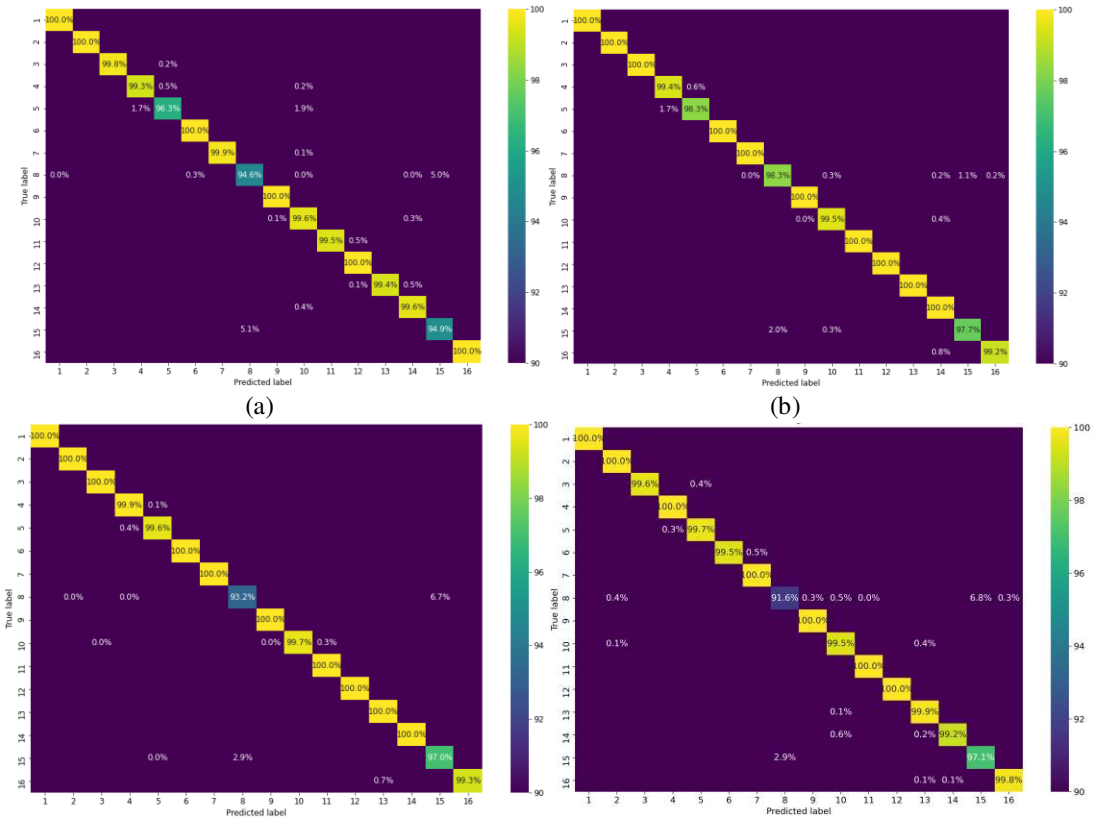


Fig. 4. Comparison of a confusion matrix for classification results on SA data set. (a) 2D-CNN. (b) 3D-CNN. (c) HybridSN. (a) 3D-2D-CNN.

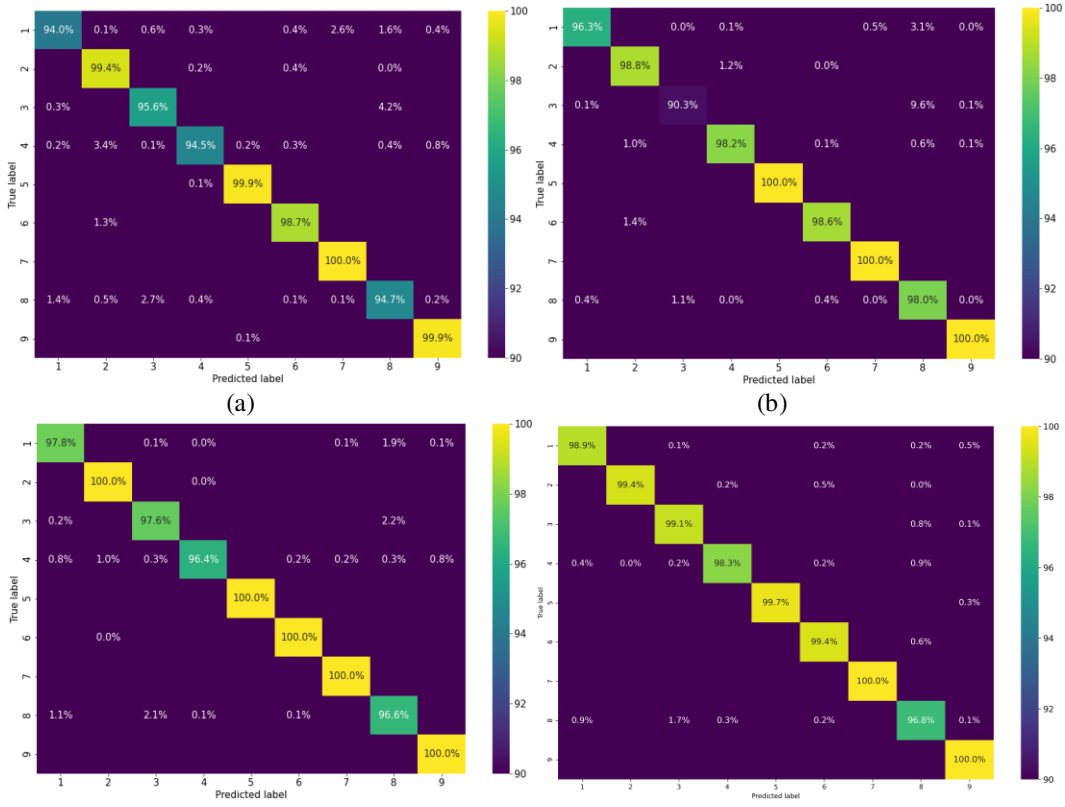


Fig. 5. Comparison of a confusion matrix for classification results on PU data set. (a) 2D-CNN. (b) 3D-CNN. (c) HybridSN. (a) 3D-2D-CNN.

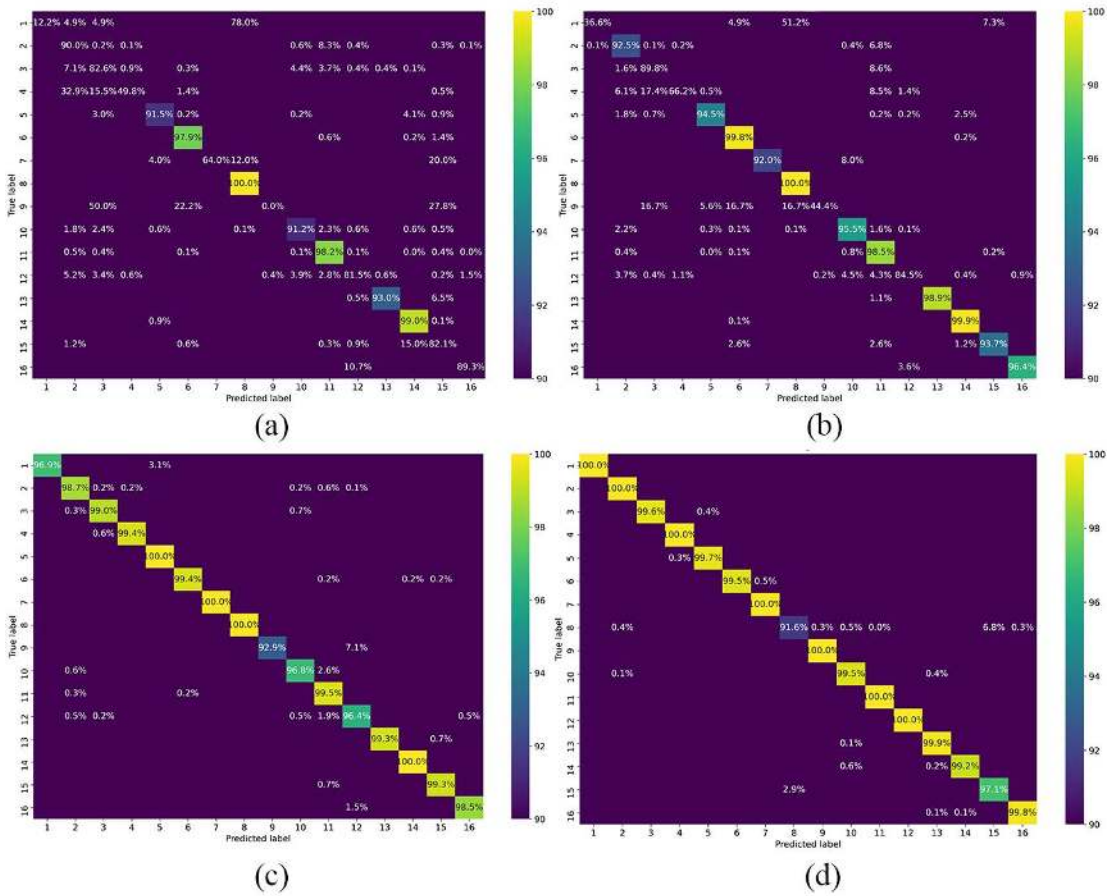


Fig. 6. Comparison of a confusion matrix for classification results on IP dataset. (a) 2D-CNN. (b) 3D-CNN. (c) HybridSN. (a) 3D-2D-CNN.

B. Experiment 2: The sensitivity of the hybrid model to the number of training samples

Experiments that increase the number of training samples per every class of dataset. One of the critical factors in the training of convolutional neural networks is the number of training samples. It is generally clear that a well-known CNN may not extract useful features unless there are many training samples available. However, having a large number of training samples for hyperspectral images is not common, so building robust and efficient networks for classification is very important. In this section, the effect of the number of training samples on the accuracy of the three data sets is also tested. Table VII shows the results obtained for the Salinas dataset, where the combined method with the lowest number of training samples (50) works better than SVM, 2D-CNN, 3D-CNN and HybridSN with 8.73, 3.11, and 1.82 and 1.42%, respectively. According to Fig. 10. (a) the more training samples we provide to models, especially neural networks, the better their accuracy. If we focus on the results obtained for the Pavia data set with 50 training samples (Table IX, the hybrid model works 19.61% better than the SVM spectral method, as well as the hybrid model of 2D-CNN, 3D-CNN and HybridSN spectral-spatial classification) In terms of OA, it performs better by 5.07, 4.26 and 2.24%, respectively. To better illustrate the accuracy of the model's accuracy with different training samples, it is given in Fig. 10. (b). Table X shows that if Indian Pines dataset more training samples are available, method 3D-2D CNN can be 99% accurate. However, with 5% of the training samples, the proposed method obtained an accuracy of 93.27%, which is 0.94% higher than the second accuracy (HybridSN).

According to Figures 10. (a)-(b) When the number of training samples is small, the model cannot extract useful information from the little data available and causes a higher error rate of classes that are complex and similar, resulting in poor model performance and classification accuracy decreases. Also, as the number of training samples increases, the model's ability to learn from it improves, indicating that more training samples can give more information to the models to extract better features. The good performance of HybridSN with limited training samples shows the importance of network optimization and the potential of 3D-2D-CNN in hyperspectral classification. The hybrid method obtained the best accuracy with a different number of training samples in three datasets.

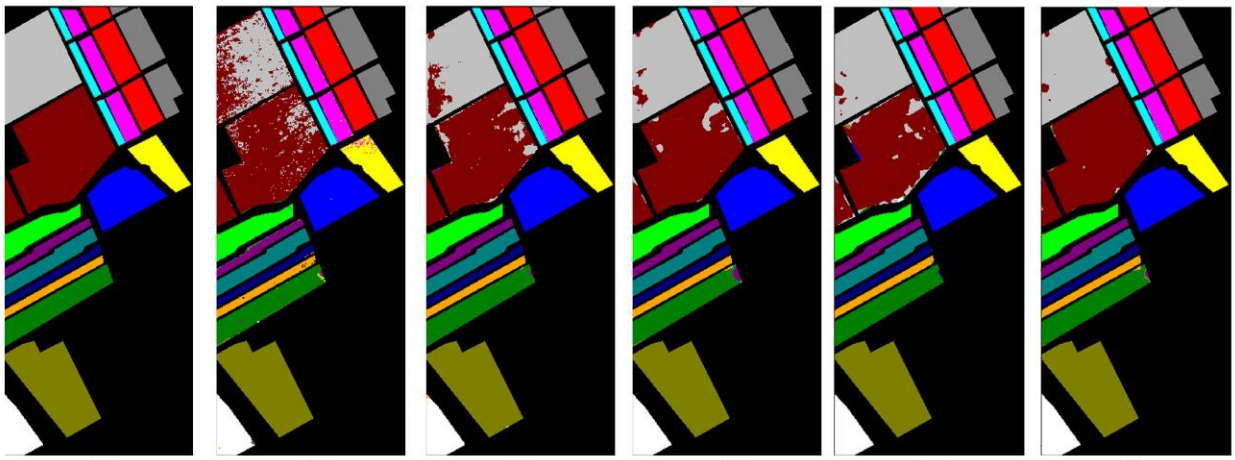
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTARS.2021.3099118, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing

TABLE VI. CLASSIFICATION ACCURACY ON THE PAVIA UNIVERSITY DATASET (100 SAMPLES FROM EACH CLASS FOR TRAINING)

Class	SVM	2D CNN	3D CNN	HybridSN	3D-2D CNN
1	75.37±2.18	94.84±1.94	96.39±1.90	97.58±1.39	**98.61±0.38**
2	79.40±1.39	97.23±1.40	97.70±1.36	98.80±0.91	**99.32±0.53**
3	70.68±1.83	93.66±2.71	92.68±2.94	96.06±1.51	**98.46±0.48**
4	93.90±1.21	97.56±0.87	97.34±1.40	97.39±1.18	**98.37±0.69**
5	99.92±0.07	100.0±0	100.0±0	100.0±0.0	99.89±0.15
6	89.16±1.39	97.64±1.94	98.46±1.02	**99.70±0.33**	98.93±0.40
7	86.05±1.95	99.58±0.49	99.84±0.36	99.91±0.09	**100.0±0.0**
8	82.53±1.61	92.87±3.26	95.34±1.72	96.41±1.82	**96.94±1.09**
9	**99.93±0.06**	99.38±0.57	99.56±0.51	99.57±0.57	99.91±0.11
OA	82.02±0.74	96.55±0.66	97.27±0.70	98.37±0.41	**98.90±0.14**
Kappa	76.88±0.89	95.42±0.87	96.38±0.93	97.83±0.55	**98.53±0.20**
AA	86.32±0.52	96.94±0.48	97.45±0.42	98.38±0.21	**98.94±0.17**

TABLE VII. CLASSIFICATION ACCURACY ON THE INDIAN PINES DATASET (10% SAMPLES FOR TRAINING)

Class	SVM	2D CNN	3D CNN	HybridSN	3D-2D CNN
1	0.0±0.0	21.27±15.65	39.51±23.69	**85.05±8.04**	81.70±7.24
2	58.82±2.13	87.85±1.97	91.16±1.6	**94.73±1.17**	93.35±1.50
3	27.83±7.10	79.94±6.69	86.54±6.19	97.85±1.44	**97.99±1.11**
4	17.73±0.0	38.72±13.67	58.87±9.26	93.80±3.68	**96.71±4.64**
5	86.01±2.84	93.47±2.76	96.08±2.28	**99.69±0.21**	97.28±1.82
6	93.25±2.84	98.44±0.68	98.98±0.56	98.96±0.87	**99.03±0.62**
7	0.0±0.0	36.0±26.70	56.64±38	75.2±26.93	**81.0±21.86**
8	99.31±1.42	99.24±0.96	99.84±0.23	99.84±0.22	**100.0±0.0**
9	11.76±3.55	19.33±17.64	20.44±21.64	**72.88±4.53**	66.55±7.91
10	44.39±7.10	90.34±3.39	94.89±1.78	**98.13±1.87**	97.66±1.24
11	85.75±0.01	96.28±1.48	98.34±0.54	98.88±0.57	**98.96±0.44**
12	23.97±0.0	70.25±8.30	81.65±5.7	**97.19±1.15**	94.51±3.03
13	93.78±2.84	93.66±4.71	97.81±1.86	98.19±1.83	**98.54±1.21**
14	96.33±1.42	99.05±0.64	98.78±0.82	99.70±0.22	**99.75±0.33**
15	32.95±7.1	80.08±12.93	86.83±5.83	**99.42±0.81**	97.86±1.41
16	79.27±0.0	72.57±19.87	84.71±8.2	88.92±7.39	**92.06±5.35**
OA	68.02±0.0	89.36±1.59	93.15±1.2	97.09±0.19	**97.14±0.34**
Kappa	62.65±0.0	87.79±1.84	92.14±1.39	96.82±0.22	**96.73±0.39**
AA	53.20±0.0	73.53±4.05	80.69±5.88	94.96±0.57	**89.06±3.02**



(a) (b) (c) (d) (e) (f)

Fig. 7. Classification maps results on SA dataset: (a) Ground truth, (b) SVM, (c) 2D CNN, (d) 3D CNN, (e) HybridSN (f) 3D-2D CNN

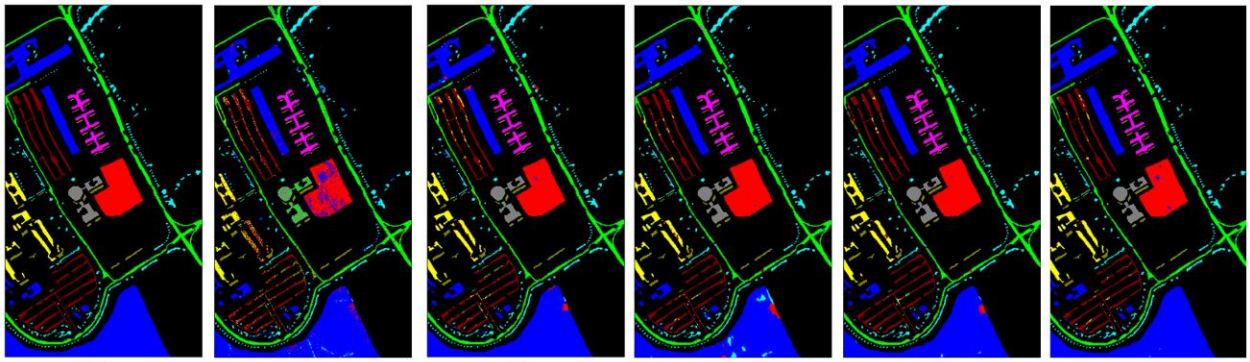


Fig.8. Classification maps results on PU dataset: (a) Ground truth, (b) SVM, (c) 2D CNN, (d) 3D CNN, (e) HybridSN (f) 3D-2D CNN.

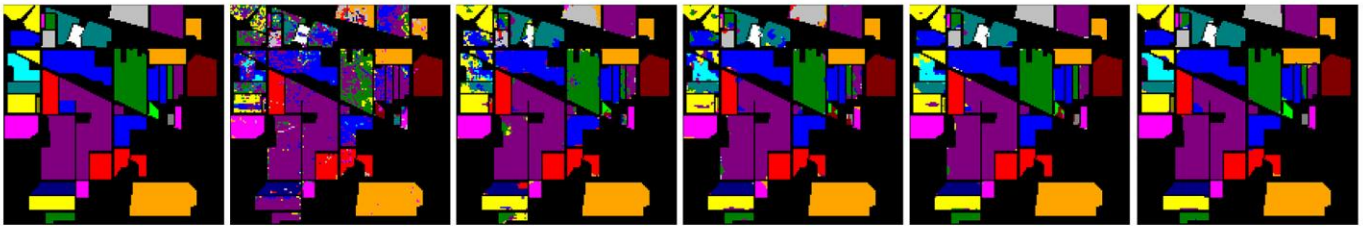


Fig. 9. Classification maps results on IP dataset: (a) Ground truth, (b) SVM, (c) 2D CNN, (d) 3D CNN, (e) HybridSN (f) 3D-2D CNN.

TABLE VII. COMPARISON OF THE CLASSIFICATION ACCURACY OF SA DATA SET WITH DIFFERENT TRAINING SAMPLES.

Tr-Samples	50			100			200			500		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	89.14	87.91	93.96	90.27	89.14	94.89	91.07	90.01	95.81	91.67	90.57	96.57
2D CNN	94.76	94.17	97.72	96.76	96.39	98.66	98.80	98.66	99.53	99.33	99.24	99.77
3D CNN	96.05	95.61	98.43	97.68	97.41	99.08	99.04	98.92	99.59	99.56	99.50	99.83
HybridSN	96.45	96.05	98.74	98.41	98.23	99.35	99.23	99.13	99.66	99.65	99.60	99.87
3D-2D CNN	**97.87**	**97.63**	**99.09**	**99.07**	**98.96**	**99.54**	**99.35**	**99.18**	**99.74**	**99.70**	**99.66**	**99.89**

TABLE IX. COMPARISON OF THE CLASSIFICATION ACCURACY OF PU DATA SET WITH DIFFERENT TRAINING SAMPLES.

Tr-Samples	50			100			200			500		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	78.53	72.63	83.38	82.02	76.88	86.33	85.52	81.04	88.28	88.01	83.83	90.21
2D CNN	93.07	90.90	94.34	96.55	95.42	96.94	98.66	98.20	98.57	99.71	99.60	99.73
3D CNN	93.88	91.91	94.22	97.27	96.38	97.45	98.01	97.33	98.29	99.67	99.55	99.67
HybridSN	95.90	94.59	96.07	98.16	97.56	98.23	99.29	99.04	99.23	99.83	99.77	99.84
3D-2D CNN	**98.14**	**97.53**	**97.97**	**98.90**	**98.53**	**98.94**	**99.45**	**99.25**	**99.51**	**99.87**	**99.82**	**99.84**

TABLE X. COMPARISON OF THE CLASSIFICATION ACCURACY OF IP DATA SET WITH DIFFERENT TRAINING SAMPLES.

Tr-Samples	%5			%10			%20			%30		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	61.18	53.84	45.52	68.02	62.65	53.20	70.94	66.20	56.17	74.18	70.13	65.05
2D CNN	77.92	74.46	57.21	89.36	87.79	73.53	91.09	89.77	76.33	97.28	96.89	90.90
3D CNN	83.33	80.76	64.73	93.15	92.14	80.69	96.77	96.31	88.06	98.42	98.20	94.44
HybridSN	92.33	91.39	87.28	97.09	96.73	94.96	98.11	97.41	96.27	**99.64**	**99.59**	**99.32**
3D-2D CNN	**93.27**	**92.54**	**89.10**	**97.14**	**96.82**	**95.06**	**98.85**	**97.85**	**96.71**	99.04	98.91	95.81

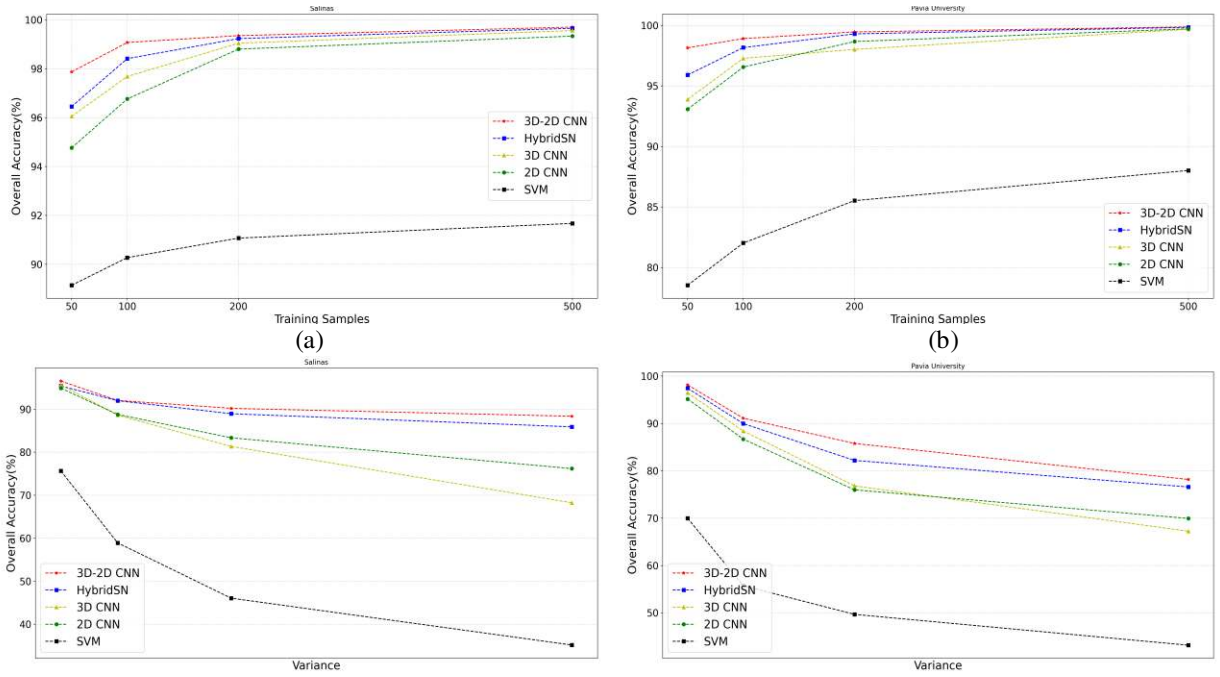


Fig. 10. (a) Overall accuracy (%) with different training samples for Salinas data set. (b) Overall accuracy (%) with different samples of training for Salinas data set. (c) Display overall accuracy (%) with a percentage of noise added to 100 of the Salinas data set training sample. (d) Display overall accuracy (%) with a percentage of noise added to 100 of the Pavia data set training sample.

C. *Experiment 3: Add Gaussian noise to hyperspectral images*

Gaussian noise is applied to all available hyperspectral image bands with zero mean and different variance on Salinas and Pavia datasets and the results are given in Tables XI - XIII.

$$PG(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \quad (7)$$

where z is the pixel intensity, σ is the variance and μ is the mean, which is a random amount of noise with a normal distribution $N(0, \sigma^2)$. To see the amount of noise added to the image, one of the bands of the original image (25) from the Salinas and Pavia datasets with noise applied with different variance on the same image is shown in Fig. 11 and Fig. 12. As we increase the percentage of noise added to the image, it makes the images appear dimmer and blurry.

TABLE XI. CLASSIFICATION ACCURACY (%) WITH A PERCENTAGE OF NOISE ADDED TO 100 OF THE SA DATA SET TRAINING SAMPLE.

Noise	Variance = 2			Variance = 4			Variance = 6			Variance = 8		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	75.63	72.96	82.46	58.91	54.67	63.72	46.05	41.10	49.21	35.16	29.92	37.18
2D CNN	94.88	94.28	97.81	88.77	87.48	95.00	83.34	81.48	90.66	76.18	73.59	83.92
3D CNN	95.63	95.13	98.29	88.62	87.33	95.05	81.33	79.25	88.29	68.24	64.85	75.06
HybridSN	95.39	94.39	98.27	91.97	91.05	96.29	88.94	87.69	94.42	85.89	84.38	89.53
3D-2D CNN	**96.52**	**96.12**	**98.63**	**92.03**	**91.12**	**96.63**	**90.19**	**89.08**	**95.26**	**88.35**	**87.73**	**93.62**

TABLE XII. CLASSIFICATION ACCURACY (%) WITH A PERCENTAGE OF NOISE ADDED TO 100 OF THE PU DATA SET TRAINING SAMPLE.

Noise	Variance = 2			Variance = 4			Variance = 6			Variance = 8		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	69.96	62.20	76.74	55.72	45.92	65.89	49.66	39.25	60.83	43.15	32.09	54.51
2D CNN	95.16	93.57	95.51	86.68	82.61	89.42	75.97	69.06	81.17	69.93	62.18	77.23
3D CNN	96.47	95.31	96.77	88.39	84.75	90.37	76.81	70.22	82.02	67.23	58.94	74.85
HybridSN	97.43	96.59	97.72	89.96	86.78	92.21	82.18	78.82	86.12	76.58	70.00	82.04
3D-2D CNN	**98.13**	**97.50**	**98.04**	**91.10**	**88.30**	**93.06**	**85.77**	**81.41**	**89.59**	**78.16**	**71.38**	**83.89**

TABLE XIII. CLASSIFICATION ACCURACY (%) WITH A PERCENTAGE OF NOISE ADDED TO %10 OF THE IP DATA SET TRAINING SAMPLES.

Noise	Variance = 2			Variance = 4			Variance = 6			Variance = 8		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	62.39	55.59	47.81	50.69	40.25	33.79	41.56	27.91	19.45	36.90	21.22	14.11
2D CNN	87.71	85.89	70.40	77.64	74.08	57.40	63.16	56.79	40.73	54.68	46.67	31.77
3D CNN	92.72	91.66	82.44	81.77	78.87	62.33	60.70	53.29	39.40	51.24	41.99	28.30
HybridSN	96.21	95.81	95.57	94.09	93.68	89.97	90.14	88.69	81.37	80.62	78.61	76.23
3D-2D CNN	**97.36**	**96.70**	**97.66**	**95.91**	**94.33**	**91.90**	**91.54**	**89.26**	**86.92**	**84.51**	**81.86**	**79.60**

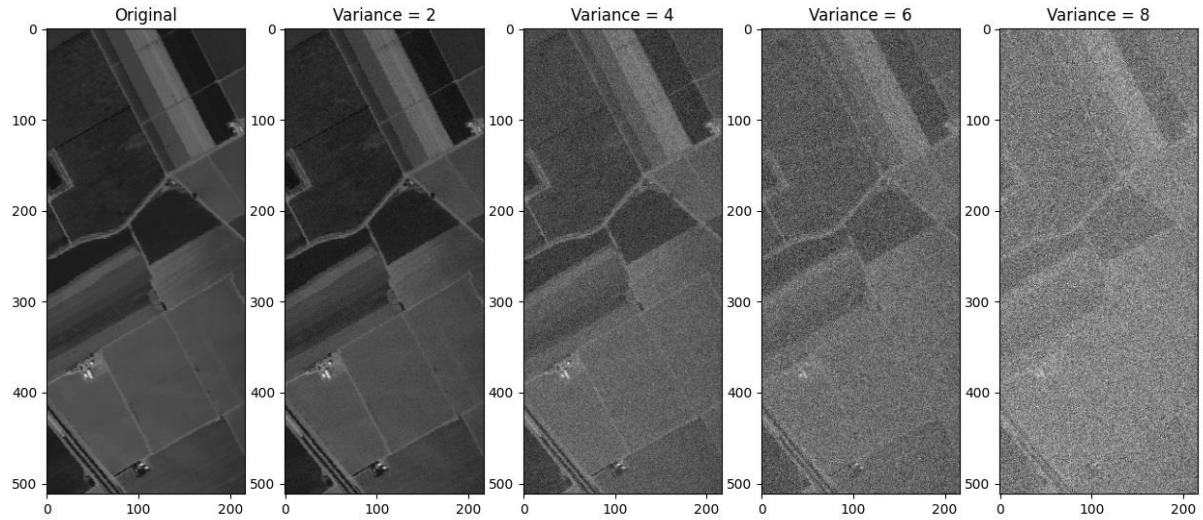


Fig. 11. Display one of the bands (25) of the hyperspectral image with the percentage of noise added to the Salinas data set.

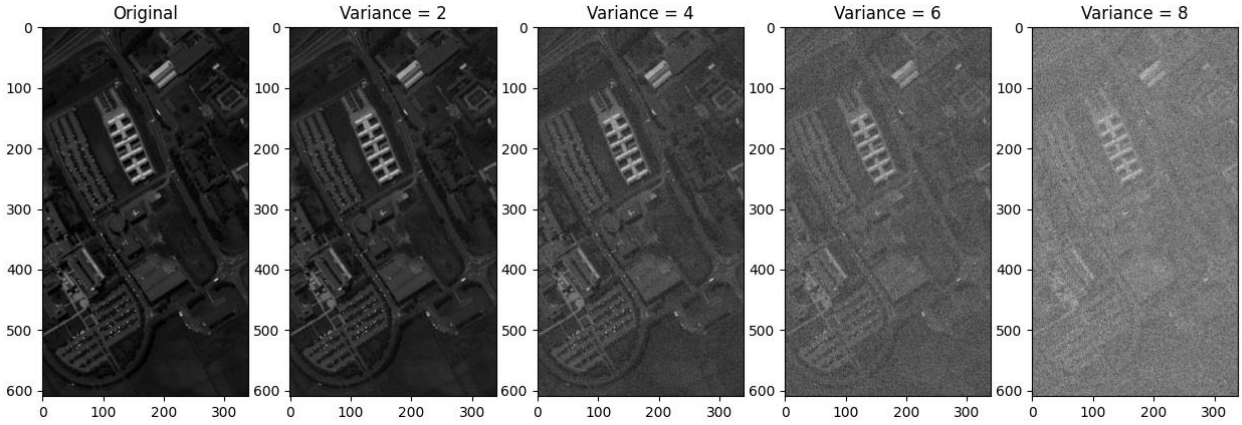


Fig. 12. Display one of the bands (25) of the hyperspectral image with percentage of noise added in Pavia data set.

According to the results obtained in Tables XI-XII, the 3D-2D CNN method with a limited number of training samples and a high-noise image has a much better performance than other methods. In Table XI the best accuracy is related to the 3D-2D CNN method with 88.35%, which is 2.46% better than the second accuracy, associated with the HybridSN method. As in the SA dataset, the 3D-2D CNN method has a high performance against noise, in the PU and IP dataset, it performs better with other methods. As shown in Figs. 10. (c)-(d) You can see that the methods perform poorly with the increasing percentage of noise. With the addition of noise to the image, the image pixels in different bands practically change. The SVM method, which relies only on spectral information of the image, has a low accuracy of 50% against high noise. However, deep learning models, due to their hierarchical structure, can reduce noise to some extent and are resistant to the noise by extracting high-level features. Although the 3D CNN model extracts spectral and spatial features, it is computationally complex. The limited training samples of the 3D CNN model are more tendency to overfitting.

D. Experiment 4: The effect of the number of training samples on image noise

According to the results obtained in Tables XIV-XVI, it shows that if more training samples are provided to neural network models, they can classify noise images with high accuracy. If the number of training samples is reduced, many trainable parameters will not be adequately trained and the network will have overfitting problems and the classification accuracy will be significantly reduced. Because the hybrid model has fewer trainable parameters and computational complexity than the 3D CNN model, it is less tendency to overfitting. On the other hand, the model has a hybrid structure and can extract more specific information than the 2D CNN model. In Fig. 13 of the a-c diagrams, in the worst-case scenario, when we see limitations in training samples and a lot of noise is added to the image, all models are overfitting. As can be seen in Fig. 13 (b), the 3D-CNN model is overfitting in the early epochs of training, one of the reasons being the high number of computational complexity compared to other models tested. Fig. 13 (c) shows diagrams a and c as expected, and the 2D-CNN and 3D-2D-CNN models also have overfitting problems, but they have been trained in more epochs than the 3D-CNN model. Fig. 13 (d)-(f) diagrams in the best possible case where there are a sufficient number of training samples, the models are well trained and have a minimal error, as a result of which the models offer high classification accuracy.

TABLE XIV. CLASSIFICATION ACCURACY (%) AND NOISE ADDED WITH VARIANCE 6 ON A DIFFERENT NUMBER OF TRAINING SAMPLES OF SA DATA SET.

Tr-Samples	50			100			200			500		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	42.58	37.43	44.08	46.05	41.10	49.21	48.76	43.93	53.37	51.29	46.31	56.80
2D CNN	74.91	72.11	80.58	83.34	81.48	90.66	89.57	88.36	95.35	95.96	95.44	98.43
3D CNN	72.19	69.27	79.69	81.33	79.25	88.29	85.56	83.87	92.24	91.20	90.00	96.28
HybridSN	81.81	79.78	88.94	88.94	87.69	94.42	93.06	92.24	97.07	98.76	98.34	99.25
3D-2D CNN	**85.53**	**83.95**	**92.63**	**90.19**	**89.08**	**95.26**	**94.88**	**94.27**	**97.75**	**99.37**	**99.29**	**99.76**

TABLE XV. CLASSIFICATION ACCURACY (%) AND NOISE ADDED WITH VARIANCE 6 ON A DIFFERENT NUMBER OF TRAINING SAMPLES OF PU DATA SET.

Tr-Samples	50			100			200			500		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	47.86	37.01	57.80	49.02	38.71	60.77	51.63	41.23	62.89	52.67	41.23	62.89
2D CNN	68.00	58.98	71.55	75.97	69.06	81.17	82.57	77.40	88.37	92.34	89.59	95.78
3D CNN	65.67	57.14	73.03	76.81	70.22	82.02	81.28	76.14	88.42	90.43	87.04	93.77
HybridSN	74.31	67.05	78.72	82.18	78.82	86.12	89.28	88.05	90.28	95.79	94.34	96.74
3D-2D CNN	**74.73**	**67.56**	**80.07**	**85.77**	**81.41**	**89.59**	**91.60**	**90.70**	**93.73**	**96.36**	**94.96**	**98.29**

TABLE XVI. CLASSIFICATION ACCURACY (%) AND NOISE ADDED WITH VARIANCE 6 ON A DIFFERENT NUMBER OF TRAINING SAMPLES OF IP DATA SET.

Tr-Samples	%5			%10			%20			%30		
Methods	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
SVM	38.38	23.54	15.59	41.56	27.91	19.45	44.27	31.96	23.18	45.34	33.35	24.76
2D CNN	51.36	42.16	29.21	63.16	56.79	40.73	75.92	72.17	55.62	91.31	90.04	75.67
3D CNN	50.35	40.22	27.45	60.70	53.29	39.40	73.45	68.90	53.00	90.18	88.71	74.42
HybridSN	81.90	79.15	65.95	90.14	88.69	81.37	95.56	93.42	93.85	98.41	98.19	97.27
3D-2D CNN	**85.72**	**81.31**	**76.67**	**91.54**	**89.26**	**86.92**	**97.41**	**97.19**	**96.27**	**99.14**	**99.02**	**98.20**

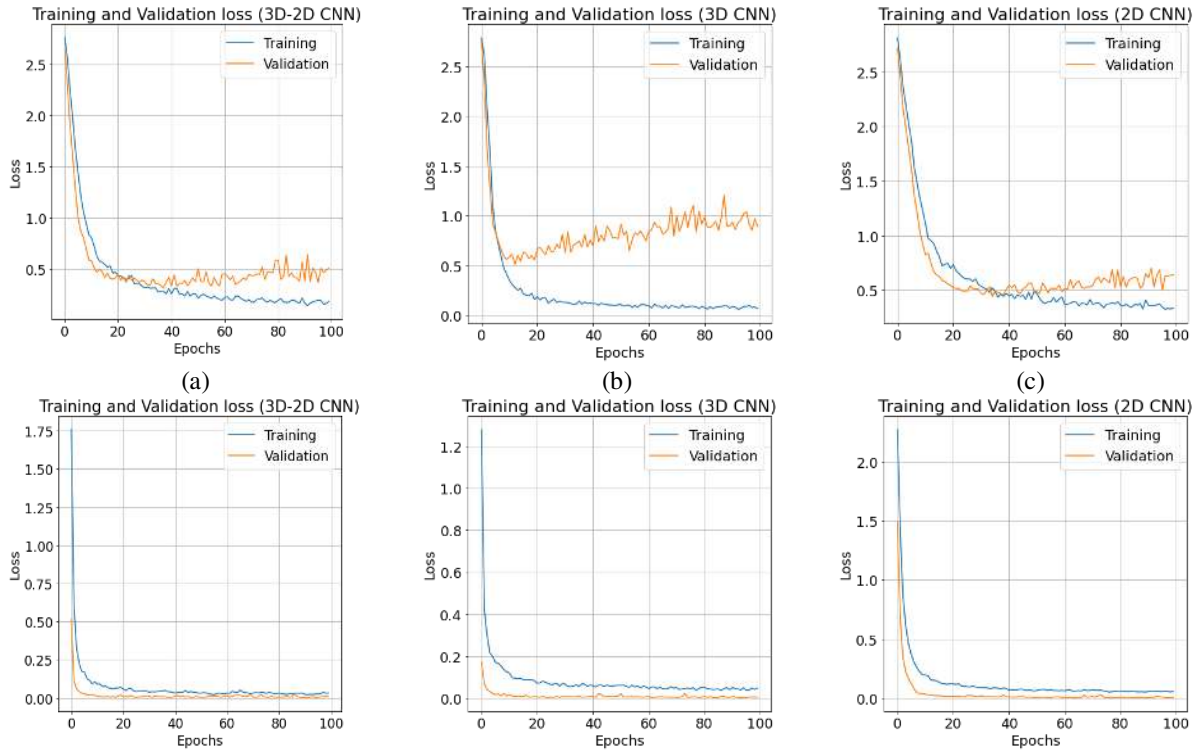


Fig. 13. The a-c error diagrams show the 2D-CNN, 3D-CNN, and 3D-2D-CNN models with a training sample of 100 and a variance of 8, respectively. The d-f error diagrams show the 2D-CNN, 3D-CNN, and 3D-2D-CNN models with a training instance of 500, respectively.

E. Experiment 5: The effect of the spatial size of the input cube on the classification accuracy

The classification accuracy of CNN models depends on the size of the entrance window. If the size of the input window is too small, there is not enough information in the image to extract the feature, as a result, some information is lost and the ability to classify is reduced. If the size of the input window is too large, it may contain pixels of different classes and additional noise will enter the input window, resulting in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTARS.2021.3099118, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing

reduced classification accuracy. Therefore, the classification function has been analyzed to find the optimal spatial size of the input cube. According to Fig. 14, when the spatial window of the input is 15×15, the evaluation indicators reach the desired values in the two data sets. Therefore, the spatial window of 15×15 is considered the most appropriate spatial size of the input cube in the conditions that the hardware platform allows.

F. Experiment 5: The effect of dimension reduction on the classification accuracy

In order to determine the number of principal components in PCA, we test the classification accuracy on the three datasets after the dimensionality reduction with different principal components. Among them, due to the large spatial scale of PU and SA, only 10, 15, and 20 are considered, while seven cases from 10 to 40 are tested in IP. The classification accuracy with different numbers of principal components among the three datasets is shown in Fig. 15.

In Fig. 15, according to the final classification accuracy, we set the number of reduced spectral bands to 15, 15 and 30 for PU, SA and IP, respectively.

Tables XVII-XIX show a qualitative summary of the experiments performed, which is the number of different training samples per percentage of noise added to the methods. In these tables, the accuracy is more than 90% green, the accuracy is 70-90% blue and the accuracy is less than 70% red. According to the results of the traditional SVM method, it has a lower performance against noise than other models of deep learning. Besides, as we increase the number of training samples, unlike deep learning models, the SVM method cannot extract features from noise data, in practice, its performance does not have much effect on classification accuracy by increasing training samples. According to the qualitative results of the tables, two case can be mentioned for deep learning models that have a significant impact on the accuracy of classification:

(1) Number of training samples: In all deep learning models, the more training samples, the more we can increase the number of network layers and deepen so that the network extracts useful and practical spectral-spatial information from data and feature maps. But as the number of trainable parameters increases, so does the network training time and tendency to overfitting.

(2) Computational complexity: If the data is too noisy and we have limited training samples, a model with less computational complexity can provide better performance. According to Tables XII and XIII, in the 3D-CNN model, three datasets have poor performance due to high noise percentage and limited training samples. The 2D-CNN model outperforms the 3D-CNN because it can extract more distinctive information than other models and has much less computational complexity than the 3D-CNN model.

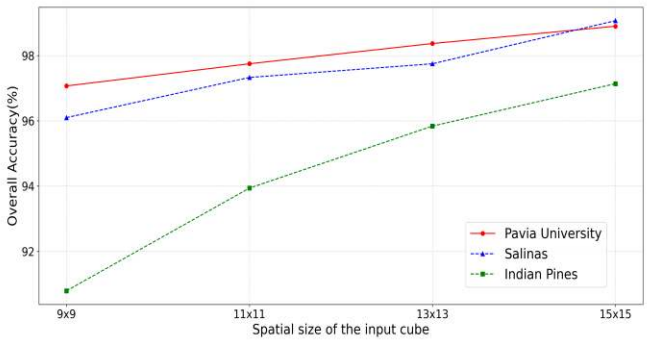


Fig. 14. The effect of different input cube sizes on accuracy with 100 training samples with PU, SA and IP datasets.

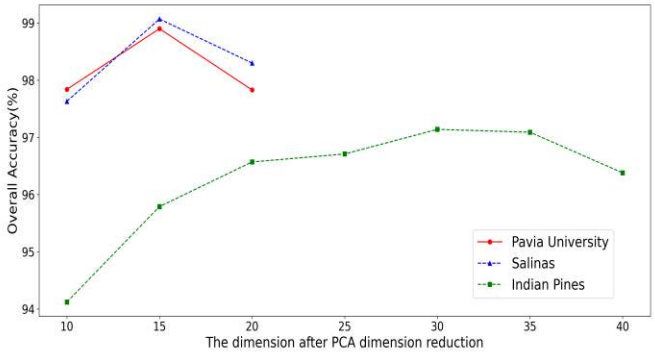


Fig. 15. The effect of different input cube sizes on accuracy with 100 training samples with PU, SA and IP datasets.

TABLE XVII. QUALITATIVE RESULTS OF THE NUMBER OF DIFFERENT TRAINING SAMPLES AGAINST THE PERCENTAGE OF NOISE ADDED TO THE SA DATA SET.

Tr-samples	100				200				500			
Methods	2	4	6	8	2	4	6	8	2	4	6	8
SVM												
2D CNN												
3D CNN												
HybridSN												
3D-2D CNN												

TABLE XVIII. QUALITATIVE RESULTS OF THE NUMBER OF DIFFERENT TRAINING SAMPLES AGAINST THE PERCENTAGE OF NOISE ADDED TO THE DATA SET OF PU.

Tr-samples	100				200				500			
Methods	2	4	6	8	2	4	6	8	2	4	6	8
SVM												
2D CNN												
3D CNN												
HybridSN												
3D-2D CNN												

TABLE XIX. QUALITATIVE RESULTS OF THE NUMBER OF DIFFERENT TRAINING SAMPLES AGAINST THE PERCENTAGE OF NOISE ADDED TO THE DATA SET OF IP.

Tr-samples	%10				%20				%30			
Methods	2	4	6	8	2	4	6	8	2	4	6	8
SVM												
2D CNN												
3D CNN												
HybridSN												
3D-2D CNN												

IV. CONCLUSION

The availability of hyperspectral images is deficient and there is limited data. One of the challenges in classifying hyperspectral images is designing a model that fits the situation. This report presents a hybrid model of 3D and 2D convolution for HSI classification. Spatial and spectral features can be used to increase classification performance. The hybrid model combines spatial-spectral and spatial information obtained from 3D and 2D convolution, respectively. The combining of 3D-CNN and 2D-CNN reduces the number of learning parameters and is computationally less complex than using 3D-CNN alone. Network optimization is better done using Adam optimizer and reduces training time. To limit the number of training samples and noise, the hybrid model has the best performance compared to other models. When we have a sufficient number of training samples, we can increase the number of layers of the model and deepen the network. All models offer high accuracy in case we have enough training samples, still hybrid model has fewer parameters and its training time is lower than using only the 3D-CNN model and compared to the 2D-CNN model due to the hybrid structure that it can make the most of all spectral and spatial information in HSI data. It is therefore economical to use a hybrid model for classification for hyperspectral images. Experiments on three datasets were compared with three classification methods, which confirms the superiority of the proposed method in the case of limited training sample and noise.

REFERENCES

[1] W. Lv and X. Wang, "Overview of Hyperspectral Image Classification," *J. Sensors*, vol. 2020, p. 4817234, 2020, doi: 10.1155/2020/4817234.

[2] Y. Cai, X. Liu, and Z. Cai, "BS-Nets: An End-to-End Framework for Band Selection of Hyperspectral Image," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 3, pp. 1969–1984, 2020, doi: 10.1109/TGRS.2019.2951433.

[3] D. Hong *et al.*, "More Diverse Means Better: Multimodal Deep Learning Meets Remote-Sensing Imagery Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 5, pp. 4340–4354, 2021, doi: 10.1109/TGRS.2020.3016820.

[4] K. G. Willson, L. E. Cox, J L Hart, and D. C. Dey, "Three-dimensional light structure of an upland Quercus stand post-tornado disturbance," *Journal of Forestry Research*, vol. 31, No. 1, pp. 141–153, 2020, doi: 10.1007/s11676-019-00907-y.

[5] A. Sharifi, "Yield prediction with machine learning algorithms and satellite images," *J. Sci. Food Agric.*, vol. 101, pp. 891–896, Feb. 2021, doi: 10.1002/jsfa.10696.

[6] B. Heinrichs and S. B. Eickhoff, "Your evidence? Machine learning algorithms for medical diagnosis and prediction," *Hum. Brain Mapp.*, vol. 41, no. 6, pp. 1435–1444, 2020, doi: 10.1002/hbm.24886.

[7] A. Sharifi, "Using Sentinel-2 Data to Predict Nitrogen Uptake in Maize Crop," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 2656–2662, 2020, doi: 10.1109/JSTARS.2020.2998638.

[8] A. Sharifi and M Hosseingholizadeh, "Application of sentinel-1 data to estimate height and biomass of rice crop in Astaneh-ye Ashrafiyeh, Iran," *J. Indian Soc. Remote Sens.*, vol. 48, pp. 11–19, Oct. 2019, doi: 10.1007/s12524-019-01057-8.

[9] A. Sharifi, J. Amini, and F. Pourshakouri, "Development of an allometric model to estimate above-ground biomass of forests using MLPNN algorithm, case study: Hyrcanian forests of Iran," *Casp. J. Environ. Sci.*, vol. 14, Jan. 2015.

[10] Y. Xu, B. Du, and L. Zhang, "Beyond the Patchwise Classification: Spectral-Spatial Fully Convolutional Networks for Hyperspectral Image Classification," *IEEE Trans. Big Data*, vol. 6, no. 3, pp. 492–506, 2020, doi: 10.1109/TBDATA.2019.2923243.

[11] Jing Wang and Chein-I Chang, "Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 6, pp. 1586–1600, Jun. 2006, doi: 10.1109/TGRS.2005.863297.

[12] A. A. Nielsen, "Kernel Maximum Autocorrelation Factor and Minimum Noise Fraction Transformations," *IEEE Trans. Image Process.*, vol. 20, no. 3, pp. 612–624, Mar. 2011, doi: 10.1109/TIP.2010.2076296.

[13] T. V Bandos, L. Bruzzone, and G. Camps-Valls, "Classification of Hyperspectral Images With Regularized Linear Discriminant Analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 3, pp. 862–873, Mar. 2009, doi: 10.1109/TGRS.2008.2005729.

[14] A. Kosari, A. Sharifi, A. Ahmadi, and M. Khoshsima, "Remote sensing satellite's attitude control system: rapid performance sizing for passive scan imaging mode," *Aircr. Eng. Aerosp. Technol,* vol. 92, no. 7, pp. 1073-1083, Jun. 2020, doi: 10.1108/AEAT-02-2020-0030.

[15] L. Bruzzone, F. Roli, and S. B. Serpico, "An extension of the Jeffreys-Matusita distance to multiclass cases for feature selection," *IEEE Trans. Geosci. Remote Sens.*, vol. 33, no. 6, pp. 1318–1321, Nov. 1995, doi: 10.1109/36.477187.

[16] N. Keshava, "Distance metrics and band selection in hyperspectral processing with applications to material identification and spectral libraries," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 7, pp. 1552–1565, Jul. 2004, doi: 10.1109/TGRS.2004.830549.

[17] E. Blanzieri and F. Melgani, "Nearest Neighbor Classification of Remote Sensing Images With the Maximal Margin Principle," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 6, pp. 1804–1811, Jun. 2008, doi: 10.1109/TGRS.2008.916090.

[18] J. Xia, L. Bombrun, Y. Berthoumieu, C. Germain, and P. Du, "Spectral-spatial Rotation Forest for hyperspectral image classification," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Jul. 2016, pp. 5126–5129, doi: 10.1109/IGARSS.2016.7730336.

[19] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*,

vol. 42, no. 8, pp. 1778–1790, Aug. 2004, doi: 10.1109/TGRS.2004.831865.

[20] S. Jia, L. Shen, J. Zhu, and Q. Li, "A 3-D Gabor Phase-Based Coding and Matching Framework for Hyperspectral Imagery Classification," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1176–1188, 2018, doi: 10.1109/TCYB.2017.2682846.

[21] D. Hong, X. Wu, P. Ghamisi, J. Chanussot, N. Yokoya, and X. X. Zhu, "Invariant Attribute Profiles: A Spatial-Frequency Joint Feature Extractor for Hyperspectral Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 6, pp. 3791–3808, 2020, doi: 10.1109/TGRS.2019.2957251.

[22] L. Zhang, L. Zhang, and B. Du, "Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016, doi: 10.1109/MGRS.2016.2540798.

[23] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep Convolutional Neural Networks for Hyperspectral Image Classification," *J. Sensors*, vol. 2015, p. 258619, 2015, doi: 10.1155/2015/258619.

[24] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep Recurrent Neural Networks for Hyperspectral Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, 2017, doi: 10.1109/TGRS.2016.2636241.

[25] Y. Li, W. Xie, and H. Li, "Hyperspectral image reconstruction by deep convolutional neural network for classification," *Pattern Recognit.*, vol. 63, pp. 371–383, 2017, doi: https://doi.org/10.1016/j.patcog.2016.10.019.

[26] X. Cao, F. Zhou, L. Xu, D. Meng, Z. Xu, and J. Paisley, "Hyperspectral Image Classification With Markov Random Fields and a Convolutional Neural Network," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2354–2367, 2018, doi: 10.1109/TIP.2018.2799324.

[27] S. Hao, W. Wang, Y. Ye, E. Li, and L. Bruzzone, "A Deep Network Architecture for Super-Resolution-Aided Hyperspectral Image Classification With Classwise Loss," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4650–4663, 2018, doi: 10.1109/TGRS.2018.2832228.

[28] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016, doi: 10.1109/TGRS.2016.2584107.

[29] Y. Li, H. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery with 3D convolutional neural network," *Remote Sens.*, vol. 9, no. 1, 2017, doi: 10.3390/rs9010067.

[30] J. Feng, H. Yu, L. Wang, X. Cao, X. Zhang, and L. Jiao, "Classification of Hyperspectral Images Based on Multiclass Spatial–Spectral Generative Adversarial Networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5329–5343, 2019, doi: 10.1109/TGRS.2019.2899057.

[31] A. Sellami, M. Farah, I. Riadh Farah, and B. Solaiman, "Hyperspectral imagery classification based on semi-supervised 3-D deep neural network and adaptive band selection," *Expert Syst. Appl.*, vol. 129, pp. 246–259, 2019, doi: https://doi.org/10.1016/j.eswa.2019.04.006.

[32] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph Convolutional Networks for Hyperspectral Image Classification," *IEEE Trans. Geosci. Remote Sens.*, pp. 1–13, 2020, doi: 10.1109/TGRS.2020.3015157.

[33] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "Deep&Dense Convolutional Neural Network for Hyperspectral Image Classification," *Remote Sens.*, vol. 10, p. 1454, 2018, doi: 10.3390/rs10091454.

[34] Q. Gao, S. Lim, and X. Jia, "Hyperspectral image classification using convolutional neural networks and multiple feature learning," *Remote Sens.*, vol. 10, no. 2, 2018, doi: 10.3390/rs10020299.

[35] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "HybridSN: Exploring 3D-2D CNN Feature Hierarchy for Hyperspectral Image Classification," *arXiv*, pp. 1–5, 2019.

[36] R. Hang, Q. Liu, D. Hong, and P. Ghamisi, "Cascaded Recurrent Neural Networks for Hyperspectral Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5384–5394, 2019, doi: 10.1109/TGRS.2019.2899129.

[37] C. Shi and C.-M. Pun, "Multi-scale hierarchical recurrent neural networks for hyperspectral image classification," *Neurocomputing*, vol. 294, pp. 82–93, 2018, doi: https://doi.org/10.1016/j.neucom.2018.03.012.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 25, pp. 1097–1105, [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[39] R. Hang, Z. Li, P. Ghamisi, D. Hong, G. Xia, and Q. Liu, "Classification of Hyperspectral and LiDAR Data Using Coupled CNNs," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 7, pp. 4939–4950, Jul. 2020, doi: 10.1109/TGRS.2020.2969024.

[40] H.-C. Li, W.-Y. Wang, L. Pan, W. Li, Q. Du, and R. Tao, "Robust Capsule Network Based on Maximum Correntropy Criterion for Hyperspectral Image Classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 738–751, 2020, doi: 10.1109/JSTARS.2020.2968930.

[41] W. Sun and Q. Du, "Graph-Regularized Fast and Robust Principal Component Analysis for Hyperspectral Band Selection," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 6, pp. 3185–3195, Jun. 2018, doi: 10.1109/TGRS.2018.2794443.

[42] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *CoRR*, vol. abs/1704.04861, 2017, [Online]. Available: http://arxiv.org/abs/1704.04861.

[43] S. Bera and V. K. Shrivastava, "Analysis of various optimizers on deep convolutional neural network

model in the application of hyperspectral remote sensing image classification," *Int. J. Remote Sens.*, vol. 41, no. 7, pp. 2664–2683, 2020, doi: 10.1080/01431161.2019.1694725.

[44] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," pp. 1–18, 2012, [Online]. Available: http://arxiv.org/abs/1207.0580.

[45] A. Ben Hamida, A. Benoit, P. Lambert, and C. Ben Amar, "3-D Deep Learning Approach for Remote Sensing Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4420–4434, Aug. 2018, doi: 10.1109/TGRS.2018.2818945.