A Unified Multiscale Learning Framework for Hyperspectral Image Classification

Xue Wang, Kun Tan, *Senior Member, IEEE*, Peijun Du, *Senior Member, IEEE*, Chen Pan, and Jianwei Ding

***Abstract*— The highly correlated spectral features and the limited training samples pose challenges in hyperspectral image classification. In this article, to tackle the issues of end-to-end feature learning and transfer learning with limited labeled samples, we propose a unified multiscale learning (UML) framework, which is based on a fully convolutional network. A multiscale spatial-channel attention mechanism and a multiscale shuffle block are proposed in the UML framework to improve the problem of land-cover map distortion. The contextual information and the spectral feature are enhanced before the last classification layer based on three strategies in this work: 1) the channel shuffle operation, which was employed to learn the more effective spectral characteristics by disordering the channels of the feature map; 2) multiscale block, which considered the contextual information in multiple ranges; and 3) spatiospectral attention, which enhanced the expression of the important characteristic among all pixels. Three hyperspectral datasets, including two airborne hyperspectral images and one spaceborne hyperspectral image, were used to demonstrate the performance of the UML framework in both classification and transfer learning. The experimental results confirmed that the proposed method outperforms most of the state-of-the-art hyperspectral image classification methods. The source code is released at https://github.com/Hyper-NN/UML.**

***Index Terms*— Fully convolutional network (FCN), hyperspectral image classification, multiscale.**

I. Introduction

With the rapid development of imaging technology, imaging spectrometers can now map the surface of the Earth with hundreds of continuous and subtle spectral bands. In hyperspectral remote sensing imagery, each pixel can be regarded as a high-dimensional vector, which describes the corresponding spectral characteristics of each material [1]. The spectral characteristics contain a wealth of information that can benefit the interpretation of the material attributes. As a result, hyperspectral remote sensing has been widely utilized in the field of land-cover classification [2], quantitative monitoring [3], [4], target detection [5], military reconnaissance [6], and so on.

Classification is the most primary and basic problem of hyperspectral image analysis. Hyperspectral image classification, which refers to assigning a certain category label to each pixel according to its spectral and/or spatial features, has attracted extensive attention in the field of remote sensing. However, the issues of high data redundancy, low spatial resolution, and small training sets are still barriers to the application and development of hyperspectral imaging technology. Most of the early hyperspectral image classification methods, such as support vector machine (SVM), sparse representation, and multiple linear regression (MLR), focused on spectral feature extraction and were designed to find the boundaries in the spectral feature space [7], [8]. However, the conflict of high dimensionality of hyperspectral imagery with limited labeled samples leads to the Hughes phenomenon. Therefore, researchers have explored methods of effective feature reduction for classification, such as band selection and feature extraction. On this basis, some studies have introduced spatial features, such as the gray-level co-occurrence matrix (GLCM) or the neighborhood texture [9], [10], as the pursuit of more effective spatial features for an improved classification performance has continued. Recently, deep learning has made great breakthroughs in many computers vision fields, such as image classification, object detection, and natural language processing. At the same time, deep learning has been introduced into hyperspectral image classification and achieved good performances. The beginning of the exploration of deep learning-based hyperspectral image classification was focused on the deep representation of spectral features. Methods, such as the stacked denoising autoencoder [11] and deep belief networks [12], have been used to extract the advanced spectral characteristics. The utilization of spatial features is carried out by concatenating the 1-D spatial vector with the spectral vector. The spatial–spectral vector is then input into the network to extract the deep features [2]. Convolutional neural networks (CNNs) can automatically extract effective high-dimensional context features and can be used to solve the issues of spatiospectral feature learning in hyperspectral learning [13]. For example, Hu *et al.* [14] proposed a spectral feature learning network based on a 1-D CNN, which includes the feature learning and classification processes in an end-to-end model. The 2-D CNNs can not only represent the spectral features but also the abundant spatial information [15].

Manuscript received December 28, 2021; revised January 21, 2022; accepted January 21, 2022. Date of publication February 1, 2022; date of current version March 29, 2022. This work was supported in part by the Natural Science Foundation of China under Grant 42001350, Grant 41871337, and Grant 42171335; and in part by the Postdoctoral Science Foundation of China under Grant 2021M691016. *(Corresponding author: Kun Tan.)*

Xue Wang and Kun Tan are with the Key Laboratory of Geographic Information Science (Ministry of Education), East China Normal University, Shanghai 200241, China (e-mail: wx_ecnu@yeah.net; tankuncu@gmail.com).

Peijun Du is with the Key Laboratory for Satellite Mapping Technology and Applications of NASG, Nanjing University, Nanjing 210023, China.

Chen Pan is with the Shanghai Municipal Institute of Surveying and Mapping, Shanghai 200063, China.

Jianwei Ding is with the Second Surveying and Mapping Institute of Hebei, Shijiazhuang 050037, China.

Digital Object Identifier 10.1109/TGRS.2022.3147198

To obtain a model with a better performance, some network modeling strategies have been proposed. For example, Hao *et al.* [16] proposed a two-stream network, where one stream uses a stacked denoising autoencoder to learn the spectral features of a certain pixel and the other uses a 2-D CNN to learn the spatial features surrounding the pixel. After the two streams perform isolated learning, the high-level hierarchical features from the two streams are ensembled for the subsequent operation. Yang *et al.* [17] employed two kinds of CNNs for two-stream modeling. The two-stream pattern has now become a frequently used solution in classification [16], where it is used to balance the size of the model parameters and the joint spatiospectral features. Moreover, the classic feature extraction methods have been employed in the deep learning framework. Yao *et al.* [18] proposed ClusterCNN, which divided the hyperspectral image pixel into deferent clusters and formed a material. Roy *et al.* [19] combined the morphological operation and deep learning and proposed a morphological CNN named MorphConvHyperNet, which presented powerful nonlinear transformations for feature extraction.

Chen *et al.* [20] introduced a 3-D CNN for deep feature extraction. In a 3-D CNN, the hyperspectral cube is viewed as one-channel data with three dimensions, and the kernel consists of 3-D matrices that perform the convolution operation along the spatial and spectral dimensions at the same time. However, the negative side of 3-D CNNs is that the expanded convolution dimension results in an excess of parameters. To address this issue, Howard *et al.* [21] proposed depthwise separable convolution, which can be regarded as a valid solution for the overparameterized issue. Jia *et al.* [22] proposed the lightweight CNN to tackle the large gap between the massive parameters and limited labeled samples. Zhang *et al.* [23] proposed a 3-D lightweight CNN for the HSI classification and the transfer application with limited labeled samples.

The above-mentioned CNN-based methods are carried out using patches, i.e., when labeling one pixel, the data patch surrounding this pixel is input into the network. In this case, if all the pixels in the entire image need to be predicted, the data patch should be obtained by sliding a window in a one-step stride, which involves repetitive calculation. These redundant operations not only waste computational memory but also increase the time consumption. Some researchers have introduced a fully convolutional network (FCN) to handle this flaw. FCNs include both a convolution operation and deconvolution operation and have been widely applied in image segmentation. The main characteristics of FCNs is that they do not incorporate any fully connected layers, which endows the model with a multidimensional input, high robustness, and a fine feature representation ability. Li *et al.* [24] proposed a deep learning framework for hyperspectral image classification based on an FCN, where the deep features of the hyperspectral imagery are enhanced through convolutional and deconvolutional layers. The final classification part of this network is an optimized extreme learning machine, which allows the FCN to obtain a good performance. Unlike the separate feature enhancement and classification design [12], Zheng *et al.* [25] proposed a fully end-to-end classification framework named the fast patch-free global learning (FPGA) framework. In the FPGA framework, the deconvolution part is substituted by interpolation, which reduces the number of model parameters. Because of the patch-free feedforward process, the FPGA framework has a definite advantage in inference speed.

FCNs need a large labeled dataset, and each training iteration needs at least one whole image for the loss calculation, which is the main obstacle in hyperspectral image classification. Zheng *et al.* [25] proposed a global sampling strategy, in which all the training samples are transformed into a stochastic sequence of stratified samples. The stochastic combination enlarges the training dataset volume effectively, which helps with the fine optimization of the model.

The other defect of CNN-based hyperspectral image classification is that the obtained land-cover map can be distorted, and the bigger the convolution kernel, the worse the distortion. In patch-based CNN classification, the label of the center pixel in the hyperspectral remote sensing image is determined by all the pixels in this patch. In this case, the center pixel will be endowed a wrong label when the pixels in the patch belong to different categories, which are of common occurrence at the boundaries of ground objects. To address this issue, some researchers have introduced postclassification optimization. For example, Zhong *et al.* [26] used the conditional random field method to modify the classification map based on the posterior probability and the spatial information, which improved the misclassification problem. Moreover, the multiscale features can be taken into consideration to improve the classification accuracy effectively [27]. Xu *et al.* [28] proposed spectral–spatial unified networks (SSUNs), which included a multiscale CNN as the spectral and spatial feature extractors. The SSUN combines both local information and global information for the classification and can yield a competitive performance. Gao *et al.* [29] designed a multiscale residual network by replacing the convolutional layer in an ordinary residual block with the mixed depthwise convolution, and the experimental results showed that the spatiospectral feature representation ability had been enhanced. Roy *et al.* [30] proposed a 3-D CNN and amplified the multiscale information using adaptive spectral–spatial kernel improved residual network for HSI classification, which proved to capture the discriminative features.

In conclusion, there are two defects in CNN-based methods: 1) the patch-based methods include redundant operations, which involve repetitive calculation and increase the running time, and 2) the obtained land-cover map can be distorted. The bigger the convolution kernel is, the worse the distortion is, which means that the boundary of the ground object is inaccuracy in the classification map. Detailed information related to object boundaries is missing due to the downsampling or convolutions with striding or pooling operations, which will lead to distortion in the classification map. To address these problems, the contextual information and the spectral feature are enhanced before the last classification layer based on three strategies in this work: 1) the channel shuffle operation, which was employed to learn the more effective spectral characteristics by disordering the channels of the feature map; 2) multiscale block, which considered the contextual

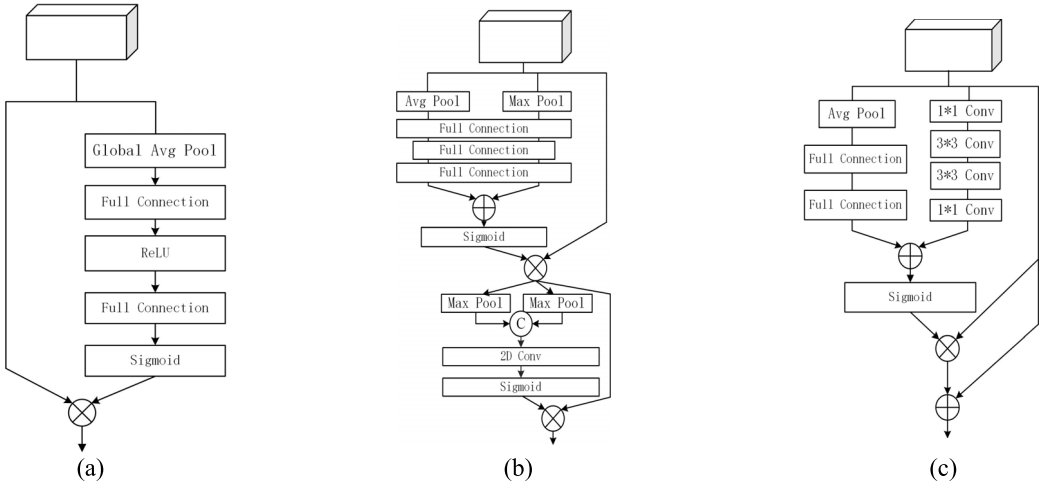

Fig. 1. Different attention modules. (a) SEM. (b) CBAM. (c) BAM.

information in multiple ranges; and 3) spatiospectral attention, which enhanced the expression of the important characteristic among all pixels. Moreover, the decoder in unified multiscale learning (UML) can also recover the detailed knowledge in the upsampling process.

The main contribution of the UML framework is to tackle the issues of redundant operation and land-cover map distortion in the hyperspectral image classification. The UML, which is based on the FCN model, does not incorporate any fully connected layers and endows the model with a multidimensional input and a faster inference. Moreover, a series of modules, such as multiscale learning, shuffle block, and spatiospectral attention, have been introduced in UML to improve the category discrimination accuracy effectively. The network workload and the training samples are balanced by introducing separated convolution, which reduces the model complexity. To enhance the features in each layer, the channels are shuffled by a cascade hierarchy. Moreover, each intermediate layer is concatenated with its corresponding upsampling feature, which ensures fine-grained feature representation.

The rest of this article is organized as follows. Section II describes the basic hyperspectral image classification methods. Section III details the proposed UML framework. Section IV introduces the three real hyperspectral images used in the experiments and the experimental results and also provides a comparison with other methods. Finally, our conclusions are drawn in Section V.

II. Previous Work

A. Convolutional Neural Networks and Fully Convolutional Networks

CNNs extract the correlation among spatial blocks by enhancing the neural network connections between adjacent neurons and realize spatial authenticity through block neurons. A CNN can be regarded as a surface-oriented model, which is different from the autoencoder. The pioneering work on CNNs is generally considered to be the LeNet architecture, as proposed in [16], which consists of the basic network composition of a CNN, including a local connection layer, a pooling layer, and a fully connected layer.

When predicting the label of a certain pixel, the input patch of the CNN is the data patch surrounding this pixel, which has some drawbacks. First, the patch size has a direct impact on the network performance, in which a larger patch size brings more rich neighborhood characteristics. Unfortunately, no matter how large the patch size is set, the patch-based calculation still causes a limited receptive field. Meanwhile, a large patch size results in worse distortion, which means that the patch size should be considered as a hyperparameter. Moreover, the patch-based feedforward operation by sliding a window in a one-step stride means repetitive calculation, which has a great impact on storage and computing consumption.

The networks used for classification usually connect several layers of fully connected layers following the convolutional feature learning part, which flattens the original multidimensional feature tensor (image) into a 1-D vector. The 1-D vector can be regarded as a numerical description (probability) of the entire input image. Thus, spatial information is lost in the flattening process. The FCN was proposed for image semantic segmentation [31], in which the fully connected layer in the conventional CNN is replaced with a convolutional layer, so that the network output is no longer a category but instead a heatmap. Unlike CNNs, FCNs are appropriate for any size of input and can generate a prediction for each pixel concurrently. The advantages of FCNs lie in three aspects: 1) the lack of fully connected layers allows the FCN to accept input images with different sizes; 2) the deconvolution operation in the upsampling process can enhance the feature size, which ensures a fine-grained output; and 3) each encoder layer has a lateral connection (which is named the skip connection) with the corresponding decoder layer, and the skip connection fuses the feature map and heatmap to better restore the features of the image, which guarantee the robustness and accuracy.

B. Attention Mechanism

An attention mechanism is proposed to solve the bottleneck problem of information loss caused by the transformation

from a long sequence to a fixed-length vector. An attention mechanism, imitating the human attention mechanism, can quickly screen the highest value information from a large amount of information. Attention mechanisms are widely used in the field of Seq2Seq models and are mainly used to solve the problem of it being difficult to obtain a final reasonable vector representation with a long input sequence in long short-term memory (LSTM)/recurrent neural network (RNN) models. The attention mechanism is a technique that enables models to focus on important information and fully learn it. The importance is valued by the final loss function and is used to weight the midlevel features, by which the learned features are specialized on the partial features. Attention mechanisms have already been introduced in the image classification and segmentation tasks, such as the squeeze and excitation module (SEM) [32], the bottleneck attention module (BAM) [33], and the convolutional block attention module (CBAM) [34], which are illustrated in Fig. 1.

The SEM emphasizes the channel information and is effective in solving the loss caused by the different weights of different channels in the feature map. In the traditional convolution processes, the importance of each channel in the feature map is equal. However, different channels are of different importance in practical problems and should be considered on a case-by-case basis. In the SEM, the feature map is squeezed into a 1-D map by global average pooling and bottom-up full connection follows, which has the advantage of adding more nonlinear information in the learning processing and fitting the complex correlations between channels. The CBAM and the BAM pay more attention to both the channel and spatial dimensions, but the difference is that the CBAM employs the 2-D convolution with a fixed kernel size, whereas the kernel size differs in the BAM.

III. Proposed Method

A. Problem of Conventional CNNs

In patch-based CNN classification, the label of the center pixel in the hyperspectral remote sensing image is determined by all the pixels in this patch. When the convolution window moves to the boundary of different categories, the center pixel will be endowed a label in accordance with the domain category in the pixel with the highest probability. In this case, the interpretation results obtained by the CNN cannot reflect the real distribution of the objects. Meanwhile, there is a valid case for the center pixel being labeled by its spectral features and the nearest pixels. The greater the distance of the pixel from the center, the less effect it should have on the label determination. This misclassification is not significant on training and test datasets because the prior knowledge constricts the discrimination of the network. Meanwhile, at the boundaries of ground objects that are unlabeled, misjudgments are more common. Moreover, in the production process of standard datasets, the purity of the labeled samples is generally considered, and researchers often do not keep the pixels at the junction of different categories, preferring to choose more pure internal pixels for labeling. Therefore, when the convolution operation is carried out for hyperspectral image classification, although the performance on the test dataset will be good, ground object boundaries will appear on a large number of the unlabeled pixels, and this phenomenon will become more significant with larger patch sizes.

To further explain the distortion arising from a CNN, the features that determine the characteristics of a certain category can be extracted and the relationship between the network weights and the contribution of the weights can be explored. In this case, the gradient information flowing into the last convolutional layer of the CNN can be extracted to establish the importance of the convolution kernel for a decision of interest. After the training process, the output of the convolution operation is

$$a^{m,n,s} = \mathcal{F}\left(z^{l}\right) = \mathcal{F}\left(\sum_{c=1}^{C_{l-1}} a^{l-1,t} * W^{l,s,c} + b^{l,s}\right) \tag{1}$$

$$\mathcal{F}(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{if } z < 0 \end{cases} \tag{2}$$

where $a^{m,n,s}$ represents the features ($m \times n$ matrix) from the sth channel in layer l, $a^{l-1,t}$ denotes the features from the tth channel, C_{l-1} is the count of all the input channels, $W^{l,s,c}$ is the weight matrix, and $b^{l,s}$ is the bias. $*$ and $\mathcal{F}()$ denote the convolution operation and the activation function, respectively. After the convolution phase, a fully connected layer follows to output the logit into the softmax layer, which is formulized as

$$\text{lgt} = z^{l_{\text{fc}}} \tag{3}$$

$$S_i = e^{\text{lgt}_i} / \sum_j e^{\text{lgt}_j} \tag{4}$$

where l_{fc} denotes the fully connected layer; lgt is the output vector before the softmax layer, which describes the confidence for each class, $\text{lgt} = (n_1, n_2, \ldots, n_{\text{cls}})$; and S_i is the output from the softmax layer.

Convolutional features naturally retain the spatial information, which is lost in the fully connected layers [35], so the final convolutional layer is used to detail the relationship between the high-level semantic features and the class information. The gradient information flowing into the last convolutional layer of the CNN is calculated to establish the importance of each neuron for a decision. The gradient on the features at the last convolutional layer is the sum of each kernel

$$\alpha_s^j = \frac{1}{Z} \sum_m \sum_n \frac{\partial \text{lgt}_j}{a^{m,n,s}} \tag{5}$$

where Z is the normalized parameter, α_s^j represents the importance of the feature map on the sth channel for target class j, and $a^{m,n,s}$ is the output of the activation function on the sth channel of the last convolutional layer. The partial derivative is then shrunk through its spatial dimensions $m \times n$. The weighted combination of the forward feature maps is carried out by [34]

$$\text{heat\_map} = \text{Relu}\left(\sum_s \alpha_s^j a^{m,n,s}\right). \tag{6}$$

TABLE I
HEATMAPS FOR DIFFERENT PATCH SIZES

Patch size	3	4	5	6	7	8	9	10
Label	4	4	6	6	6	6	6	6
Original								
Heatmap								
Label	9	1	9	4	4	4	4	4
Original								
Heatmap								

To explore the heatmaps of CNNs based on different patch sizes, the patch size was divided into seven grades from small to large. The CNN was then trained based on the patch-based model using the seven different patch sizes. The output labels and the corresponding heatmaps are shown in Table I.

Two different scenes were chosen. One of the scenes is at the junction of bare soil and tree, and the true label of the center pixel is tree. The other scene is at the junction of asphalt and tree, and the true label of the center pixel is asphalt. From Table I, the heatmaps with a small patch size illustrate that the information for the category discrimination is localized in the center region, while the heatmap highlights the bare soil region when the patch size is larger than 6, which causes a wrong label. In the asphalt-tree scene, the highlighted area spreads to the edge of the patch as the patch size increases. In this regard, multiscale features or differentiated attention should be considered.

B. Channel Shuffle Operation and Multiscale Block

The bigger the kernel is, the more features are learned, but the error at the boundaries of different ground objects follows. To handle this problem, there should be a tradeoff between the convolutional features and the kernel size. On the one hand, the spatial–spectral features should be learned effectively. On the other hand, the receptive field should be even larger. In the proposed network, to enhance the receptive field, a multiscale block is employed. The multiscale theory is widely used in remote sensing interpretation [36], and it refers to pixels in multiple ranges being included for the calculation.

In this work, convolution operations with different kernel sizes are used for the multiscale feature learning on the same patch. Dilated convolution is applied for the different kernel sizes to solve the problem of excessive training parameters with large convolution kernels [37]. The multiple kernels are performed in both cascade and parallel patterns. The features are exported as

$$V_{\text{cascade}} = \text{conv2}(W_3, (\text{conv2}(W_2, \text{conv2}(W_1, X)+b_1))+b_2)) + b_3 \quad (7)$$

$$V_{\text{parallel}} = (\text{conv2}(W_1, X) + b_1) + (\text{conv2}(W_2, X) + b_2) + (\text{conv2}(W_3, X) + b_3) \quad (8)$$

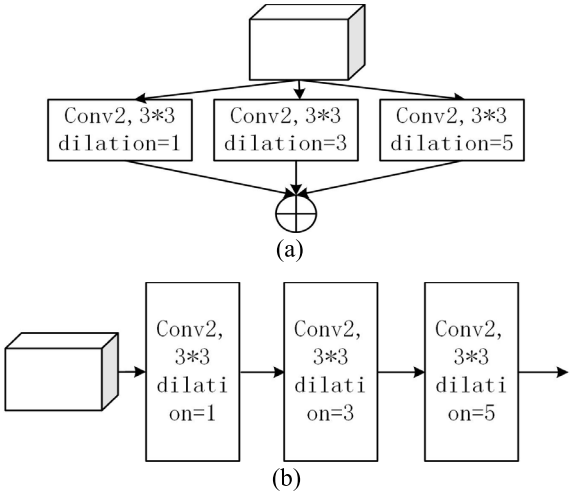

Fig. 2. Illustration of multiscale feature learning. (a) Parallel pattern. (b) Cascade pattern.

where the weight matrix has different kernel sizes and dilation rates to represent the different scale features. The difference between cascade and parallel patterns is shown in Fig. 2.

The output of the convolution operation should be of the same size after the different dilation rates. In this regard, padding is included around the input feature block to maintain the same size. The output size is calculated as follows:

$$S_{\text{out}} = \frac{S_{\text{in}} - k + (k-1) \times (\text{dilation} - 1) + 2\text{pd}}{\text{std}} + 1 \quad (9)$$

where S_{in} is the width of the input features, k and std denote the kernel size and stride, respectively, and pd is the number of padding pixels. In this work, S_{out} is equal to S_{in}, so pd should be set to

$$\text{pd} = \frac{S_{\text{in}} \times (\text{std} - 1) + (k-1)(2 - \text{dilation})}{2} \quad (10)$$

i.e., the number of zero pixels that should be added around the input matrix is calculated to ensure that the postconvolution matrix is as big as the input matrix.

In addition to multiscale feature learning, a channel shuffle operation [38] is included in the proposed network for feature

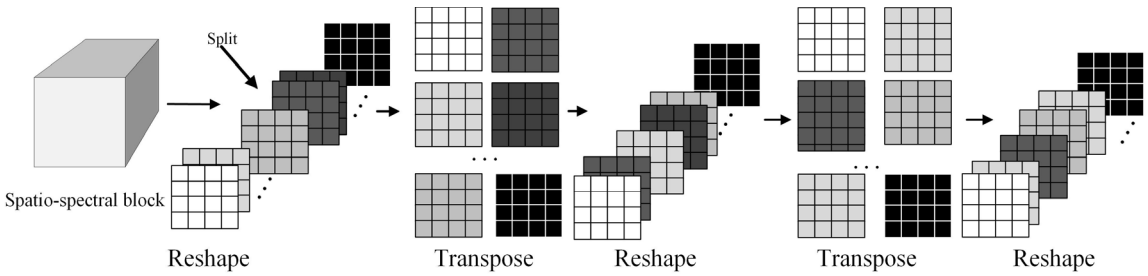

Fig. 3. Illustration of the shuffle phase.

enhancement, where it disorders the original channels of the feature map and helps to establish the connections between different channels. The shuffle process is divided into "reshape–transpose–reshape" operations. After the full iterative shuffle process, the channel is fully disordered. The information flow between features is considered for different groupings to improve the network performance. Moreover, the dimensional rearrangement operation in the shuffle process is supported by many built-in library functions on universal computing platforms, which benefits the computational acceleration.

In the proposed network, the multiscale learning is executed by convolutional operations with different dilation rates, which are embedded in the shuffle block. The channel shuffle operation iterates twice, and the channel is divided into two branches in each iteration, as shown in Fig. 3. The multiscale shuffle block is divided into two branches, and the multiscale branch is used for the multiscale feature extraction. These multiple kernels are performed in a parallel pattern. After the channel learning by 1×1 convolution operation, these three convolution layers with different dilation rates are carried out for the multiscale features extraction. These features are obtained by the elementwise addition of parallel output and the dilation rates are set to 1, 3, and 5.

C. Spatiospectral Attention

A CNN consists of a series of convolutional layers, nonlinear layers, and subsampling layers so that enhanced image features can be captured from the global receptive field for the image description. As an essential part of a CNN, the convolution operation is used to learn the spatial information, but not the information between channels. However, the channel information is important information in hyperspectral remote sensing classification, and the internal relationship among the channel features should be accessed directly and explicitly. The importance degree of each feature channel is automatically obtained by representative feature learning. The information that is conducive for the classification is then promoted according to the learning degree, and the trivial features that are not useful for the current task are suppressed.

An attention mechanism not only indicates where we should focus but also enhances the expression of this. Therefore, both spatial attention and spectral attention should be considered simultaneously. The three kinds of attention mechanisms mentioned in Section II differ in their focus. The SEM aims to tackle the issue of exploiting channel dependencies so that the squeeze process for the global spatial information and excitation of the specific input weight are included. The CBAM, which differs from the SEM that considers the channel attention only, engages the spatial and channel information in a cascade pattern. In the proposed network, deep spatiospectral coupling is introduced for attention information learning. Inspired by the depthwise convolution approach [39], which proposes tradeoffs between the size of the model parameters and the model performance, flexible attention learning is performed. The spatiospectral coupling uses a two-branch learning pattern. First, the input features are replicated into two branches, each of which aims to learn the spatiochannel attention. After the branch learning, the channelwise importance is wrapped with the spatial attention. The channel branch resembles the SEM and is performed by squeeze and excitation.

First, the global information in a certain channel is embedded by global average pooling to generate channelwise statistics

$$\mathrm{Se}_s = \frac{1}{Z} \sum_m \sum_n a^{m,n,s} \tag{11}$$

where Z is the normalized parameter and $a^{m,n,s}$ is the output of the activation function on the sth channel. The global pooling is used to shrink the 3-D feature matrix through its spatial dimensions. Se_s can be regarded as a descriptor for the whole image characteristics, which is widely used in computer vision tasks. The local descriptor then follows with a sequential operation to learn the channel importance. A simple gating mechanism with a sigmoid activation function is used to capture the channelwise dependencies

$$\mathrm{Ext}_s = \mathrm{Sigmoid}(W_2 \mathrm{Relu}(W_2 \mathrm{Se})) \tag{12}$$

where W_2 and W_1 are performed by two fully connected layers, which ensures that the attention weights are learnable, and Se is the global information after the generation of the channelwise statistics.

The spatial attention mechanism is applied on the other branch. In this branch, the spatial context descriptor is applied first. The spatial context consists of the "average context" and the "max context." The former selects the spatial features with better classification identification, and the latter is more reflected in the dimension of the complete transmission of the

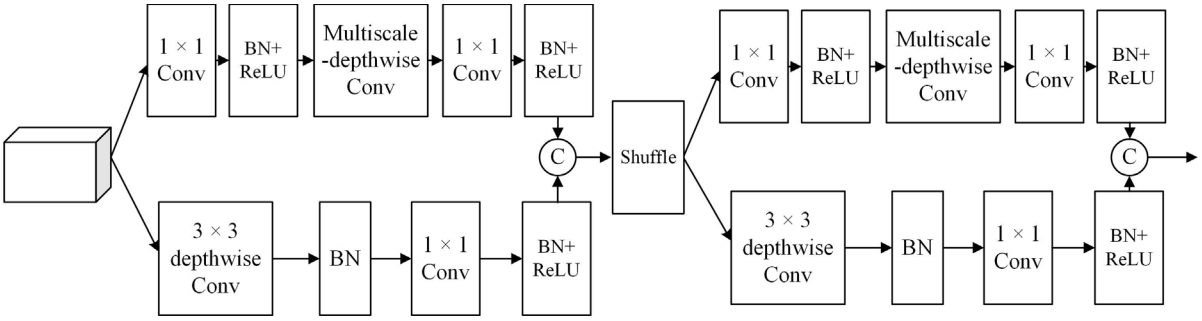

Fig. 4. Illustration of the multiscale shuffle block.

spatial information

$$\mathrm{Des} = \left[\frac{1}{Z}\sum_{s} a^{m,n,s}, \mathrm{Max}_s(a^{m,n})\right] \tag{13}$$

$$\mathrm{Sa} = \mathrm{Sigmoid}(\mathrm{Conv}\left(\left[\frac{1}{Z}\sum_{s} a^{m,n,s}, \mathrm{Max}_s\left(a^{m,n}\right)\right]\right) \tag{14}$$

where Des is the descriptor of the joint spatial context. The former part is the average through the channel dimensions, and the second part is the maximum value through the channel dimensions. After the matrix coupling, an additional convolution operation follows, and a nonlinear continuous output is achieved by the sigmoid activation function. The final attention descriptor is the cross product of Sa and Ext, which extends the spatial attention weight to three dimensions with different channel attention weights. After the spatial and channel attention coupling phase, the original input multiplies Ext and Sa to obtain the attention-applied features.

D. Unified Multiscale Learning

The UML network consists of an initial block, shuffle blocks 1 and 2, upsampling blocks 1 and 2, and a classification block. The initial block is used for the preliminary feature extraction, and each of the two shuffle blocks carries out two iterations of convolutional learning to take the information flow between features into account, which improves the network performance. In the shuffle blocks, multiscale learning with different dilation rates is applied, and the stride is set to 2 to downsample the data size.

The 2-D convolution with multiple channels is performed in some works [40] as depthwise convolution and pointwise convolution, which is a tradeoff between the size of the model parameters and the model performance. The depthwise convolution and pointwise convolution are integrated as depthwise separable convolution. The depthwise spatial convolution is a convolution performed independently over each channel of an input, which is carried out using group convolution, and the pointwise convolution is a 1×1 convolution, which projects the channels' output by the depthwise convolution onto a new channel space. However, an excess of group convolution will result in a massive memory access cost. The number of convolution operations is related to the channels in the group convolution, which means high-frequency memory access. In some cases, the execution speed of a certain algorithm is mainly affected by the input or output of the memory in terms of the parallel computing framework [38] because the simple instruction set prefers whole shared memory with intensive computing, rather than scattered memory with small floating-point operations (FLOPs).

In this regard, an advanced shuffle mechanism is used, where the equal channel width minimizes the memory access cost. Each shuffle block in the UML network is performed, as shown in Fig. 4.

The multiscale shuffle block consists of branchwise multiscale convolution and channel shuffle operations. The channel shuffle operation is carried out, as shown in Fig. 3. The branchwise convolution divides the two branches, and the multiscale branch is used for the multiscale feature extraction. The 1×1 convolution and depthwise convolution are used to downscale the training parameters. The output features are concatenated along the channel, and the channel shuffle operation follows to disrupt the channel order. After this, the other shuffle block is carried out for the next shuffle step. The UML framework is performed, as shown in Fig. 5.

Fig. 5 shows the UML framework, which is composed of an encoding phase and a decoding phase. The initial block, shuffle block 1, and shuffle block 2 are used for the hyperspectral image feature encoding. The spatial-channel attention mechanism is applied after each block. Deconvolution and interpolation are commonly used for upsampling. The output channel number for the initial block is 80. In shuffle block 1, the input is divided into two branches and the stride is set to 2. Forty channels are used for the shuffle group. After two shuffle iterations, the output channels of shuffle block 1 number 160. The following shuffle block 2 has the same branches and stride as shuffle block 1. The shuffle group number is 80, which is in accordance with half the output size of the last block. The output channels after the two shuffle blocks' number 320. In the proposed network, interpolation with convolution is employed in the decoder phase, which can avoid the checkerboard artifacts caused by deconvolution [41]. Moreover, when the result after the encoding phase is directly deconvolved, the result is often relatively rough because the sequential convolution causes a loss of fine features, and at the end of the downsampling, the size is a quarter of the original size. In this regard, a skip connection is used to couple the features from the encoder and decoder that are under the same scale. The features from the encoder are concatenated

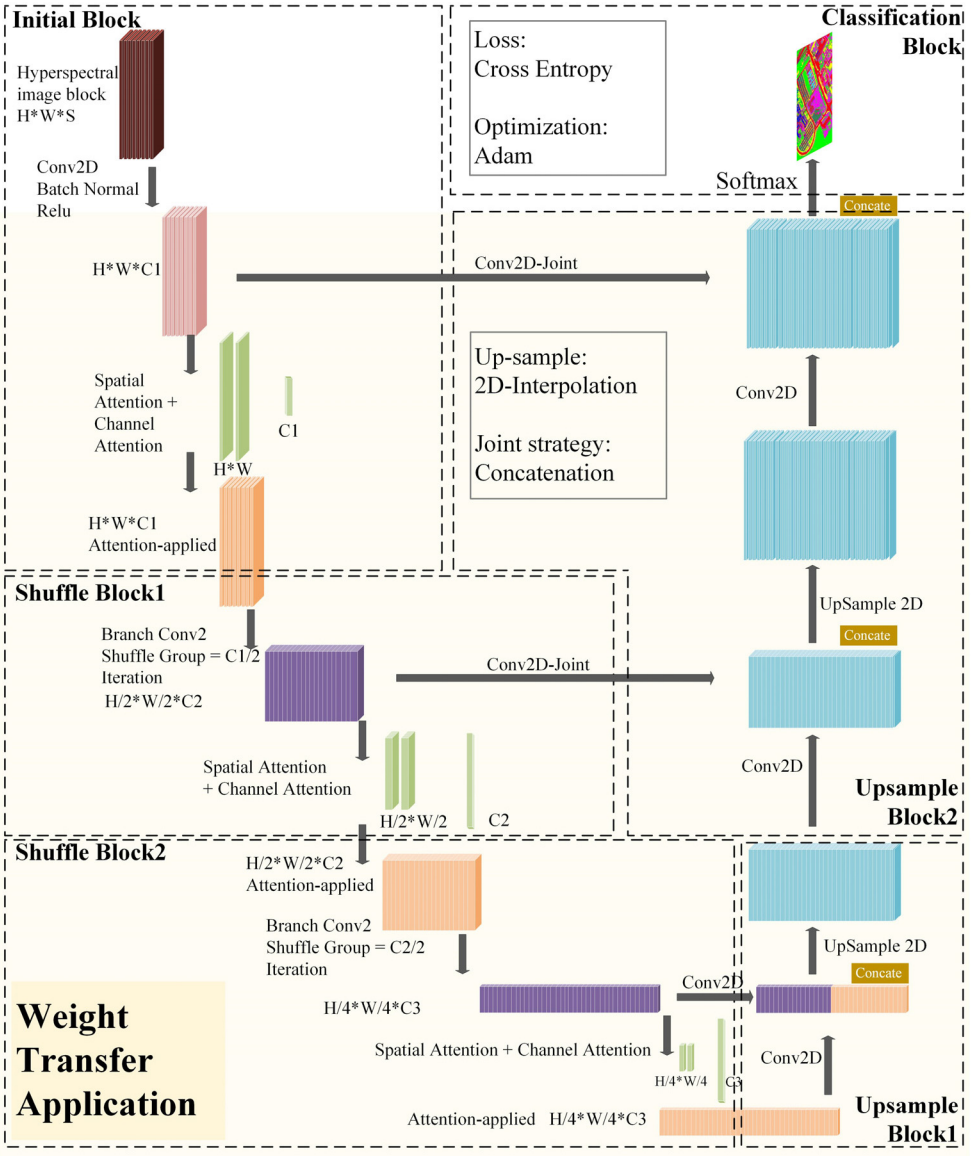

Fig. 5. Structure of the UML framework for hyperspectral image classification.

with the features from the decoder. The classification block is responsible for the labeling task, and the output of the classification block is the classification map that labels every pixel. The loss of the UML framework is the summation of each pixel's cross entropy.

E. Cross-Scene Classification Based on UML

The transfer learning for cross-scene classification should have the same patch size in the patch-based method, which restricts the application condition. The optimal patch size depends on the hyperspectral image [24]. UML includes both a convolution operation and an upsampling operation, which allows the input hyperspectral image to have different samples and lines. This characteristic endows the UML with an effective cross-scene classification ability.

The encoder–decoder framework of UML can learn the advanced spatiospectral features of the hyperspectral imagery, which is generally effective for the classification of other scenes. As shown in Fig. 5, when performing transfer learning, the initial block and the classification block are replaced by the particular block corresponding to the target domain, while the other blocks are preserved from the source domain. The first convolution layer in the initial block is used to transform the spectral features S from the target domain to the intermediate dimension $C1$. The convolution layer before the softmax in the classification block is used to transform the category space to the target domain. After fine-tuning these two layers by few priori knowledge in the target scene, the UML is capable of handling the cross-scene classification.

In this way, the UML framework can achieve a high classification accuracy in a case of limited labeled samples in the target domain, which solves the problem of poor classification performance in the case of limited labeled samples.

Fig. 6. Pseudo-color composite image and the corresponding ground truth for the Pavia University ROSIS dataset.

IV. Experiments

In the following, to report the accuracy and efficiency of the UML framework, the classification accuracy and kappa coefficient are used. The training dataset was obtained with a fixed number in each category. Conventional classifiers, such as SVM and the mainstream deep learning algorithms based on CNNs, were chosen as the baseline methods. The CNN was carried out as a patch-based method, and the patch size was set to 4 and 8. The channel shuffle operation and multiscale feature learning were appended to the conventional patch-based CNN to obtain MSNet. FPGA [25] was used as a counterpart of an FCN. SVM used a Gaussian radial basis function (RBF) kernel, and the gamma and C of SVM were optimized by grid search. The SVM classifier then used the optimal values of C and gamma. Because the compute unified device architecture (CUDA) is commonly used as a deep learning arithmetic accelerator, all the inference parts were carried out on an NVIDIA GeForce RTX 2080 graphics processing unit (GPU) with CUDA. In each batch, the patch-based networks consist of sliding window sampling with a fixed step, memory mapping to the GPU, and inference with CUDA.

In summary, the comparative methods consisted of SVM with RBF kernel, CNN-4 (with a patch size of 4), CNN-8 (with a patch size of 8), MSNet-4 (with a patch size of 4), MSNet-8 (with a patch size of 8), and FPGA. In order to assess the performance of the proposed approach, three indicators are adopted by comparing the classification results with the ground truth: 1) precision, denoting the ratio of the number of correctly classified samples to the total number of samples in a certain category; 2) overall accuracy (OA), referring to the ratio of the number of correctly classified pixels to the total number of test pixels; and 3) kappa coefficient, representing whether the model prediction results are consistent with the actual classification results. In order to demonstrate the performance of the proposed technique, three real hyperspectral images were utilized—the Indian Pines dataset, the Pavia University dataset, and the Kansas dataset—which were, respectively, captured by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), the Reflective Optics System Imaging Spectrometer (ROSIS), and the visible-shortwave infrared (SWIR) Advanced Hyperspectral Imager (AHSI) of Gaofen 5 (GF-5).

A. Dataset Description

1) Pavia University ROSIS Dataset: The Pavia University dataset was acquired by the ROSIS sensor over the Engineering School of the University of Pavia, Italy. This dataset consists of 610 × 340 pixels, with a spatial resolution of 1.3 m/pixel. The number of spectral bands is 115, ranging from 430 to 860 nm, and 12 noisy bands were removed because of the absorption of water vapor. The remaining 103 spectral bands were used in the experiments. The dataset contains nine categories of interest. Because of the high spatial resolution, the scattered objects, such as the trees and the footpath, bring a great difficulty to CNN-based feature learning. The pseudo-color composite image and the labeled categories are shown in Fig. 6.

2) Kansas AHSI Dataset: The Kansas AHSI dataset was collected by the visible-SWIR AHSI designed by the Shanghai Institute of Technical Physics, Chinese Academy of Sciences, which was one of the main payloads onboard the GF-5 satellite. The Kansas AHSI dataset covers 330 spectral bands to characterize the solar reflective regime from 400 to 2500 nm. The spectral resolution is about 5 nm for the visible and near-infrared (VNIR) region from 400 to 1000 nm, and 10 nm for the SWIR region from 1000 to 2500 nm. The spatial resolution is around 30 m. The dataset was acquired over Kansas in USA in November 2018. The dataset size is 650 × 340 pixels. After geometric correction, radiometric correction, and other preprocessing, the interior objects in the study area were divided into seven categories by referring to Google Earth images of the same period and place. This dataset contains many crossroads and regular town districts, which can be used to demonstrate the visual performance of the different classification models. The specific ground features of the Kansas AHSI dataset are shown in Fig. 7.

3) Indian Pines AVIRIS Dataset: The Indian Pines dataset was collected by the AVIRIS sensor over the northwestern Indiana agricultural test site. The dataset size is 145 × 145 pixels, with a spatial resolution of 17 m/pixel. A total of 200 bands ranging from 400 to 2500 nm were used in

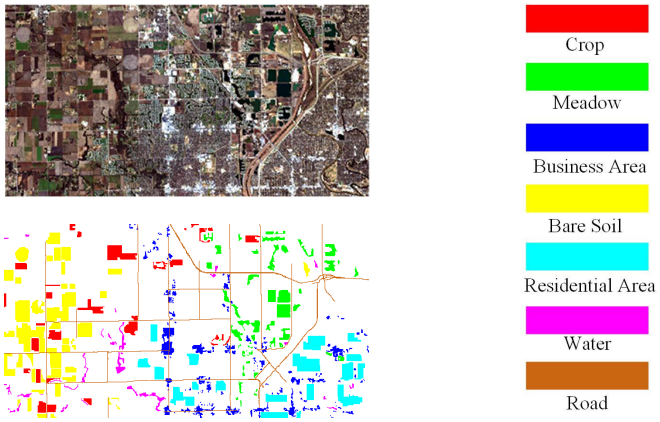

Fig. 7. Pseudo-color composite image and the corresponding ground truth for the Kansas AHSI dataset.

TABLE II
NUMBER OF LABELED PIXELS IN EACH CLASS IN THE ROSIS DATASET

Class name	Labeled sample number	Training set number
Asphalt	6631	100
Meadows	18649	100
Gravel	2099	100
Trees	3064	100
Painted Metal Sheets	1345	100
Bare Soil	5029	100
Bitumen	1330	100
Self-Blocking Bricks	3682	100
Shadows	947	100

the experiments, and 24 bands near 1400 and 1900 nm that were affected by the absorption of water vapor were removed. The available training samples cover 16 categories of interest, which are mostly different types of vegetation. The boundaries of the vegetation types are not clear, which brings great difficulty to the sliding convolution operation. Moreover, the vegetation, farm tracks, and boundaries in this dataset cannot be mapped precisely, on account of the insufficient training dataset. The pseudo-color composite image and the labeled categories are shown in Fig. 8.

B. Experiments With the Pavia University ROSIS Dataset

One hundred labeled samples were sampled in each category for the Pavia University ROSIS dataset training data, as shown in Table II.

For the optimized phase of SVM, the validation dataset was 5% of the whole training data. The batch size of the deep learning methods was set to 30. FPGA and UML were trained using the random selection strategy proposed in [13]. After the model training, all the pixels were predicted and the evaluation indicators were calculated. The experiments were in fact repeated ten times over the randomly split training and test data. In this dataset, a large number of trees are found, including both rows of trees and scattered single trees. Meanwhile, the ground objects are varied and include both circular buildings and square buildings. With the high spatial resolution, this dataset is suitable for verifying the mapping authenticity and ground object deformation of the different methods.

The precision of the different approaches for the Pavia University ROSIS dataset is shown in Table III, where the best results are marked in bold. UML and FPGA obtain the best and second best results of 99.75% and 99.51%, respectively, on the Pavia University ROSIS dataset. The CNN-8 method and all the MSNet methods obtain a higher OA than SVM with the optimal parameter settings. It can also be seen that the bigger the patch size, the higher the accuracy of the CNN-based methods. The precision of all the classes for CNN-8 is higher than that for CNN-4, except for the "Shadows" class. UML obtains a precision of more than 99% for all the classes. However, as mentioned above, the performance on the test dataset cannot be used to fully assess the performance of a classification method. Although the performance on the test dataset will be good, ground object boundaries will appear on a large number of unlabeled pixels. The mapping results of each method are shown in Fig. 9.

Fig. 9 shows that the salt-and-pepper noise is very serious in the result of SVM. The surfaces obtained by CNN-8 and FPGA are smooth, but the scattered objects are missed and the boundaries between two different land-cover types are misclassified. For example, there is a flower bed in the top-right corner, which is highlighted by the red box in Fig. 9. However, the flower bed is missed in the results of FPGA and CNN-8, which is caused by the spatial information being lost during the CNN downsampling. The introduction of multiscale learning allows MSNet-8 to retain the flower bed, but its accuracy is low. The OA of A^2S^2RN is lower than that of the FCN-based methods and higher than that of other patch-based methods. The result of MSRN shows oversmooth phenomenon, which may be caused by its biggest patch size. Moreover, many trees without labels are missing in the results of CNN-8 and FPGA. FPGA has oversmooth phenomenon, while the proposed UML can solve this problem better. In the UML, spatial–spectral attention has enhanced the importance of the center pixel. Furthermore, the shuffle and multiscale learning improve the diversity and separability of the extracted features, which can handle the distortion. The results of UML show that the scattered objects such as a single tree are fully preserved and the salt-and-pepper noise is well restrained, and the results are closer to the real land-cover distribution map. From Fig. 9 and Table III, it can be seen that UML obtains the best performance on the Pavia University ROSIS dataset.

C. Experiments With the Kansas AHSI Dataset

One hundred labeled samples were sampled in each category for the Kansas AHSI dataset training data, as shown in Table IV. For the optimized phase of SVM, the validation dataset was 5% of the whole training data. The batch size of the deep learning methods was set to 30. After the model training, all the pixels were predicted and the evaluation indicators were calculated. The experiments were again repeated ten times over the randomly split training and test data.

Fig. 8. Pseudo-color composite image and the corresponding ground truth for the Indian Pines AVIRIS dataset.

TABLE III
PRECISION OF THE DIFFERENT APPROACHES FOR THE PAVIA UNIVERSITY ROSIS DATASET

Category	SVM	CNN-4	CNN-8	MSNet -4	MSNet -8	MSRN	A^2S^2RN	FPGA	UML
Asphalt	79.96	82.67	91.15	94.20	86.29	96.17	99.60	99.10	99.16
Meadows	88.06	85.61	94.50	97.88	94.59	99.38	96.53	99.49	99.85
Gravel	85.19	99.14	96.55	96.97	87.73	97.27	95.92	99.79	99.53
Trees	93.29	96.92	99.29	97.09	97.58	90.84	98.34	98.95	99.69
Painted metal sheets	99.44	98.96	100.00	99.92	95.67	97.83	97.75	100.00	100.00
Bare soil	87.71	83.53	100.00	96.61	90.02	99.45	99.86	100.00	100.00
Bitumen	93.82	75.45	100.00	91.30	91.07	96.99	83.82	100.00	100.00
Self-blocking bricks	83.72	19.35	98.38	81.91	87.23	94.89	92.26	99.77	99.97
Shadows	99.76	99.06	97.99	98.11	99.27	95.87	98.81	99.33	100.00
OA	87.36	80.93	95.79	95.51	91.96	97.52	97.3	99.51	**99.75**
KAPPA	0.8348	0.7540	0.9444	0.9408	0.8950	0.9678	0.9648	0.9935	**0.9967**

TABLE IV
NUMBER OF LABELED PIXELS IN EACH CLASS IN THE KANSAS AHSI DATASET

Class name	Labeled sample number	Training set number
Crop	5203	100
Meadow	5468	100
Business Area	4695	100
Bare Soil	12088	100
Residential Area	8235	100
Water	1683	100
Road	4581	100

Because this dataset was collected by satellite, the spatial resolution is low and the phenomenon of mixed pixels is serious. Moreover, the shapes of the ground objects are complex, including regular crossroads, irregular roads, rivers, areas of cultivated land and bare soil in various shapes, and many round or square plots, which poses a challenge for many classifiers.

The precision of the different approaches for the Kansas AHSI dataset is shown in Table V, where the best results are marked in bold.

Table V shows that UML and FPGA obtain the best and second best result of 95.57% and 94.50%, respectively, on the Kansas AHSI dataset. SVM with the optimal parameter settings obtains the worst accuracy. The OA of UML is higher than that of SVM by over 10%. The precision for the "Road" class in all the methods is below 85%, which can be ascribed to the fact that a pixel of "Road" is just one pixel in a hyperspectral image with only a 30-m spatial resolution. In this scene, the accuracy of the CNN and MSNet methods with a patch size of 8 is better than that with a patch size of 4, and all the patch-based methods have an accuracy of less than 90%.

Fig. 10 shows that the salt-and-pepper noise is again very serious in the result of SVM. Although the results of CNN-8 and FPGA contain very little salt-and-pepper noise, which suggests that the convolution operation can restrain the salt-and-pepper noise, the pixels belonging to the "Road" category are misclassified. Moreover, the farm track in the result of FPGA is merged with the "Crop" category. The result of A^2S^2RN and MSRN cannot obtain the best performance in patch-based methods, and the "Road" pixel cannot be classified by the MSRN correctly. With the increase of the patch size, the phenomenon of merged classes is less significant in the results of MSNet, and the trivial characteristics are retained well, which proves that the multiscale shuffle block can tackle the loss of spatial information. The missing trivial objects in the results of UML are clearly improved when compared with FPGA and CNN-8, but there are still more missing trivial objects than in the results of MSNet and SVM. Therefore, a tradeoff should be made between restraining the salt-and-pepper noise and keeping the trivial characteristics. From Fig. 10 and Table V, it can be seen that a good tradeoff is achieved by UML, compared with the other counterpart methods.

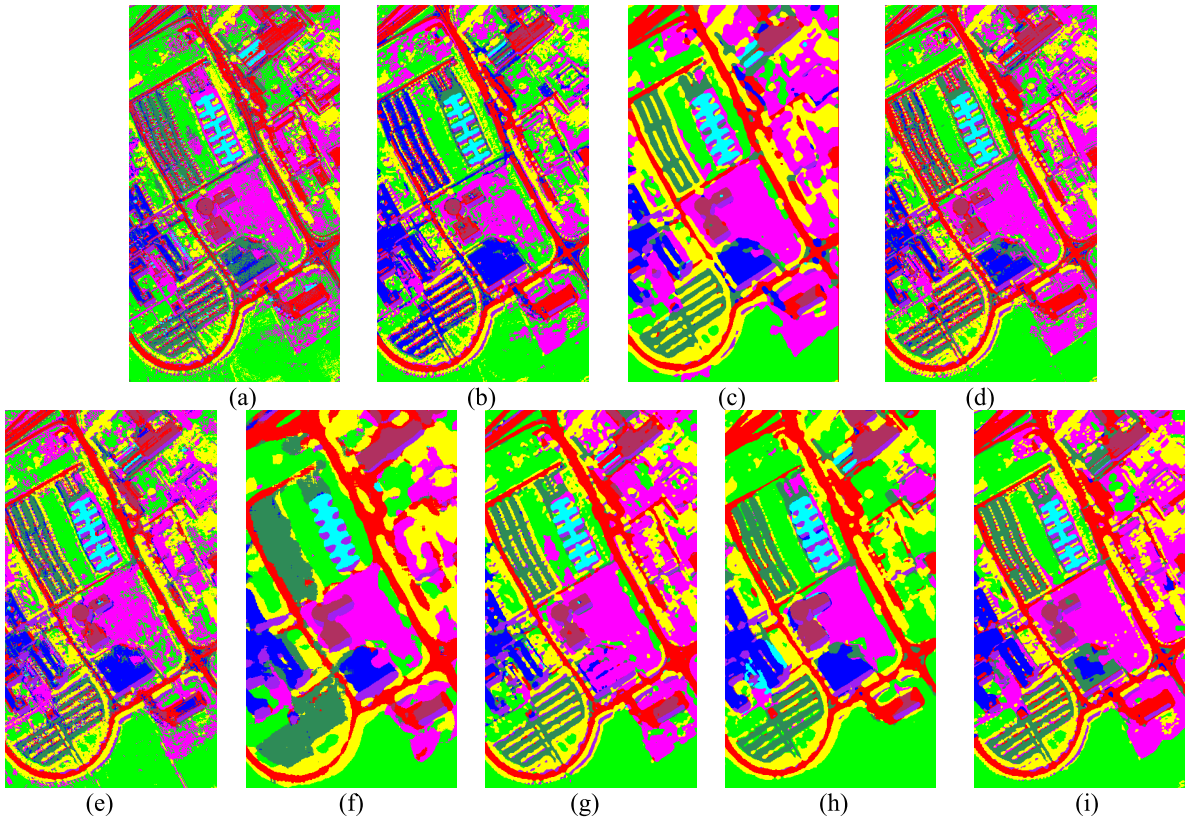

Fig. 9. Results obtained by the different approaches with the Pavia University ROSIS dataset. (a) SVM. (b) CNN-4. (c) CNN-8. (d) MSNet-4. (e) MSNet-8. (f) MSRN. (g) A^2S^2RN. (h) FPGA. (i) UML.

TABLE V
PRECISION OF THE DIFFERENT APPROACHES FOR THE KANSAS AHSI DATASET

Category	SVM	CNN-4	CNN-8	MSNet - 4	MSNet - 8	MSRN	A^2S^2RN	FPGA	UML
Crop	97.73	95.59	96.80	94.41	94.15	94.23	94.75	98.59	99.10
Meadow	95.18	92.53	97.17	96.49	95.08	87.46	81.78	99.39	99.07
Business Area	82.26	77.50	75.96	75.71	78.75	89.03	93.98	91.25	95.04
Bare Soil	83.10	89.32	90.35	92.95	94.62	91.13	94.25	94.04	96.32
Residential Area	87.16	90.39	97.47	95.46	98.17	98.14	97.11	98.24	97.74
Water	90.81	83.03	86.12	92.22	91.11	96.13	94.45	98.24	98.23
Road	62.02	60.65	67.18	55.70	55.65	53.96	53.63	80.47	81.01
OA	85.20	86.04	89.12	88.11	89.17	88.29	88.32	94.50	**95.57**
KAPPA	0.8212	0.8303	0.8676	0.8547	0.8675	0.8573	0.8568	0.9331	**0.9459**

D. Experiments With the Indian Pines AVIRIS Dataset

The Indian Pines AVIRIS dataset has much fewer labeled samples and more classes than the other two datasets. The labeled sample number for "Alfalfa," "Grass-Pasture-Mowed," and "Oats" is less than 50. In this case, all the labeled samples in the classes of "Alfalfa," "Grass-Pasture-Mowed," and "Oats" were used as training data. As shown in Table VI, 50 labeled samples of the other categories were sampled as the training data.

For the optimized phase of SVM, the validation dataset was 5% of the whole training data. The batch size of the deep learning methods was set to 30. After the model training, all the pixels were predicted and the evaluation indicators were calculated. The experiments were again repeated ten times over the randomly split training and test data. This dataset is composed of various types of crop fields. The spectral differences between the different vegetation types are also slight. The crop fields are in many kinds of regular shape, among which some small blocks of artificial objects are distributed. There are two large roads running from left to right in the whole scene, and there are many paths between the different crop fields.

The precision of the different approaches for the Indian Pines AVIRIS dataset is shown in Table VII, where the best

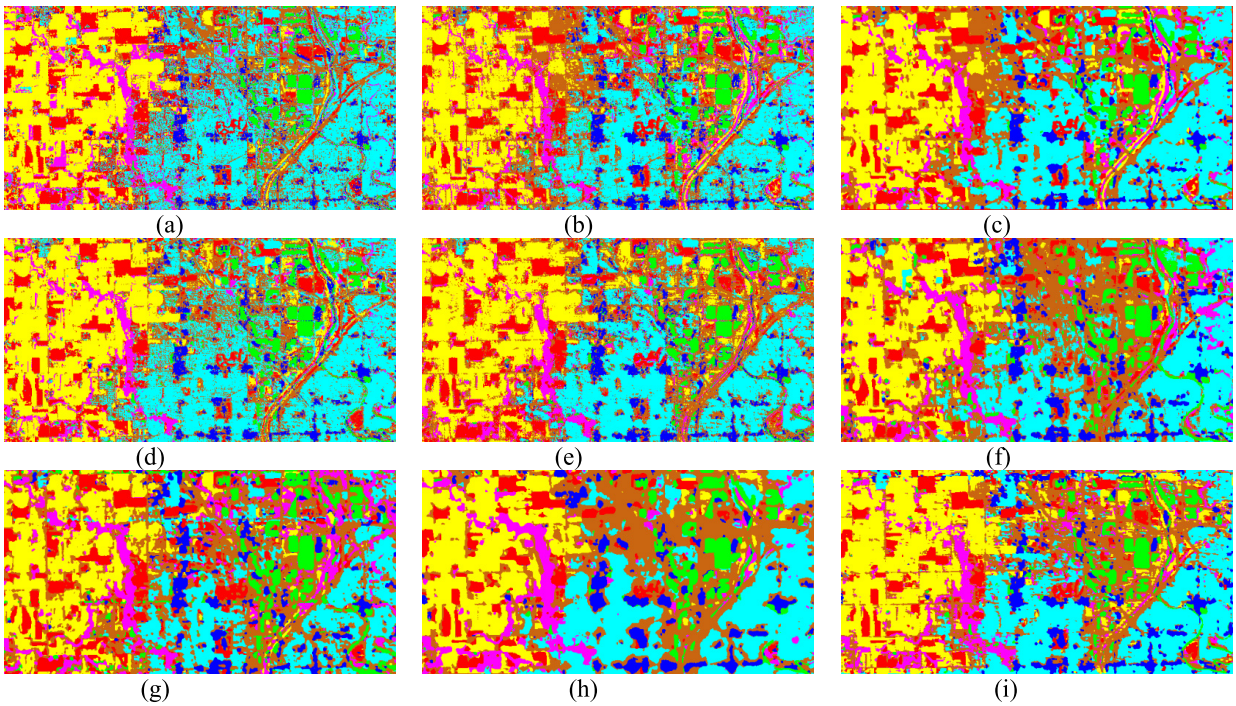

Fig. 10. Results obtained by the different approaches for the Kansas AHSI dataset. (a) SVM. (b) CNN-4. (c) CNN-8. (d) MSNet-4. (e) MSNet-8. (f) MSRN. (g) A^2S^2RN. (h) FPGA. (i) UML.

TABLE VI
NUMBER OF LABELED PIXELS IN EACH CLASS IN THE INDIAN PINES AVIRIS DATASET

Class name	Labeled sample number	Training set number
Alfalfa	46	46
Corn-Notill	1428	50
Corn-Mintill	830	50
Corn	237	50
Grass-Pasture	483	50
Grass-Trees	730	50
Grass-Pasture-Mowed	28	28
Hay-Windrowed	478	50
Oats	20	20
Soybean-Notill	972	50
Soybean-Mintill	2455	50
Soybean-Clean	593	50
Wheat	205	50
Woods	1265	50
Bs-Grss-Trs-Drs	386	50
Stone-Steel-Towers	93	50

results are marked in bold. UML and FPGA obtain the best and second best results of 96.44% and 96.18%, respectively, on the Indian Pines AVIRIS dataset. SVM with the optimal parameter settings obtains the worst accuracy. The OA of UML is higher than that of SVM by over 20%. There are six categories where UML achieves a 100% accuracy. There are significant gaps in the OA and kappa coefficient between the patch-based CNN and the FCN-based methods. MSNet obtains a worse performance than the conventional CNN for both patch sizes. Because of the additional block, the trainable parameters in MSNet are more than in the CNN. Moreover, the result of A^2S^2RN obtains the best performance of 93.34% in patch-based methods. The classification map of MSRN presents the sawtooth phenomenon on some categories. The 50 samples per category may be insufficient for the MSNet training, which may be the reason why it obtained a poor performance.

Fig. 11 shows that the results of SVM and MSNet-4 contain heavy salt-and-pepper noise. FPGA obtains the smoothest results among the seven methods. The two roads running through the scene can be recognized in the results of SVM, CNN-4, MSNet-4, MSNet-8, and UML, while in the results of CNN-8 and FPGA, the road boundaries are obscured and the road is classified as the neighboring crop types. The red box marks a square cropland area without any label information. The results of UML and MSNet present this cropland area as having a very clear square border, while the boundary does not appear as a square in the results of CNN-8 and FPGA. Fig. 11 and Table VII show the superiority of the UML framework over the other counterpart methods.

E. Experiments on Different Training and Test Samples

To explore the performance of the proposed method on different datasets, two kinds of training scenarios have been included in this section: 1) using the raining dataset with different sizes per class and 2) using the spatially disjoint training and test samples, which provided by the IEEE GRSS DASE (http://dase.grss-ieee.org/) [42].

In the first kind of training scenario, the training size is set to 10, 30, 50, 70, and 100 for different court parts. The OAs have been reported in Fig. 12. The OA of UML is not

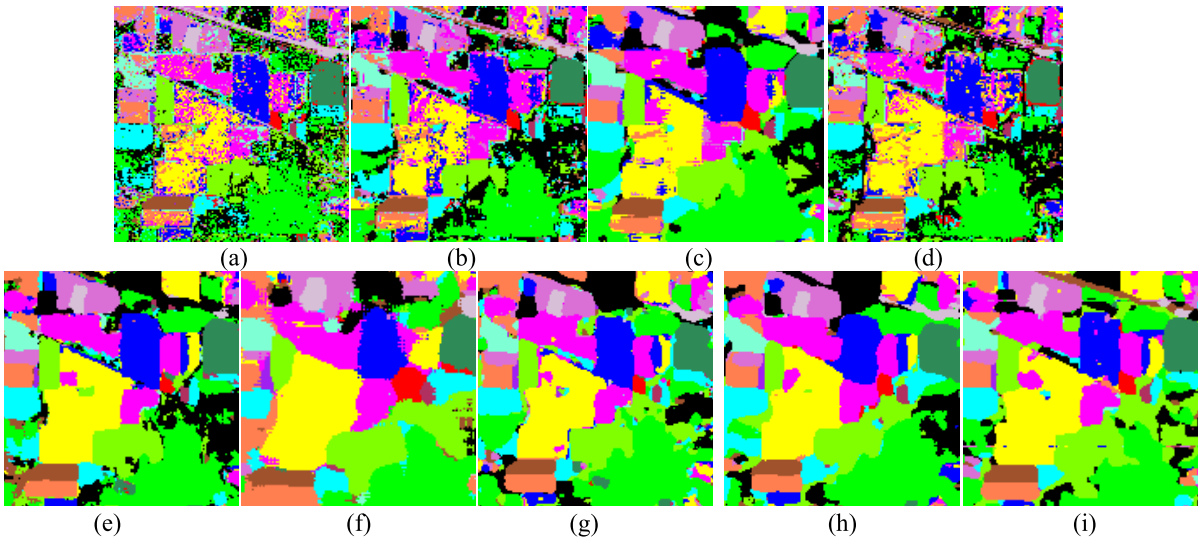

Fig. 11. Results obtained by the different approaches for the Indian Pines AVIRIS dataset. (a) SVM. (b) CNN-4. (c) CNN-8. (d) MSNet-4. (e) MSNet-8. (f) MSRN. (g) A^2S^2RN. (h) FPGA. (i) UML.

TABLE VII
PRECISION OF THE DIFFERENT APPROACHES FOR THE INDIAN PINES AVIRIS DATASET

Category	SVM	CNN-4	CNN-8	MSNet -4	MSNet -8	MSRN	A^2S^2RN	FPGA	UML
Alfalfa	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Corn-Notill	82.76	81.93	80.48	53.27	57.60	70.93	90.09	93.92	97.25
Corn-Mintill	82.7	64.19	87.27	74.8	58.68	83.79	94.31	97.2	98.85
Corn	90.51	89.84	96.26	77.01	91.53	98.73	94.86	100.00	100.00
Grass-Pasture	93.21	92.84	90.12	88.91	90.68	92.57	99.23	96.28	99.08
Grass-Trees	93.81	86.76	99.26	91.02	96.72	93.97	99.85	99.69	100.00
Grass-Pasture-Mowed	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Hay-Windrowed	98.15	95.09	100.00	94.86	98.33	99.12	99.75	99.5	100.00
Oats	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Soybean-Notill	89.79	85.96	89.49	78.83	82.24	77.74	90.98	97.42	97.51
Soybean-Mintill	56.57	56.5	78.13	63.421	68.47	82.13	89.31	98.88	92.69
Soybean-Clean	95.94	69.61	93.52	67.59	64.53	73.98	96.18	96.49	97.43
Wheat	95.23	85.16	92.26	85.16	81.38	82.26	91.33	99.2	99.360
Woods	67.47	84.86	96.54	88.89	90.46	88.92	96.45	99.24	99.26
Bs-Grss-Trs-Drs	99.26	79.87	86.18	83.12	75.95	87.77	96.00	100.00	100.00
Stone-Steel-Towers	100.00	88.37	86.05	100.00	93.94	100.00	94.74	100.00	100.00
OA	77.88	76.16	87.55	73.86	75.46	83.21	93.34	96.25	**97.12**
KAPPA	0.7504	0.7306	0.8579	0.7044	0.7196	0.8062	0.9233	95.67	**0.9669**

the highest with ten samples per class, and when the training size becomes bigger, the UML has higher OA than others. The reason is that UML contains more modules than other methods which need a sufficient labeled sample to train.

In the second kind of scenario, the classification performance of different methods with spatially disjoint training and test samples is compared among different methods on the Indian Pines AVIRIS dataset and the Pavia University ROSIS dataset. The experimental setting of each method is the same as the previous experiments, while the training samples, the test samples, and the evaluation standards are provided by the IEEE GRSS DASE. The experimental results are reported in Table VIII. The performance for all methods is degraded compared with previous classification results. It can be noticed that the OAs of CNN-4, CNN-8, and MSnet-4 are lower than 80% on the two datasets, which reveals that the random selection strategy for samples will lead to an overestimation of classification performance. The location of training and test samples is adjacent and the adjacent samples are easily classified.

Despite the performance degradation phenomenon, the UML still achieves the highest OA of 83.07 and 90.24 and kappa of 0.79 and 0.88 on the Pavia University ROSIS dataset and the Indian Pines AVIRIS dataset, respectively. The results have been demonstrated that the proposed method has the effective spatiospectral feature learning ability.

F. Experiments in Transfer Learning on the Three Datasets

These experiments were carried out to investigate the performance of the UML framework in transfer learning. The UML

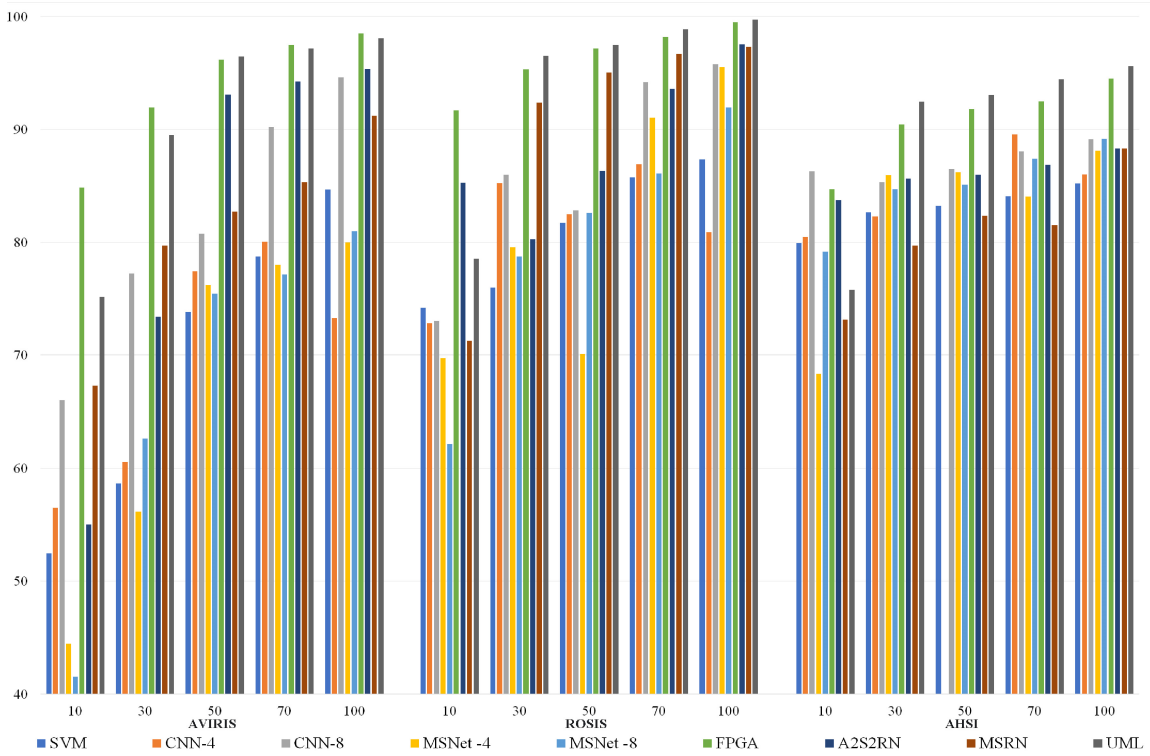

Fig. 12. OAs of the different approaches on different training sizes.

TABLE VIII
PRECISION OF THE DIFFERENT APPROACHES ON THE DISJOINT TRAIN TEST

Methods	ROSIS		AVIRIS	
	OA	Kappa	OA	Kappa
SVM	73.5	0.70	81.33	0.79
CNN-4	77.96	0.76	71.16	0.70
CNN-8	76.87	0.74	74.48	0.71
MSNet -4	79.52	0.76	73.61	0.70
MSNet -8	80.6	0.78	81.07	0.75
FPGA	81.24	0.78	89.75	0.88
A^2S^2RN	81.16	0.79	86.52	0.83
MSRN	80.33	0.76	87.29	0.85
UML	**83.07**	**0.79**	**90.24**	**0.88**

TABLE IX
PRECISION OF THE TRANSFER LEARNING EXPERIMENTS FOR THE KANSAS AHSI DATASET

Category	ROSIS _to_AHSI	AVIRIS _to_AHSI	AHSI
Crop	97.69	90.74	91.72
Meadow	96.79	95.29	72.26
Business Area	91.87	79.45	86.87
Bare Soil	83.53	85.78	66.44
Residential Area	88.69	85.46	88.45
Water	89.54	96.29	91.75
Road	61.08	57.14	46.82
OA	**86.75**	84.16	75.81
KAPPA	**0.8402**	0.8087	0.7117

framework was first trained on the source image to a sufficient degree. When performing classification in the other scenes, the initial block and the classification block of UML were replaced by the particular block corresponding to the target scene, which had a different input dimension and softmax dimension. The input dimension was consistent with the number of spectral bands in the other sensor, and the softmax dimension was set according to the category numbers in different tasks. The other blocks of UML were preserved from the source domain. After this, the labeled samples in the target hyperspectral image were extracted for fine-tuning. Ten labeled samples were selected in each category for the target domain training data. For the comparative experimental setting, an additional UML framework was trained with only ten labeled samples in each category in the target domain scene, which can be regarded as an experiment without any transfer information. The experiments included nine transfer learning cases, i.e., training on the ROSIS dataset and transferring to the AVIRIS dataset, training on the AHSI dataset and transferring to the AVIRIS dataset, training on the AVIRIS dataset, training on the AVIRIS dataset and transferring to the ROSIS dataset, training on the AHSI dataset and transferring to the ROSIS dataset, training on the ROSIS dataset, training on the AVIRIS dataset and transferring to the AHSI dataset, training on the ROSIS dataset and transferring to the AHSI dataset, and training on the AHSI dataset. The results of the different transfer learning experiments are shown in Tables IX–XI and Fig. 13.

TABLE X
PRECISION OF THE TRANSFER LEARNING EXPERIMENTS FOR THE PAVIA UNIVERSITY ROSIS DATASET

Category	AHSI to_ROSIS	AVIRIS_to_ ROSIS	ROSIS
Asphalt	96.90	87.52	91.66
Meadows	90.65	88.66	73.60
Gravel	96.84	85.88	96.55
Trees	95.15	94.99	94.56
Painted Metal Sheets	100.00	100.00	100.00
Bare Soil	78.68	95.60	94.98
Bitumen	91.97	98.41	94.70
Self-Blocking Bricks	82.79	92.37	96.70
Shadows	99.47	98.19	99.89
OA	90.69	**90.80**	85.58
KAPPA	0.8778	**0.8806**	0.8175

TABLE XI
PRECISION OF THE TRANSFER LEARNING EXPERIMENTS FOR THE INDIAN PINES AVIRIS DATASET

Category	ROSIS_to_ AVIRIS	AHSI _to_ AVIRIS	AVIRIS
Alfalfa	94.44	100.00	100.00
Corn-Notill	46.97	56.91	54.23
Corn-Mintill	89.39	87.20	71.95
Corn	70.93	63.44	69.60
Grass-Pasture	82.66	90.27	88.37
Grass-Trees	93.19	89.31	99.44
Grass-Pasture-Mowed	100.00	100.00	100.00
Hay-Windrowed	100.00	99.57	97.44
Oats	100.00	100.00	100.00
Soybean-Notill	79.63	87.01	80.25
Soybean-Mintill	78.77	76.36	56.24
Soybean-Clean	81.82	88.51	80.10
Wheat	99.49	100.00	100.00
Woods	86.29	81.04	93.63
Buildings-Grass-Trees-Drives	99.47	98.40	92.82
Stone-Steel-Towers	97.59	98.80	100.00
OA	79.82	**80.78**	75.20
KAPPA	0.7721	**0.7829**	0.7213

Table IX presents the results of the transfer learning experiments from the ROSIS and AVIRIS datasets to the AHSI dataset, which shows that the OA is improved by 10% by the transfer learning, which is the biggest among the three datasets. The source domain ROSIS and AVIRIS images are aerial images and have a high spatial resolution at 1.3 and 17 m/pixel, respectively, and thus contain more plentiful information than the target domain at 30 m/pixel. The trained UML framework is able to learn the abundant spatiospectral features in the aerial imagery, which is very helpful for the target domain fine-tuning. Moreover, the ROSIS dataset contains more labeled samples, resulting in higher accuracy than that of the AVIRIS dataset.

Table X lists the results for the transfer learning from the AHSI and AVIRIS datasets to the ROSIS dataset. The OA is improved by nearly 5% by the transfer learning, which is lower than in the previous scene. The reason for this may come down to the fact that the target domain has the highest spatial resolution, which limits the accuracy improvement. The AHSI dataset has more labeled samples, while the AVIRIS dataset has a higher resolution, which causes the difference between these two results to be only slight.

From Table XI, it can be seen that the results for the AVIRIS dataset have lower OAs than the results for the other two datasets, which may have been caused by the target domain AVIRIS dataset containing the most categories, i.e., the limited labeled data in the target domain are the biggest limitation for the performance. The UML framework trained from the AHSI dataset with a spectral range from 400 to 2500 nm has a better performance than when trained from the ROSIS dataset with a spectral range from 400 to 1000 nm. In terms of the AVIRIS dataset, a source domain with a consistent spectral range is more helpful for fine-tuning.

G. Complexity Analysis of the Methods

In this section, the FLOPs and running time are utilized for the complexity analysis of each method. The calculation of FLOPs is conducted on the whole image for the patch-based methods, such as CNN-4, CNN-8, MSnet-4, MSnet-8, A^2S^2RN, and MSRN. Each patch is utilized for the center pixel label prediction and the stride of prediction is set to 1. Therefore, the total FLOPs is a summation of all layers in the network for input hyperspectral patch times the pixel number. The running time of all methods is calculated on RTX 2080. Table XII reports the FLOPs and the running time of different methods on three datasets. The A^2S^2RN has the longest running time and biggest FLOPS, compared with all network-based methods. The UML is the second faster and is longer than FPGA within 1 s. The inference speed of FCN-based methods is faster than patch-based because the data patch should be obtained by sliding a window in a one-step stride, which involves many repetitive calculations. These redundant operations not only waste computational memory but also increase time consumption.

H. Ablation Analysis of the UML

The proposed UML contains many modules, such as shuffle block, upsampling, and multiscale strategy. To further demonstrate the contribution of different modules of the proposed UML, we carry out the ablation study on the three hyperspectral datasets. In this context, the UML has been designed upon the same network architecture of original UML but without each module individually. "Ablation_shuffle" indicates that the shuffle block is discarded in the UML. "Ablation_upsample" indicates that neither the deconvolution nor the interpolation is utilized in the decoder process of UML. "Ablation_attention"

TABLE XII

COMPLEXITY ANALYSIS OF THE DIFFERENT APPROACHES ON THREE HYPERSPECTRAL DATASETS

Dataset	AHSI		AVIRIS		ROSIS	
Indicator	Running time(s)	GFLOPs	Running time(s)	GFLOPs	Running time(s)	GFLOPs
CNN-4	26.53	120.34	5.58	11.63	22.02	113.24
CNN-8	35.24	129.04	5.07	12.47	25.19	121.54
MSNet -4	80.94	450.84	6.03	18.1	52.15	107.47
MSNet -8	92.03	1208.87	6.47	71.27	54.27	425.17
FPGA	2.47	175.76	1.23	17.13	1.41	128.34
A^2S^2RN	105.35	68867.05	12.96	3838.14	38.41	20775.63
MSRN	41.62	1996.21	5.82	161.75	16.72	1928.52
UML	2.49	211.16	1.98	21.46	2.03	168.51

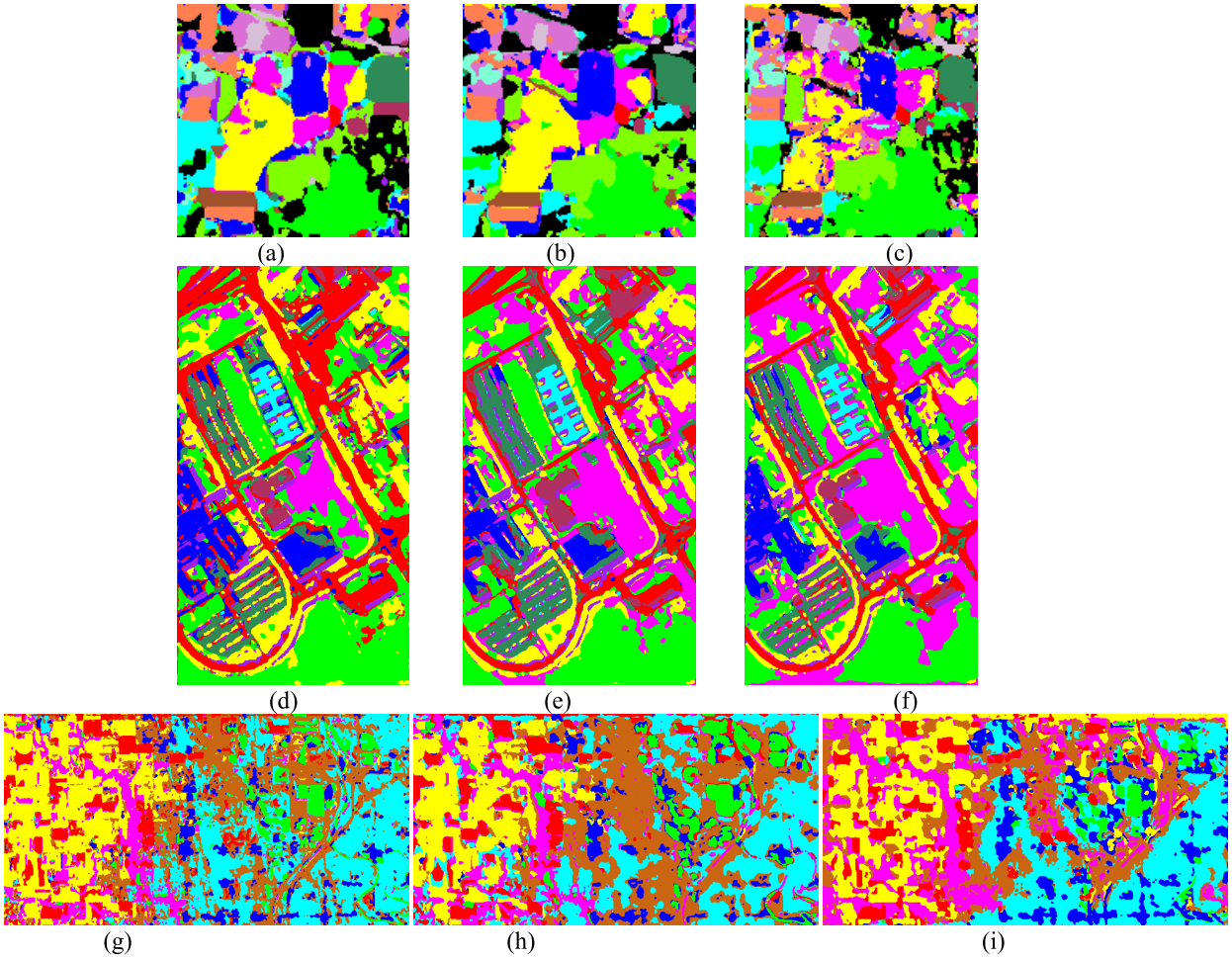

Fig. 13. Results obtained in the different transfer learning experiments for the three datasets. (a) ROSIS_to_ AVIRIS. (b) AHSI _to_ AVIRIS. (c) AVIRIS. (d) AHSI _to_ROSIS. (e) AVIRIS_to_ ROSIS. (f) ROSIS. (g) ROSIS_to_AHSI. (h) AVIRIS_to_AHSI. (i) AHSI.

TABLE XIII

ABLATION STUDY ON THE THREE HSI DATASETS

Description	ROSIS		AVIRIS		AHSI	
	OA	KAPPA	OA	KAPPA	OA	KAPPA
Ablation_shuffle	98.24	0.976	93.48	0.9215	94.92	0.9321
Ablation_upsample	98.01	0.9736	93.81	0.9293	92.04	0.9029
Ablation_attention	98.97	0.985	94.93	0.9417	95.16	0.9407
Ablation_mutiscale	96.46	0.9532	92.01	0.9089	94.25	0.9369
Original UML	**99.75**	**0.9967**	**97.12**	**0.9669**	**95.57**	**0.9459**

means each CNN operation in the UML without spatiospectral attention. "Ablation_mutiscale" means that the convolution has one fixed kernel in the shuffle block. The OA and kappa of the experiments are listed in Table XIII. The UML without multiscale obtained the worst OA and kappa on the ROSIS and AVIRIS datasets, which demonstrated that the multiscale

module can improve spatial feature learning and yield the better performance. The result achieved by the original UML is the best, which proved that each module is beneficial for the proposed network.

V. Conclusion

In this article, to tackle the issues of redundant operation and land-cover map distortion in hyperspectral image classification, we propose an innovative FCN named the UML framework, which combines both an encoding phase and a decoding phase. The multiscale spatiochannel attention mechanism and a multiscale shuffle block are included to obtain a more powerful classification performance so that the UML framework can not only perform classification but can also be used in transfer learning applications. From the results obtained with three hyperspectral datasets, including two airborne hyperspectral images and one spaceborne hyperspectral image, it was confirmed that the UML framework can outperform the other state-of-the-art hyperspectral image classification methods and can obtain a good tradeoff between restraining salt-and-pepper noise and keeping trivial characteristics. Moreover, the transfer learning ability of the UML framework was also confirmed in the transfer learning experiments. When the source domain is an aerial hyperspectral image and the target domain is a spaceborne hyperspectral image, the UML framework can obtain a superior performance. However, there are two limitations to the UML framework: 1) the information from a large amount of unlabeled pixels cannot be made full use of in the encoding and decoding framework and 2) the promotion of the transfer learning is limited.

In our future work, semisupervised learning will be considered in the UML framework. In addition, domain adaptation will also be considered in the UML framework during the transfer learning process.

Acknowledgment

The authors would like to thank Prof. David Landgrebe, Prof. Paolo Gamba, and the Shanghai Institute of Technical Physics, Chinese Academy of Sciences, for providing the data used in the experiments.

References

[1] K. Tan *et al.*, "Vicarious calibration for the AHSI instrument of Gaofen-5 with reference to the CRCS Dunhuang test site," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 4, pp. 3409–3419, Apr. 2021.

[2] X. Wang, K. Tan, Q. Du, Y. Chen, and P. Du, "Caps-TripleGAN: GAN-assisted capsnet for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 7232–7245, Sep. 2019.

[3] K. Tan, H. Wang, L. Chen, Q. Du, P. Du, and C. Pan, "Estimation of the spatial distribution of heavy metal in agricultural soils using airborne hyperspectral imaging and random forest," *J. Hazardous Mater.*, vol. 382, Jan. 2020, Art. no. 120987.

[4] K. Tan *et al.*, "Estimating the distribution trend of soil heavy metals in mining area from HyMap airborne hyperspectral imagery based on ensemble learning," *J. Hazardous Mater.*, vol. 401, Jan. 2021, Art. no. 123288.

[5] S. Wang, X. Wang, L. Zhang, and Y. Zhong, "Auto-AD: Autonomous hyperspectral anomaly detection network based on fully convolutional autoencoder," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–14, 2022.

[6] M. Shimoni, R. Haelterman, and C. Perneel, "Hypersectral imaging for military and security applications: Combining myriad processing and sensing techniques," *IEEE Geosci. Remote Sens. Mag.*, vol. 7, no. 2, pp. 101–117, Jun. 2019.

[7] J. Peng and Q. Du, "Robust joint sparse representation based on maximum correntropy criterion for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 12, pp. 7152–7164, Dec. 2017.

[8] J. Peng, L. Li, and Y. Y. Tang, "Maximum likelihood estimation-based joint sparse representation for the classification of hyperspectral remote sensing images," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 6, pp. 1790–1802, Jun. 2019.

[9] Y. Zhang, X. Wang, X. Jiang, and Y. Zhou, "Marginalized graph self-representation for unsupervised hyperspectral band selection," *IEEE Trans. Geosci. Remote Sens.*, early access, Oct. 21, 2021, doi: 10.1109/TGRS.2021.3121671.

[10] J. Jiang, J. Ma, C. Chen, Z. Wang, Z. Cai, and L. Wang, "SuperPCA: A superpixelwise PCA approach for unsupervised feature extraction of hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4581–4593, Aug. 2018.

[11] C. Wang, L. Zhang, W. Wei, and Y. Zhang, "When low rank representation based hyperspectral imagery classification meets segmented stacked denoising auto-encoder based spatial–spectral feature," *Remote Sens.*, vol. 10, no. 2, p. 284, 2018.

[12] P. Zhong, Z. Gong, S. Li, and C.-B. Schönlieb, "Learning to diversify deep belief networks for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 55, no. 6, pp. 3516–3530, Jun. 2017.

[13] S. K. Roy, P. Kar, D. Hong, X. Wu, A. Plaza, and J. Chanussot, "Revisiting deep hyperspectral feature extraction networks via gradient centralized convolution," *IEEE Trans. Geosci. Remote Sens.*, early access, Oct. 14, 2021, doi: 10.1109/TGRS.2021.3120198.

[14] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. Sensors*, vol. 2015, Jul. 2015, Art. no. 258619.

[15] Z. Ge, G. Cao, X. Li, and P. Fu, "Hyperspectral image classification method based on 2D–3D CNN and multibranch feature fusion," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 5776–5788, 2020.

[16] S. Hao, W. Wang, Y. Ye, T. Nie, and L. Bruzzone, "Two-stream deep architecture for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 2349–2361, Apr. 2018.

[17] J. Yang, Y.-Q. Zhao, and J. C.-W. Chan, "Learning and transferring deep joint spectral-spatial features for hyperspectral classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4729–4742, Aug. 2017.

[18] W. Yao, C. Lian, and L. Bruzzone, "ClusterCNN: Clustering-based feature learning for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 11, pp. 1991–1995, Nov. 2021.

[19] S. Kumar Roy, R. Mondal, M. E. Paoletti, J. M. Haut, and A. Plaza, "Morphological convolutional neural networks for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 8689–8702, 2021.

[20] 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.

[21] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[22] S. Jia *et al.*, "A lightweight convolutional neural network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 5, pp. 4150–4163, May 2021.

[23] H. Zhang, Y. Li, Y. Jiang, P. Wang, Q. Shen, and C. Shen, "Hyperspectral classification based on lightweight 3-D-CNN with transfer learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5813–5828, Aug. 2019.

[24] J. Li, X. Zhao, Y. Li, Q. Du, B. Xi, and J. Hu, "Classification of hyperspectral imagery using a new fully convolutional neural network," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 2, pp. 292–296, Feb. 2018.

[25] Z. Zheng, Y. Zhong, A. Ma, and L. Zhang, "FPGA: Fast patch-free global learning framework for fully end-to-end hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 8, pp. 5612–5626, Aug. 2020.

[26] Y. Zhong, X. Hu, C. Luo, X. Wang, J. Zhao, and L. Zhang, "WHU-Hi: UAV-borne hyperspectral with high spatial resolution (H^2) benchmark datasets and classifier for precise crop identification based on deep convolutional neural network with CRF," *Remote Sens. Environ.*, vol. 250, Dec. 2020, Art. no. 112012.

[27] N. Wambugu *et al.*, "Hyperspectral image classification on insufficient-sample and feature learning using deep neural networks: A review," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 105, Dec. 2021, Art. no. 102603.

[28] Y. Xu, L. Zhang, B. Du, and F. Zhang, "Spectral-spatial unified networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 10, pp. 5893–5909, Oct. 2018.

[29] H. Gao, Y. Yang, C. Li, L. Gao, and B. Zhang, "Multiscale residual network with mixed depthwise convolution for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 4, pp. 3396–3408, Apr. 2021.

[30] S. K. Roy, S. Manna, T. Song, and L. Bruzzone, "Attention-based adaptive spectral–spatial kernel ResNet for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 9, pp. 7831–7843, Sep. 2021.

[31] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

[32] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[33] J. Park, S. Woo, J.-Y. Lee, and I. So Kweon, "BAM: Bottleneck attention module," 2018, *arXiv:1807.06514*.

[34] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[35] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

[36] W. Zhao, Z. Guo, J. Yue, X. Zhang, and L. Luo, "On combining multiscale deep learning features for the classification of hyperspectral remote sensing imagery," *Int. J. Remote Sens.*, vol. 36, no. 13, pp. 3368–3379, 2015.

[37] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.

[38] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 116–131.

[39] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[40] J. Yue, S. Mao, and M. Li, "A deep learning framework for hyperspectral image classification using spatial pyramid pooling," *Remote Sens. Lett.*, vol. 7, no. 9, pp. 875–884, 2016.

[41] J. Gauthier, "Conditional generative adversarial nets for convolutional face generation," *Class Project Stanford CS231N: Convolutional Neural Netw. Vis. Recognit., Winter Semester*, vol. 2014, no. 5, p. 2, 2014.

[42] M. Ahmad *et al.*, "Hyperspectral image classification—Traditional to deep models: A survey for future prospects," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 15, pp. 968–999, 2021, doi: 10.1109/JSTARS.2021.3133021.