Image-based neural architecture automatic search method for hyperspectral image classification

Zhonggang Hu,[a,b] **Wenxing Bao**,[a,b,*] **Kewen Qu**,[a,b] **and Hongbo Liang**[a,b]

[a]North Minzu University, School of Computer Science and Engineering, Yinchuan, China
[b]North Minzu University, State Ethnic Affairs Commission, Key Laboratory of Images and Graphics Intelligent Processing, Yinchuan, China

Abstract. Convolutional neural networks (CNNs) have shown excellent performance for hyperspectral image (HSI) classification due to their characteristics of both local connectivity and sharing weights. Nevertheless, with the in-depth study of network architecture, merely manual empirical design can no longer meet the current scenario needs. In addition, the existing CNN-based frameworks are heavily affected by the redundant three-dimensional cubes of the input and result in inefficient description issues of HSIs. We propose an image-based neural architecture automatic search framework (I-NAS) as an alternative to CNN. First, to alleviate the redundant spectral–spatial distribution, I-NAS feeds a full image into the framework via a label masking fashion. Second, an end-to-end cell-based structure search space is considered to enrich the feature representation. Then, it determined the optimal cells by employing a gradient descent search algorithm. Finally, the well-trained CNN architecture is automatically constructed by stacking the optimal cells. The experimental results from two real HSI datasets indicate that our proposal can provide a competitive performance in classification. [DOI: 10.1117/1.JRS.16.016501]

Keywords: full image; hyperspectral image classification; neural architecture; automatic search; convolution neural network; feature representation.

Paper 210553 received Aug. 27, 2021; accepted for publication Dec. 20, 2021; published online Jan. 6, 2022.

1 Introduction

Hyperspectral image (HSI) consists of hundreds of narrow and continuous spectral bands, in which abundant spectral information enhances the ability to identify land cover materials.[1–5] With the rapid development of imaging sensor technology, the acquisition of high spatial resolution HSI is no longer a challenge, which provides an ideal research base for environmental protection,[6] land monitoring,[7] and urban development.[8] Hyperspectral image classification (HSIC), which provides a unique land cover category for each pixel according to spectral signatures and spatial context, is an important task in these remote sensing applications.[9–11] Typical HSIC methods consist of two separate steps: feature extraction and classifier training. However, facing the challenges of high-dimensional nonlinear distribution and redundant bands of information, traditional methods generally suffer from low generalization ability and limited degree of feature representation.

With the emergence of deep learning thinking in various vision tasks, scholars have tried to apply the remarkable feature representation capability of neural networks to hyperspectral interpretation.[12–15] In such approaches, feature extraction and classification are integrated into an end-to-end network framework. It allows researchers to focus more on the design of the network architecture to obtain superior recognition performance. For example, Hu et al.[16] first proposed a convolutional neural network (CNN) constructed by multiple one-dimensional (1D) spectral convolution layers for HSIC; Yu et al.[17] improved the classification performance by

*Address all correspondenc to Wenxing Bao, baowenxing@nun.edu.cn

embedding hash features extracted from the spectrum into the CNN architecture. However, unlike common natural images, HSI is essentially a third-order tensor containing two spatial dimensions and one spectral dimension.[18] It is very important for HSIC to integrate spatial and spectral information.[19–21] CNNs allow the use of spatial HSI patches as data input, providing a natural way to merge spatial contextual information through its local receptive domain to improve classification performance.[22] To effectively utilize spatial information, Li et al.[23] combined CNN model with pixel pairs to learn discriminative features and use a majority voting strategy to obtain final classification results. Zhao et al.[24] combined a CNN model-based spatial feature extraction process with a spectral feature extraction process based on balanced local discriminative embedding to superimpose the obtained features and then perform the final classification step. Although these methods incorporate different techniques on the basis of CNN to extract spectral–spatial information separately, they do not consider the inherent continuity of the three-dimensional (3D) HSI cubes. In contrast, the 3D CNN approach takes the neighborhood cube of the raw HSI as input data and computes a 3D convolution kernel for each pixel and its spatial neighborhood and the corresponding spectral information. For example, Mei et al.[25] used a 3D convolutional autoencoder network to learn the spectral–spatial features of the HSI. Roy et al.[26] proposed a hybrid spectral CNN for HSIC, which consists of a spatial 2D-CNN and a spectral–spatial 3D-CNN to join spectral–spatial feature representation. However, the classification accuracy of the above CNN models decreases with increasing network depth, and the network structure of depth is prone to Hughes phenomenon.

To alleviate the above problem, Zhong et al.[27] constructed spectral and spatial residual blocks for accelerating the backpropagation of the network framework to prevent gradient explosion, respectively. Wang et al.[28] proposed an end-to-end fast, dense spectral–spatial convolution framework, which uses dynamic learning rates, parametric corrected linear units, batch normalization, and dropout layers to increase speed and prevent overfitting. In addition, to take full advantage of the positional relationships in the HSI pixel vector, Zhu et al.[29] proposed a CNN-based capsule network (CapsNet). The CapsNet architecture uses local connections and shared transform matrices. In CapsNet, the number of trainable parameters is reduced, which has the potential to alleviate the overfitting problem when the number of available training samples is limited. In addition, the generative model can generate high-quality samples to alleviate the above problems.[30,31] Wang et al.[32] designed an adaptive dropblock and embedded it into a generative adversarial network (ADGAN) to alleviate problems such as training data imbalance and pattern collapse in HSIC. However, the architectures of these models were designed manually by experts. Designing an appropriate neural structure is a key aspect of classifiers based on neural network, which requires a large amount of prior knowledge and is a time-consuming and trial and error process.

Recently, neural architecture search (NAS) framework has attracted much attention because it can automatically search the structure of neural networks. To design a suitable CNN architecture, Chen et al.[33] proposed an automated CNN approach for HSIC for the first time, constructed a cell-based search space, which uses NAS to search CNN architectures, and 1D auto-CNN and 3D auto-CNN based on the gradient descent method as spectral and spectral space HSI classifiers, respectively. Zhang et al.[34] applied the particle swarm optimization (PSO) method to CNN architecture search, which is able to obtain the global optimal architecture, and designed a new direct encoding strategy to encode the structure into particles and use PSO algorithm to find the optimal deep structure from the particle swarm. Compared with existing deep learning methods, the NAS-based method has better performance.

Although the above methods have made significant progress in HSIC tasks in recent years, when these methods are used for HSIC, there is a large amount of redundant information between neighboring cubes due to the high data complexity of HSI [see Fig. 1(a)], and it is easy to see from Figs. 1(a) and 1(b) that the redundant information increases as the patch size keeps increasing. Therefore, training on the neighborhood cube will consume more training time and memory than on the raw image, and it is difficult to design a classification model based on HSI patches to fit arbitrary images. Meanwhile, the model uses HSI neighborhood cubes as data input, which limits the scope of using spatial neighborhood information.

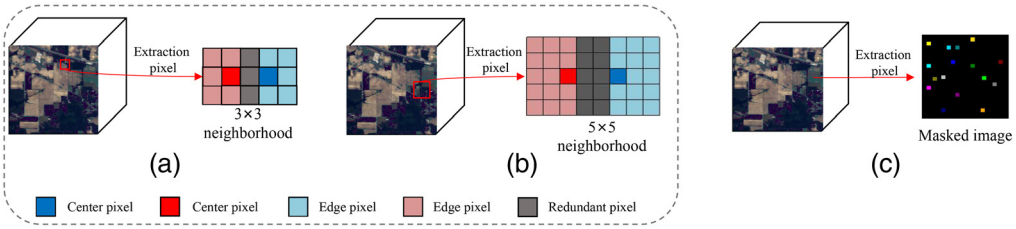

Fig. 1 Feature extraction of HSI: (a) a 3×3 neighboring block, (b) a 5×5 neighboring block, and (c) a masked image.

To alleviate the data redundancy problem of HSI patches, expand the application scope of spatial neighborhood information and improve the processing efficiency in training and testing, Cui et al.[35] proposed an image-based classification framework using image as data input [see Fig. 1(c)] and proposed a multiscale spatial–spectral CNN (HyMSCN) for HSI to integrate multiple receptive fields fused features and different levels of multiscale spatial features. However, it does not enable automatic search of the neural network structure.

Although using images as data input can improve processing efficiency in training and testing, the pooling operation in CNN is based on downsampling to a manageable level of feature space size, which logically, inevitably causes information loss. To alleviate this problem, this paper proposes an end-to-end cell structure in which the input nodes are merged into the output node to further come to enrich the feature information of the input images and ensure the accuracy of classification.

In this context, an image-based neural structure automatic search (I-NAS) method in this paper is proposed to further improve the processing efficiency of training and testing and to ensure the classification accuracy while realizing the neural structure automatic search. Specifically, I-NAS first uses the masked image containing training samples as the input to the network architecture search and classification model, while extracting the spatial location coordinate information of pixel points. Second, an end-to-end cell search space is constructed and select some operations including convolution and pooling. Then a gradient descent-based search algorithm is used to find the cell structure with the best classification performance on the validation dataset. Finally, a CNN classification model is constructed by stacking the cells. In the testing phase, the masked image containing the test samples is used as the input to the CNN classification model and the corresponding labels of all pixels are predicted. The experimental results on multiple datasets show that, compared with using neighboring cubes as the network input, the proposed method improves the running time and has good classification performance.

This proposal in this paper addresses the following aspects:

1. An image-based neural architecture automatic search method (I-NAS) is proposed. The image-based framework increases the image spatial receptive field, reduces data redundancy, and improves the efficiency and performance of classification.
2. An end-to-end cell structure is proposed. In this cell structure, two input nodes are merged into the output node, reducing the feature information loss of the input image due to convolution and pooling operations.
3. On two well-known HSI datasets, I-NAS takes significantly less time in the training and testing phases than other deep learning models. I-NAS takes significantly less time and memory in the architecture search phase than the patch-based neural architecture search algorithm (P-NAS).

The remainder of this paper is organized as follows. In Sec. 2, related works are briefly introduced. In Sec. 3, we introduce our algorithm, including the image-based neural-architecture automatic search framework and cell-based network-structure search. Section 4 presents the experimental results of our method and its comparison with other HSIC methods, discusses the time and space complexity of the experiment, and describes the optimal cells. Finally, the advantages and disadvantages of this method are presented in Sec. 5.

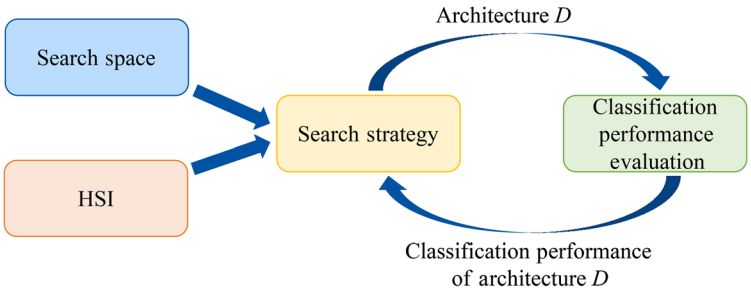

Fig. 2 Abstract description of NAS method. The search strategy selects an architecture from a defined search space. The architecture is passed to the performance evaluation strategy, which returns the estimated performance of the architecture to the search strategy. Architecture D is an optional architecture in the search space.

2 Related Work

2.1 *NAS*

NAS consists of three main components: search space, search strategy, and classification performance evaluation.[33,36] Figure 2 shows the process of designing an architecture using the NAS approach. The general process of NAS is to first construct a search space, which is a collection of optional CNN architectures. Then, the best network architecture in the search space is searched using a search strategy according to the results of the classification performance evaluation. The classification performance evaluation is to evaluate the classification performance of the optional CNN architectures in the search space using evaluation metrics.

Many different search strategies have been used for NAS, including reinforcement learning (RL), evolutionary method, and gradient-based method. For instance, Zoph et al.[37] proposed an RL-based NAS approach that uses an RNN to generate a model description of a neural network and uses RL to train this neural network to maximize the expected accuracy of the generated structures on the validation set.[38] In the search strategy based on evolutionary method,[39,40] each neural network structure is encoded as a digital sequence. Each digital sequence is trained and the performance on the validation set is used as the fitness of the digital sequence. Based on the fitness, a new high-performance neural network structure is generated. However, both search methods require several thousand hours of running with GPU during the architecture search phase to obtain the optimal architecture, which is too time-consuming. In contrast, in the gradient-based search strategy, Liu et al.[36] proposed a method to convert the discrete search space into a continuous space, which enables the gradient-based search method to obtain suitable neural structures and greatly reduces the training time required for the architecture search phase. Due to the simplicity and effectiveness of gradient-based search methods, gradient-based search strategies have become a hot research topic in neural structure search. In this paper, the gradient-based method is also used to search the network structure of cells.

3 Proposed Methodology

This section presents the framework of I-NAS. I-NAS first preprocesses the HSI dataset to obtain the masked training, validation, and test groups. Then, the training and validation groups are used as input for architecture search to obtain the optimal cell architecture to determine the final classification model. Finally, the test group is used as the input of the final classification model to obtain the classification map. The framework diagram of I-NAS is shown in Fig. 4.

3.1 *Image Data Preprocessing*

Take the Indian Pines dataset as an instance, Fig. 3 shows the progress of the preprocessing to the input HSI data. Suppose an HSI cube $X \in \mathbb{R}^{h \times w \times b}$ as the input, where h and w are the spatial sizes of X, and b indicates the number of spectral bands. It then is transformed to the index set I

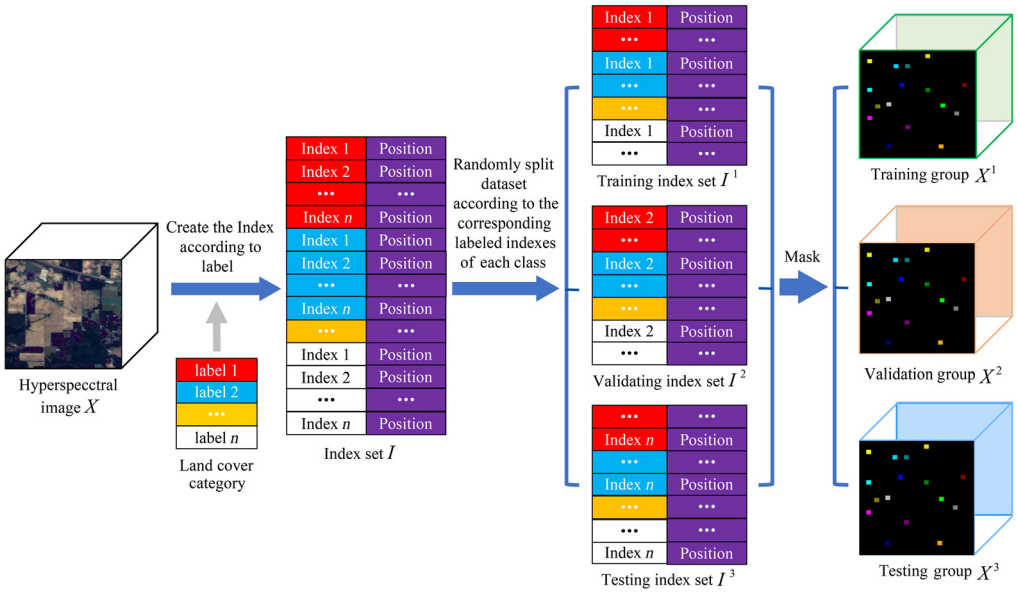

Fig. 3 HSI data preprocessing process, where X^1, X^2, and X^3 are the training, validation, and test datasets, respectively. I^1, I^2, and I^3 are the training, validation, and test index sets, respectively.

with its labels of each land cover category, which consists of both pixel indexes and corresponding position information. Each element of I can be described as (i, p), in which i is the index of each sample and p is the position information of the corresponding sample. Therefore, the dataset can be randomly divided into three groups of sets according to the corresponding labeled indexes of each class, including the training index set I^1, the validation index set I^2, and the test index set I^3.

To reconstruct to the raw input shape of the HSI cube X, we employ a masking operation to eliminate the interference of irrelevant pixels for each index set according to the position information of the selected pixels in the I^1, I^2, and I^3, respectively. And the image groups of X^1, X^2, and X^3 are generated within mutual independent spectral pixel sets. At this time, each group of datasets contains more sparse data distribution, which is convenient to identify the homogeneous spectral energy and efficient to extract the high correlation spatial contexts for classification.

3.2 *Image-Based NAS Framework*

The image-based NAS (I-NAS) framework for HSI classification is shown in Fig. 4. In which the whole structure can be divided into two steps: the architecture search phase and the model testing phase. Take an HSI cube X as the input, the training group X^1 and validation group X^2 that generated by the image data preprocessing are employed for training. In which, both X^1 and X^2 are fed into the optional architecture D in cell-based search space to extract discriminative features of HSI within various deep neural operations. Then it outputs the transformation matrix T that consists of the feature vector with probabilities of each land cover category for each pixel by the softmax classifier. The corresponding predicted labels are selected according to the positions of the training and test pixels. Then the predicted label vector $Y^* = [y_1^*, y_2^*, \ldots, y_S^*]$ is transformed during the select operation. In addition, each element of labeled pixel has its corresponding annotation $Y = [y_1, y_2, \ldots, y_S]$. Therefore, the parameters of D are updated by backpropagating the gradient of the cross-entropy objective function[41] in Eq. (1), which represents the difference between both Y^* and Y. The equation of the cross-entropy objective function is

$$C(y^*, y) = \sum_{i=1}^{S} y_i \left(\log \sum_{j=1}^{S} e^{y_j^*} - y_i^* \right), \tag{1}$$

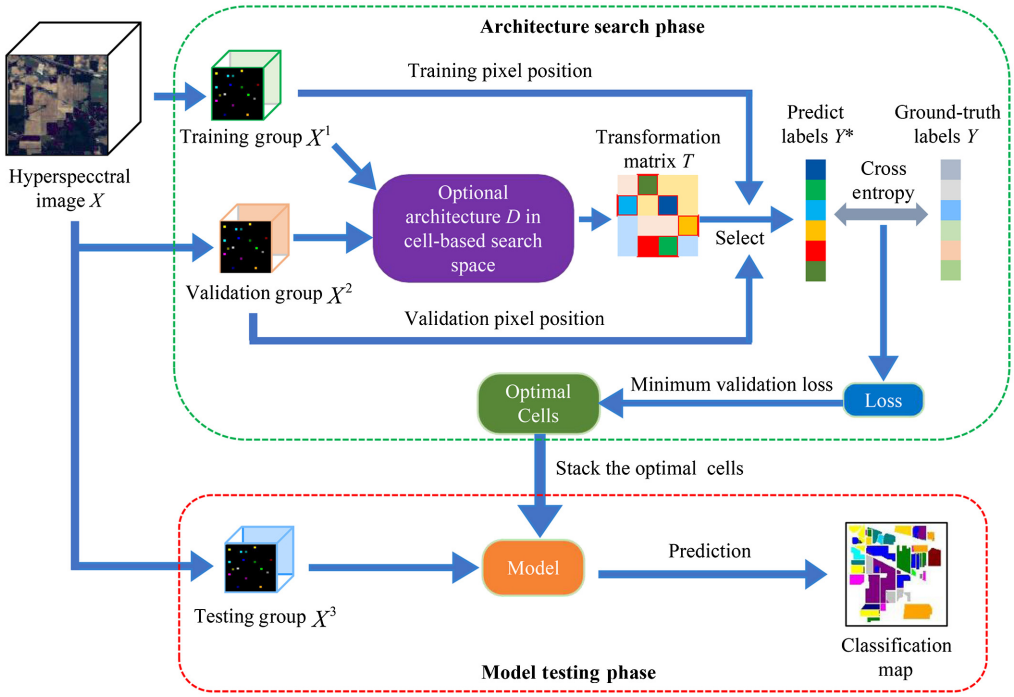

Fig. 4 I-NAS for the HSIC instances. The model can take an input of any size and produce an output with a corresponding size. In training, the predicted label is selected in the output according to the position of the training pixel. The cross-entropy represents the gradient of the cross-entropy objective function [Eq. (1)]. Architecture D is an optional architecture in the search space.

where S represents the number of land-cover categories, and $C(\cdot)$ represents the difference between the predicted label vector y^* and the ground reality label vector y. Finally, the optimal cells can be acquired from the search space.

The model testing phase aims at constructing the well-trained CNN classification model in a cellwise stack fashion with the above optimal cells, which is utilized for hyperspectral interpretation. In the model testing phase, the testing group X^3 containing the testing sample is used as the input of the final CNN classification model, and the corresponding labels of all pixels are predicted. The image-based classification framework can make full use of the graphics processing unit during testing to accelerate the reasoning process of nonredundant information. In addition, because there is no slicing operation, the testing process is straightforward. Thus, the result of the testing image can be directly output as an inference. More computing resources are conserved, and efficiency is improved.

3.2.1 *Cell structure search*

To figure out the optimal settings of the well-trained framework, the neural cells are implemented and evaluated during the architecture search phase in the search space. In which, we suppose a cell with an ordered sequence that consists of two input nodes, three intermediate nodes, and one output node. To demonstrate the cell search stream, we equipped the cell structure search space with three independent neural cells, which is shown in Fig. 5. It can be seen that each intermediate node with two operated sets O can be regarded as an element, which aims to determine the optimal operations collaboratively. Thus, we considered various neural operations to construct the operated sets and learn latent spectral–spatial distribution. Each operation of the set O is participated in the neural calculation to explore the variety of the spectral signatures and predict the characteristic parameters for performing the operation in the optimal cell.

At each operated set O in a cell, we define eight operations to parallel extract both available homogeneous area and neighboring contextual correlation of HSIs. They are separable convolutions with the kernel size 3×3 and 5×5, dilated convolutions[36] with the kernel size 3×3 and

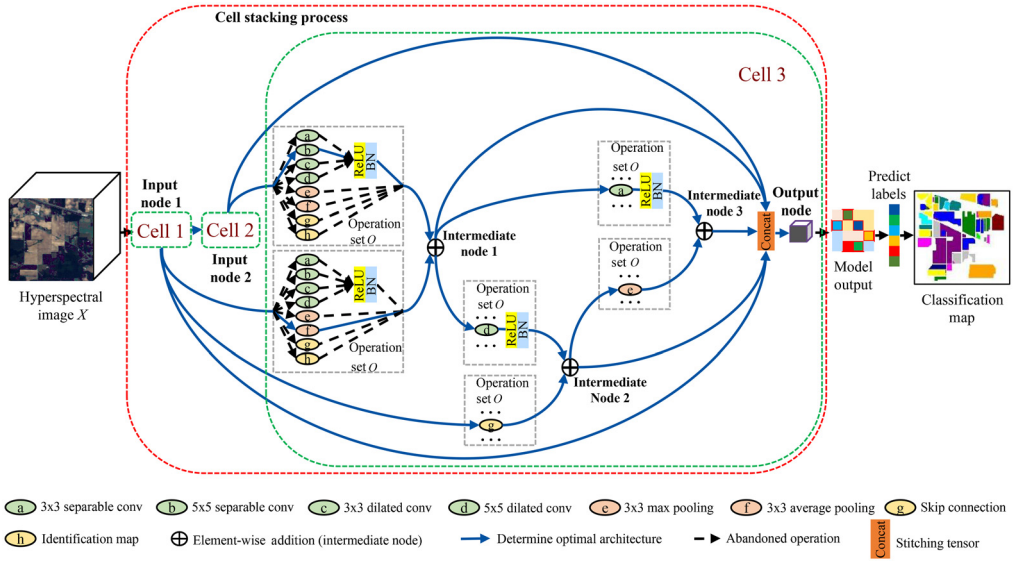

Fig. 5 CNN framework of cells stacking for HSI classification.

5×5, an average pooling operation, a max pooling operation, identification map, and the skip connection, respectively. Once each element of O has been operated and generated corresponding feature maps of HSIs, the most suitable intermediate nodes that construct the optimal cell can be determined by selecting high utility operated sets with corresponding feature maps of the highest weight parameters. Finally, the search structure predict the label of the input traing group X in a concatenation fashion and optimize the cell structure space via a gradient descent method.

In particular, a neural cell can be regarded as a directed acyclic graph that contains N sequential nodes. Each directed edge indicates a series of operations that transform the input node to the intermediate node. At this time, the potential representation of HSIs can be explored within various operations. If $P^{(m)}$ and $P^{(n)}$ represent the intermediate node and the input node of the cell, respectively, the structure of the neural cell can be formulated as

$$P^{(m)} = \sum_{m>n} o^{(n,m)}(P^{(n)}), \tag{2}$$

where $o^{(n,m)}$ is the operation between $P^{(m)}$ and $P^{(n)}$, such as convolution, pooling, and skip connections. In addition, (n, m) indicates the directed edge between $P^{(n)}$ and $P^{(m)}$ and is associated with the operation $o^{(n,m)}$ that transforms $P^{(n)}$.

To visualize the cell structure search strategy, Fig. 6 shows the search process distributed in a cell within one input node, two intermediate nodes, and an output node. In which we annotate independent digital labels for the various nodes, $P^{(n)}$ and $P^{(m)}$ represent the intermediate node 1 and intermediate node 2 of the cell, respectively. First, each node of the cell is connected with a dotted line, in which all operations of the directed edges are "unknown." Second, to initialize the parameters of the cell, we employed the operation set O to activate each node of the cell for the data generalization of HSIs. Then, it will eliminate the operations that contribute poor performance to the feature extraction during updating the parameters of weights and determine the optimal operations between both intermediate nodes $P^{(n)}$ and $P^{(m)}$ as shown in Fig. 6. It can be seen that the operation of the directed edge between $P^{(n)}$ and $P^{(m)}$ with bright color indicates the optimal operation within effective feature descriptions of HSI. The dark one representants the operations that have been abandoned during the optimization. Finally, the optimal cell structure is built by concatenating all of the operated nodes that contain optimal operations.

In addition, to reduce the loss of feature information and capture the complete characteristics for HSIs, we merge two input nodes of the cell as the output of the cell to avoid the inefficient description issues. The mathematical expression can be formulated as

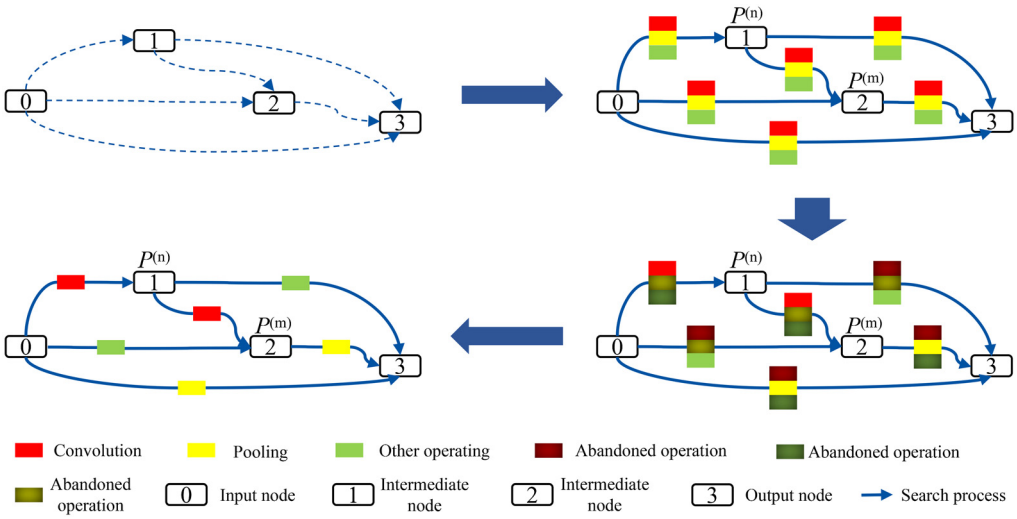

Fig. 6 Cell structure search process diagram.

$$h = \sum_{l} \text{concat}(P^{(l)}), \tag{3}$$

where $P^{(l)}$ represents all nodes in the cell except the output nodes, including the node $P^{(m)}$ and node $P^{(n)}$. The function of concat$(\cdot)$ is to concatenate two tensors and h is the output of the cell.

3.2.2 *Optimization*

To accelerate the search processing, we introduce the gradient descent algorithm to update the hyperparameters of the cell-based architecture search space. It can be reflected in the optimization and selection within all of the possible operations of O by the softmax classifier. In this way, the output of O can be regarded as the weighted sum for each result of various operations defined in O, such as convolution and pooling. If given the $f_o^{(n,m)}$ indicates the operation coefficient of i'th operation o between nodes $P^{(n)}$ and $P^{(m)}$ $(o \in O)$. Thus, the optimal o can be determined within the highest probability cofficient score. The optimized objective function of $f_o^{(n,m)}$ takes the form[36]

$$f_o^{(n,m)} = \frac{\exp(B_o^{(n,m)})}{\sum_{o' \in O} \exp(B_{o'}^{(n,m)})}, \tag{4}$$

$$o^{(n,m)}(P^{(n)}) = \sum_{o \in O} f_o^{(n,m)} o(P^{(n)}), \tag{5}$$

where O is the set of operations that contains all the operations shown in Fig. 5. The sum of the operation coefficients $f_o^{(n,m)}$ for each operation in the operation set O is 1, which is obtained by the softmax operation. $o(\cdot)$ is an operation function that represents the operation applied to the node $P^{(n)}$. $B^{(n,m)}$ is a vector of dimension $|O|$, and the mixture weights of operations on the directed edge (n, m) are parameterized by $B^{(n,m)}$. Then, the architecture search task is simplified by learning a set of continuous variables $B = \{B^{(n,m)}\}$. The gradient descent method is used to initialize and optimize $B_o^{(n,m)}$. Through the above methods, the search space is changed from being discrete to continuous. When the search is completed, according to $o^{(n,m)} = \arg\max_{o \in O} B_o^{(n,m)}$, the most probable operation is selected as the final operation, and then the cell structure is determined.

As with artificial neural network structures, the performance on the validation dataset is used to guide the structure design. The stochastic gradient descent (SGD) is used to accelerate the

search process and optimize the architectural variable B and network weight w in this paper. This is a two-level optimization process. The mathematical expression is as follows:

$$\min_B L_{\text{val}}(w^*(B), B), \tag{6}$$

$$\text{s.t. } w^*(B) = \arg\min_w L_{\text{train}}(w, B), \tag{7}$$

where L_{train} and L_{val} denote the training and validation loss, respectively. L_{train} and L_{val} depend on architectural variable B and network weight w. The goal of NAS is to find the variable B^* corresponding to the minimum validation loss $L_{\text{val}}(w^*, B^*)$ and the weight value w^* corresponding to the minimum training loss $w^* = \arg\min_w L_{\text{train}}(w, B^*)$. B^* and w^* are used for the architecture design.

For the cell search strategy, SGD or similar optimizer is introduced to find and evaluate the optimal architecture on the validation dataset. It can be applied as a predictive constraint item during validation after training at each epoch and can be formulated as

$$\theta_{L_{\text{val}}}{}' = \theta_{L_{\text{val}}} + (y^i - h_\theta(x^i))x^i_{L_{\text{val}}}, \tag{8}$$

where $\theta_{L_{\text{val}}}$ denotes the parameter weights and biases to be optimized, and x^i and y^i are the validation samples and their corresponding annotations. $h_\theta(\cdot)$ represents the cell operations for predicting the probability vectors containing the land cover categories. Finally, the gradients of $-\nabla_{\theta_{L_{\text{val}}}}$ are employed to update $\theta_{L_{\text{val}}}$ through SGD or similar optimization methods for learning the discriminative spectral–spatial distribution and constructing a well-trained CNN architecture for classification.

When the cell search process is finished, the output of each node is calculated based on only the two strongest previous nodes. Here, the variable $B_o^{(n,m)}$ defines the strength of the connection between two nodes. Then, the architecture of I-NAS is determined (as shown in Fig. 5). We train the I-NAS architecture from scratch on the training dataset. Algorithm 1 shows the whole process of I-NAS for HSIC.

4 Experimental Results and Analysis

This section describes the results of experiments conducted on two open HSI datasets, Indian Pines (IN) and University of Pavia (UP). The algorithms were evaluated on overall accuracy (OA), classification accuracy of each class (CA), average accuracy (AA), and kappa coefficient ($K \times 100$). All of the experiments were performed on a Tesla V100-SXM2 equipped with an NVIDIA SMI 418.67 32 GB GPU. The software environment consisted of Ubuntu18.04.3 as the operating system, CUDA10.1, the PyTorch deep learning acceleration library, and Python3.6 as the programming language. All results of the experiments were given as the average ± standard deviation over 10 independent runs.

4.1 *Experimental Dataset*

The IN dataset was collected by the AVIRIS sensor. The spatial resolution of the sensor is 20 m. IN has 224 spectral bands with wavelengths ranging from 400 to 2500 nm. The spatial size of the IN is 145×145. Because some bands cannot be reflected by water, the remaining 200 bands are generally studied. The dataset includes 16 land cover categories. The false-color image and the real image of the ground, the true value of the background, and the color code are shown in Fig. 7.

The UP dataset was collected using the reflective optics spectrographic imaging system. The UP data contain 115 spectral bands with wavelengths ranging from 430 to 860 nm, among which 12 bands are eliminated because of noise, so the image formed by the remaining 103 spectral bands is generally used. The spatial resolution was 1.3 m. The spatial size of the UP data is 610×340. The dataset includes nine land cover categories. Figure 8 shows the false-color image, ground true image, background truth value, and color code.

Algorithm 1 I-NAS for HSI classification.

Require: Initialize number of training group X^1, validation group X^2, and operation set O.

for each pixel **do**

  Create the index according to each class of label.

end for

split the sample set carrying position information into training, validation and test dataset according to X^1 and X^2.

Mask the training, validation and test dataset respectively.

Architecture search phase:

initialize search epochs, learning rate ξ and ε, the architecture variable B, and the CNN weight w.

for every search epoch **do**

  $w \leftarrow w - \xi \nabla_w L_{\text{train}}(w, B)$;

  $B \leftarrow B - \varepsilon \nabla_B L_{\text{val}}(w - \xi \nabla_w L_{\text{train}}(w, B), B)$.

end for

choose the best B^* according to the performance on validation dataset.

according to $o^{(n,m)} = \arg\max_{o \in O} B_o^{*(n,m)}$, acquire the best I-NAS architecture Ar.

Train and test the optimal I-NAS:

initialize training epochs, the weight w^* of Ar and learning rate ξ.

for every training epoch **do**

  $w^* \leftarrow w^* - \xi \nabla_w^* L_{\text{train}}(w^*)$.

end for

for every test epoch **do**

  according to test dataset carrying position information, acquire the predict by I-NAS.

end for

obtain OA, AA, Kappa by evaluating the predict and test labels.

(a) (b) (c)

Fig. 7 IN dataset. (a) False-color image. (b) Ground truth map. (c) Class name and color code and the corresponding number of classes.

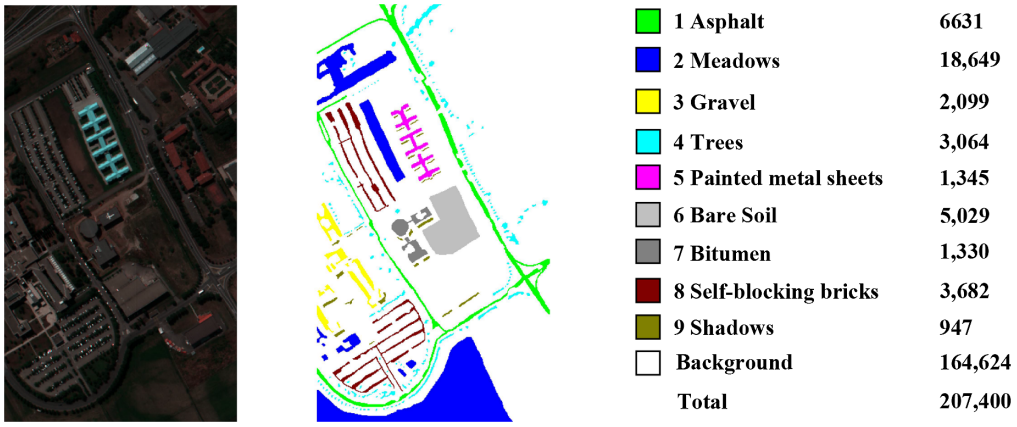

Fig. 8 UP dataset. (a) False-color image. (b) Ground truth map. (c) Class name and color code and the corresponding number of classes.

4.2 *Experimental Setup*

The epochs of the cell architecture search and final model training evaluation were 150 and 300, respectively. Considering the stability of model training, the learning rate of the CNN weight was initialized to 0.016 and 0.008, respectively, and the learning rate of the architecture variable was 0.0003. We used the Adam optimizer to optimize the loss function.[42] Because we selected the image containing samples as input for the architecture search, the batch size was set to 1, and the stride of each convolution kernel was set to 1 to avoid feature loss. In addition, to explore the diversity of the searched cells, we also searched for two types of cells, as shown in Figs. 9 and 10.

For dataset sampling, 50 samples were randomly selected in each land cover category for training, and if the number of labeled samples in a category was <50, half the samples of that category were randomly selected for training. Similarly, the validation dataset was identical to the training dataset extraction method. The only difference was that the number of randomly

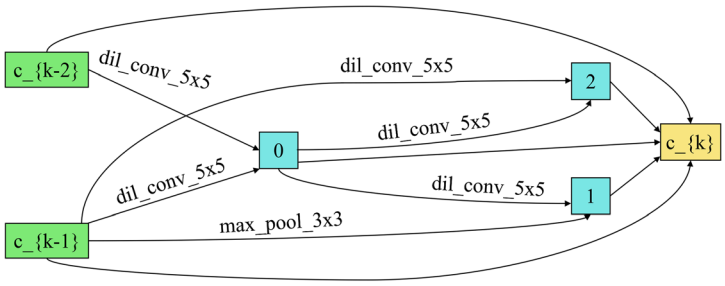

Fig. 9 The optimal structure of a cell based on the I-NAS method on IN.

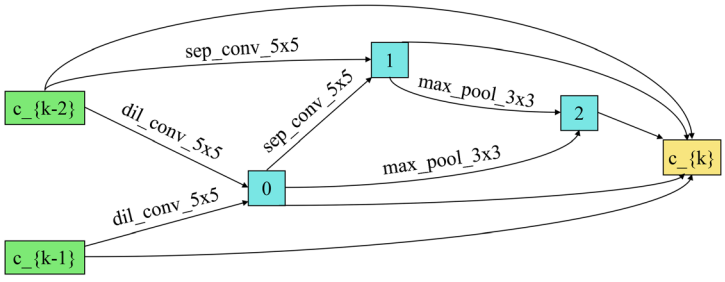

Fig. 10 Optimized structure of another cell based on the I-NAS method on IN.

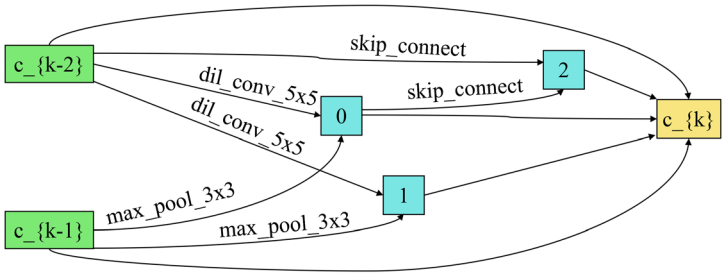

Fig. 11 The optimal structure of a cell based on I-NAS method on UP.

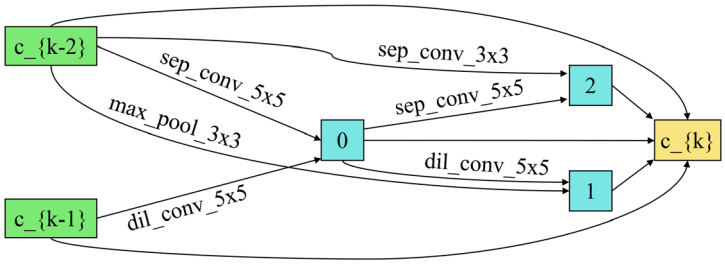

Fig. 12 Optimized structure of another cell based on I-NAS method on UP.

selected samples in each category was half that of the training dataset. All the remaining labeled samples are used as the testing datasets.

4.3 *Optimal Architecture*

We show the optimal cell structures in IN (see Figs. 9 and 10) and UP (see Figs. 11 and 12).

In the architecture search phase, the architecture variable B is selected according to the classification performance on the validation dataset, and thus the optimal architecture is obtained. Finally, I-NAS is trained from scratch and the classification accuracy of the network is evaluated on the test dataset. Here, we take the IN and UP as examples, showing the detailed structure of each cell from Figs. 9–12. As shown in the figures, the input of the cell is the output of the two previous cells $c\_\{k-2\}$ and $c\_\{k-1\}$, which has three intermediate nodes 0, 1, and 2, and the output of the cell $c\_\{k\}$ is a splice of two input nodes and three intermediate nodes.

4.4 *Classification Results on Hyperspectral Datasets*

To evaluate the effectiveness of the I-NAS method, we used five representative HSI classification methods for qualitative comparison, including the support vector machine (SVM), SSRN,[27] ADGAN,[32] 3DCNN,[43] and CapsNet.[29] In addition, to evaluate the feature characterization capability and computational efficiency of the proposed method in this paper, we also compared P-NAS to explore the difference between the neighborhood cube input and the image-based input. In Tables 1 and 2, the best experimental results are in bold.

The purpose of using the IN dataset is to verify the robustness of the algorithm when dealing with unbalanced samples. For the IN dataset, as shown in Table 1, although the SVM has good robustness in HSIC, training the classifier using only spectral features limits the degree of feature generalization of the SVM, resulting in the worst classification accuracy. For example, the fourth and ninth classes reached 44.02% and 50.11%, respectively. Compared with SVM, 3DCNN exhibits promising classification performance. In addition, SSRN, ADGAN and CapsNet all possess different degrees of classification performance due to the introduction of different degrees of spatial information. Among them, CapsNet can achieve results that compete with I-NAS and P-NAS due to the use of pixel spatial location information. In addition, the I-NAS proposed in this paper possesses a more advanced classification performance, reaching

Table 1 Classification results of different methods for IN dataset.

Class	SVM	3DCNN	SSRN	ADGAN	CapsNet	P-NAS	I-NAS
1	67.10 ± 1.67	89.80 ± 1.23	71.08 ± 2.48	98.04 ± 2.99	68.50 ± 9.25	**100 ± 0.00**	**100 ± 0.00**
2	70.11 ± 2.58	90.00 ± 2.22	94.67 ± 4.81	97.72 ± 2.18	93.94 ± 2.78	86.52 ± 3.59	89.76 ± 4.42
3	63.09 ± 5.81	86.82 ± 1.71	88.17 ± 6.18	95.55 ± 3.57	88.77 ± 5.11	96.97 ± 1.58	96.85 ± 1.81
4	44.02 ± 4.27	94.18 ± 1.93	99.13 ± 9.49	96.02 ± 2.67	71.03 ± 9.33	99.87 ± 0.25	98.82 ± 2.07
5	85.97 ± 2.37	92.20 ± 1.06	92.61 ± 7.25	96.43 ± 3.13	87.05 ± 3.29	96.41 ± 2.35	95.78 ± 1.72
6	92.25 ± 1.35	99.50 ± 1.32	97.03 ± 5.56	97.76 ± 2.89	98.83 ± 1.26	98.47 ± 1.01	99.32 ± 0.55
7	81.81 ± 1.66	82.75 ± 2.30	50.68 ± 4.39	69.23 ± 9.65	72.64 ± 9.61	**100 ± 0.00**	**100 ± 0.00**
8	98.47 ± 1.03	99.61 ± 0.97	98.55 ± 4.10	99.54 ± 0.78	99.85 ± 0.42	**100 ± 0.00**	99.95 ± 0.15
9	50.11 ± 5.74	99.20 ± 0.97	68.49 ± 3.70	71.78 ± 9.71	56.66 ± 9.62	**100 ± 0.00**	**100 ± 0.00**
10	68.38 ± 1.97	87.71 ± 5.12	94.58 ± 1.05	95.73 ± 1.64	94.71 ± 3.01	92.57 ± 1.11	96.19 ± 1.62
11	82.22 ± 2.59	91.19 ± 0.92	89.55 ± 5.26	96.90 ± 1.84	97.25 ± 1.34	91.83 ± 3.20	90.49 ± 4.09
12	68.26 ± 2.29	93.10 ± 1.15	74.24 ± 9.72	97.14 ± 2.33	87.01 ± 5.15	94.24 ± 1.55	95.32 ± 1.97
13	94.03 ± 2.92	99.58 ± 1.32	99.69 ± 9.71	96.93 ± 5.99	96.94 ± 4.58	**100 ± 0.00**	99.84 ± 0.30
14	94.95 ± 0.81	96.22 ± 0.27	98.02 ± 7.36	99.76 ± 0.31	98.25 ± 1.58	99.42 ± 0.34	98.24 ± 2.32
15	57.62 ± 5.29	75.21 ± 2.11	91.75 ± 9.40	97.66 ± 1.95	88.12 ± 8.86	99.74 ± 0.52	99.12 ± 1.07
16	83.68 ± 6.22	99.00 ± 2.67	94.36 ± 9.39	84.02 ± 8.05	91.13 ± 7.57	97.77 ± 2.72	**100 ± 0.00**
OA (%)	76.83 ± 1.19	91.68 ± 1.44	90.74 ± 3.49	87.97 ± 6.56	94.39 ± 0.63	94.25 ± 0.39	**94.64 ± 1.23**
AA (%)	75.08 ± 1.49	92.31 ± 0.57	87.66 ± 5.68	93.19 ± 2.72	85.91 ± 2.97	97.11 ± 0.32	**97.49 ± 0.47**
$K \times 100$	73.69 ± 1.31	89.40 ± 0.95	89.39 ± 4.02	**96.76 ± 0.92**	93.47 ± 0.74	93.39 ± 0.45	93.83 ± 1.41

Table 2 Classification results of different methods for UP dataset.

Class	SVM	3DCNN	SSRN	ADGAN	CapsNet	P-NAS	I-NAS
1	95.38 ± 1.37	86.20 ± 3.23	88.88 ± 7.36	89.59 ± 8.60	95.80 ± 0.81	95.96 ± 3.10	96.66 ± 1.88
2	94.60 ± 0.88	93.11 ± 2.11	92.21 ± 1.73	91.44 ± 9.65	99.83 ± 0.24	95.91 ± 1.68	96.86 ± 1.46
3	65.34 ± 4.27	63.09 ± 6.54	92.79 ± 5.99	95.95 ± 3.47	68.74 ± 7.59	99.22 ± 0.73	98.28 ± 1.54
4	77.73 ± 7.36	95.81 ± 1.77	96.19 ± 0.82	88.65 ± 6.25	96.75 ± 1.49	94.41 ± 2.06	97.44 ± 0.54
5	94.57 ± 2.83	94.14 ± 4.78	97.76 ± 3.05	97.73 ± 2.29	99.90 ± 0.18	99.49 ± 0.43	**100 ± 0.00**
6	67.55 ± 4.42	93.06 ± 1.99	97.03 ± 0.36	91.07 ± 9.49	99.49 ± 0.61	99.32 ± 0.55	97.34 ± 1.09
7	61.03 ± 6.42	57.81 ± 5.44	68.23 ± 9.55	93.56 ± 4.87	77.27 ± 4.73	99.97 ± 0.07	99.91 ± 0.13
8	78.50 ± 5.96	76.10 ± 2.86	90.16 ± 3.77	72.77 ± 9.66	87.03 ± 3.23	97.45 ± 1.10	99.25 ± 0.40
9	99.89 ± 0.06	83.20 ± 4.92	**100 ± 0.00**	73.05 ± 9.67	95.88 ± 4.11	95.02 ± 3.69	99.92 ± 0.05
OA (%)	84.45 ± 1.88	87.32 ± 1.89	91.20 ± 1.58	81.25 ± 9.33	95.44 ± 0.81	96.71 ± 0.71	**97.45 ± 0.75**
AA (%)	81.62 ± 1.98	82.51 ± 2.17	91.58 ± 1.60	88.20 ± 6.06	91.31 ± 1.68	97.42 ± 0.83	**98.41 ± 0.36**
$K \times 100$	79.87 ± 2.25	83.56 ± 2.71	88.17 ± 2.17	87.53 ± 9.14	94.16 ± 1.04	95.65 ± 0.93	**96.62 ± 0.98**

94.64% on OA. Hwever, the OA difference between P-NAS and I-NAS is not significant, which may be due to the similar spectral information of neighboring pixels in homogeneous regions of P-NAS. The latter does not have the same advantages as the neighboring spectral information, but it has a larger spatial receptive field and can extract abundant spatial texture information. Compared with the SVM, the OA, AA, and K of I-NAS are improved by 17.81%, 22.41%, and 20.14%, for 3DCNN by 3.15%, 5.18%, and 4.43%, for SSRN by 4.57%, 13.29%, and 5.19%, and for CapsNet by 0.25%, 11.58%, and 0.36%, respectively. The OA of I-NAS is 6.67% higher than that of ADGAN and 0.39% higher than that of P-NAS. In I-NAS, the classification accuracy of the first, seventh, ninth, and sixteenth categories reached 100%. This may be due to the small number of samples in these classes and the proportion of training samples in the homogeneous classes was larger than that in other classes. This confirms that I-NAS can achieve a better classification effect when dealing with hyperspectral data with unbalanced labeled samples, which indicates that I-NAS is a powerful HSIC framework.

The UP dataset is used to examine the effectiveness of the algorithm in processing high-resolution data samples. As shown in Table 2, due to the finer spectral features of the UP dataset, the pixel-based approach (SVM) can provide relatively good experimental results on UP, especially for the ninth category. In addition, I-NAS and P-NAS possess more advanced classification performance compared with 3DCNN, SSRN, ADGAN, and CapsNet, and they reach 97.45% and 96.71% on OA. From Table 2, the OA, AA, and K of I-NAS are 13%, 16.79%, and 16.75% higher than the SVM, 10.13%, 15.90%, and 13.06% higher than 3DCNN, 6.25%, 6.83%, and 8.45% higher than SSRN, 16.20%, 10.21%, and 9.09% higher than ADGAN, 2.01%, 7.10%, and 2.46% higher than CapsNet, 0.74%, 0.99%, and 0.97% higher than the P-NAS. The validity of the algorithm was verified. For the UP dataset, the classification accuracy of the five land cover categories in I-NAS is higher than that of other classification algorithms, among which the classification accuracy of the fifth, seventh, eighth, and ninth categories reached more than 99%. This indicates that I-NAS can achieve high classification accuracy when dealing with high-resolution data samples.

Figures 13 and 14 show the classification maps for various classifiers using the IN and UP datasets. It is evident from the resulting images that SVM using spectral features alone always produces noisy scatter and depicts more errors than the spectral–spatial approach [see Figs. 13(b), 13(c), 14(b), and 14(c)], as there are more spatially homogeneous regions in the IN dataset. The lower the spatial resolution, the higher the probability of mixed pixels and the

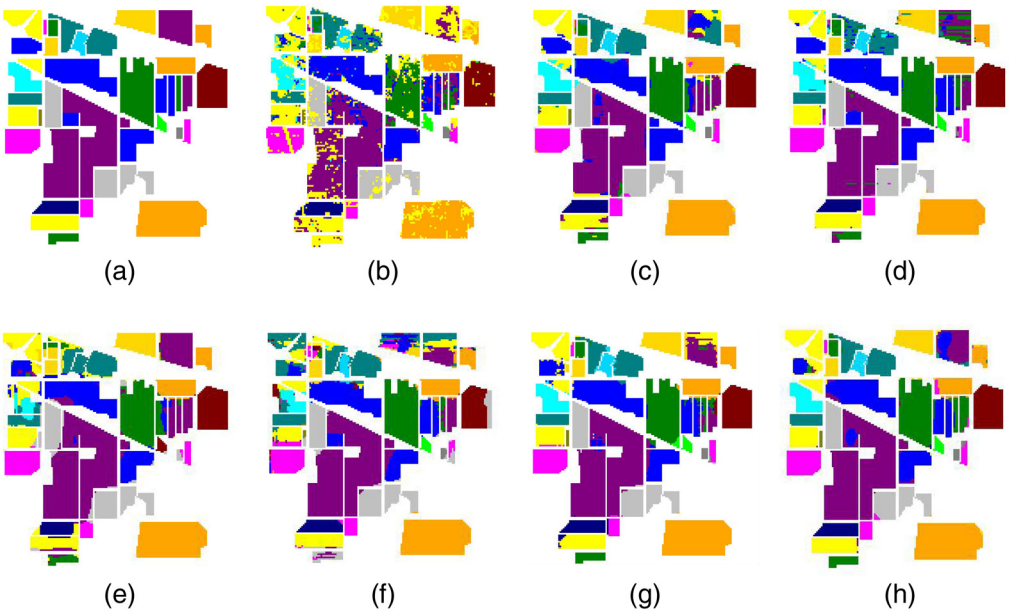

Fig. 13 Classification maps obtained by all considered algorithms on IN dataset. (a) Ground truth, (b) SVM, (c) 3DCNN, (d) SSRN, (e) ADGAN, (f) CapsNet, (g) P-NAS, and (h) I-NAS.

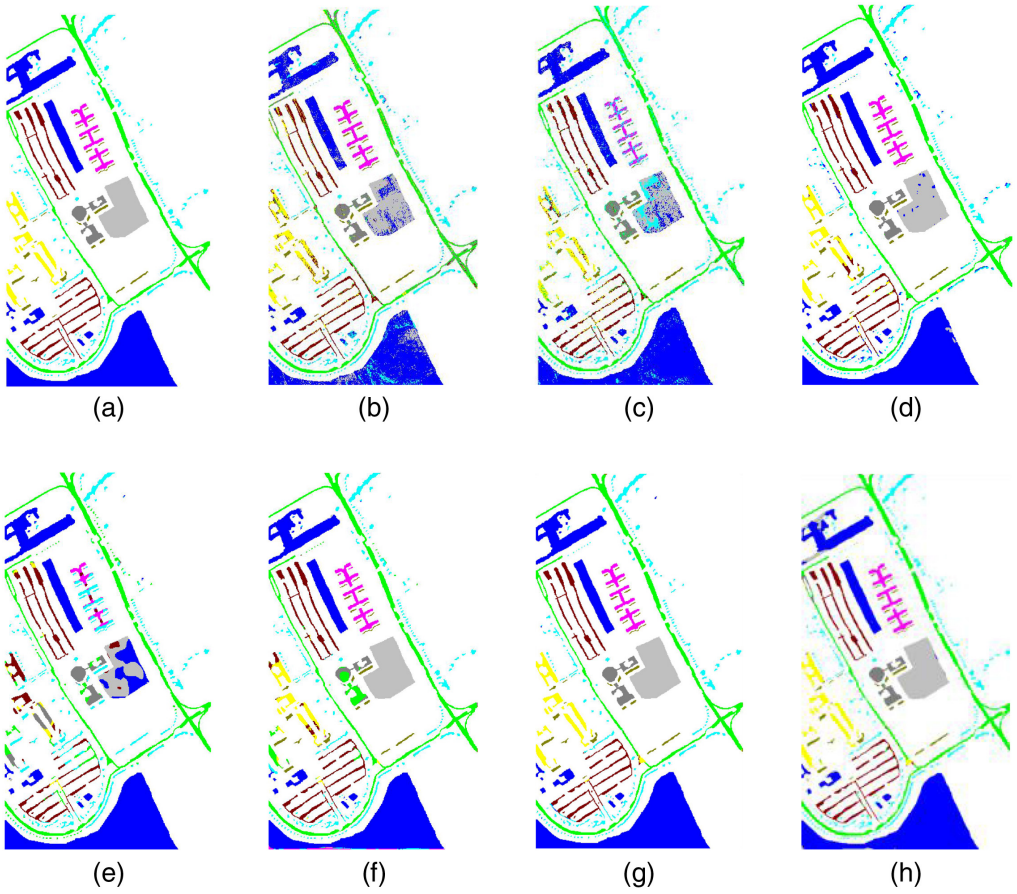

Fig. 14 Classification maps obtained by all considered algorithms on UP dataset. (a) Ground truth, (b) SVM, (c) 3DCNN, (d) SSRN, (e) ADGAN, (f) CapsNet, (g) P-NAS, and (h) I-NAS.

greater the difficulty of classification. Due to the spectral similarities of categories 2, 10, and 11, misclassification easily occurs. However, I-NAS has achieved good results on small samples, such as categories 1, 4, 7, 9, and 16 [see Figs. 13(c)–13(h)]. By comparing the real ground reference with the classification map, I-NAS is shown to obtain more accurate classification results.

Similarly, due to the higher spatial resolution of the UP dataset, the SVM has better classification performance than on the IN dataset. Compared with the spatial–spectral-based algorithms, P-NAS and I-NAS show better classification maps [see Figs. 14(c)–14(h)]. Compared with P-NAS, I-NAS uses the whole image information as the model input to generate a smoother classification map [see Figs. 14(g) and 14(h)]. Because the numbers of samples in categories 2 and 4 are relatively large and the number of training samples is small, it is easily affected by noise and make classification errors [see Figs. 14(c)–14(f)]. Due to the spectral similarity of categories 3 and 8, the classification errors easily occur [see Figs. 14(c)–14(f)]. However, I-NAS achieved good results in these categories.

4.5 *Investigation of Number of Training Samples*

We evaluated the sensitivity of the model to the number of samples by selecting different numbers of training samples separately and investigated the effect of different numbers of training samples on OA. Figure 15(a) shows the OA of all classification methods using different numbers of training samples on the IN dataset, indicating that an increase in the number of training samples has a decisive effect on the classification performance of the seven classification methods. Although the classification accuracy of the SVM can be improved by selecting the best training sample to improve the generalization performance,[44] it was observed in the experiment that the

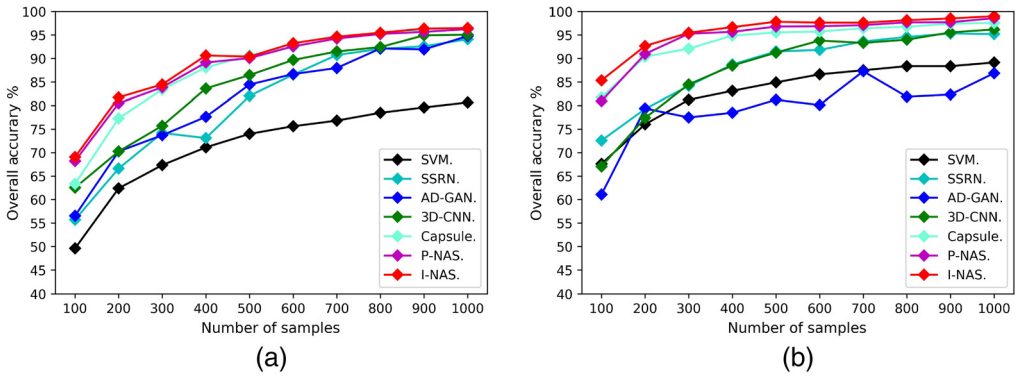

Fig. 15 Impact of different number of training samples on OA results for training. OA results were obtained by all algorithms on (a) IN dataset and (b) UP dataset.

SVM had the worst classification accuracy on all the above-mentioned training samples. Second, compared with the SSRN, ADGAN, CapsNet, and P-NAS methods, the classifier based on I-NAS provides better performance. Finally, the OA of the I-NAS tended to increase as the number of samples increases, and high classification accuracy can be obtained on different numbers of training samples.

Figure 15(b) shows the OA of all classification methods using different numbers of training samples on the UP dataset. Although the performance relationships among the SVM, 3DCNN, SSRN, ADGAN, CapsNet, and P-NAS are different from the above experiments, I-NAS provides the best results in most cases, and the OA manifests an upward trend. The I-NAS can achieve high classification accuracy on different numbers of training samples.

To validate the contribution of different classes to OA by modifying the training samples, a new experiment was conducted on the IN and UP datasets. Figure 16(a) shows the CA for each class obtained by the I-NAS algorithm with different numbers of training samples on the IN dataset. It can be seen that the CA values of I-NAS are stable for classes 1, 7, 9, 13, and 16, no matter how many training samples are considered, which is due to the relatively small number of samples in these five classes and the intraclass variation in land coverage can be negligible. Thus, although the number of training samples is relatively small, beneficial classification performance can be obtained. For the remaining classes, when the number of samples increases, their contribution to CA gradually increases and then stabilizes. Similar evidence is shown in Fig. 16(b), which shows the CA of each class on different numbers of training samples obtained from the I-NAS algorithms on the UP dataset. It is shown in Fig. 16(b) that for classes 5, 7, and 9, the variability of CA values is negligible as the number of training samples increases. For the other classes, the CA values gradually stabilize as the number of samples increases. In summary, by modifying the training samples, not all classes contribute equally to OA. The reason may be that the spectral signatures confront the challenge of spectral variability due to

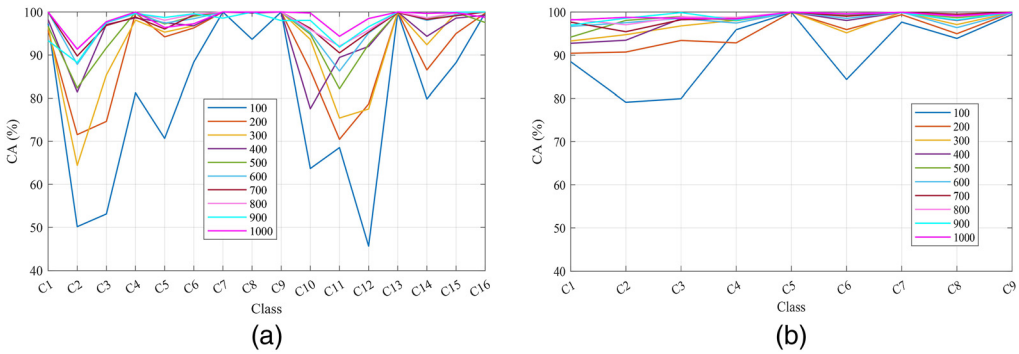

Fig. 16 CA results for each class with different number of total labeled samples for training over the I-NAS on (a) IN dataset and (b) UP dataset.

illumination and atmospheric conditions, which means that several classes have high intraclass difference, and these classes may need more training samples to characterize the class features, so those classes that are not disturbed by spectral variability only need small training samples.

4.6 *Time Complexity and Space Complexity Experiments*

The time and space complexity of experiments is always an important point in the study of HSIC. This section discusses the training and testing times required by different algorithms and studies the number of training parameters for different algorithms. The time consumption and the number of parameters in the architecture search phase of P-NAS and I-NAS algorithms were studied.

Tables 3 and 4 show the time consumption of all algorithms for training and testing. For the IN dataset, compared with the SVM, other deep learning methods needed more time to train the network. I-NAS is faster than other deep learning methods because its input is complete image rather than the neighboring cube, so it reduces data redundancy and computational burden. Of all the methods compared, SSRN and 3DCNN are the most time-consuming. For the former, more training time is required due to the many network layers and needing to extract spectral and spatial features. For the latter, a large number of parameters in the 3D convolution kernel lead to extensive training time. Because the CapsNet considers the spatial location information of pixels, it takes a relatively long time. The P-NAS method requires more training time than CapsNet because P-NAS has more network layers and concat the information of each node in each cell, resulting in more redundant information. In addition to I-NAS, ADGAN requires least training time, because ADGAN is a semisupervised classification method, which has less label information than other methods. Somewhat similar results were obtained on the UP dataset.

Table 3 Time consumption of different algorithms on the IN dataset.

Method	Training time (s)	Testing time (s)
SVM	0.15 ± 0.01	3.18 ± 0.08
DCNN	1502.29 ± 6.03	57.68 ± 3.98
SSRN	3368.62 ± 51.05	47.58 ± 5.97
ADGAN	752.49 ± 4.92	3.24 ± 0.10
CapsNet	1293.09 ± 43.26	14.77 ± 0.96
P-NAS	1670.81 ± 7.94	102.53 ± 2.34
I-NAS	153.74 ± 4.38	0.33 ± 0.04

Table 4 Time consumption of different algorithms on the UP dataset.

Method	Training time (s)	Testing time (s)
SVM	0.09 ± 0.01	5.96 ± 0.48
DCNN	458.29 ± 3.11	246.78 ± 57.58
SSRN	1358.58 ± 70.69	82.25 ± 1.39
ADGAN	1227.62 ± 4.77	14.61 ± 0.05
CapsNet	871.47 ± 38.74	78.99 ± 1.57
P-NAS	1145.18 ± 15.68	319.97 ± 2.79
I-NAS	427.60 ± 6.68	1.20 ± 0.18

Table 5 Total number of trainable parameters in different algorithms (MB).

	Datasets	
Method	Indian Pines	Pavia University
SSRN	0.330108	0.189316
CapsNet	0.262329	0.174164
P-NAS	0.067012	0.073683
I-NAS	0.082724	0.089795

Similar results as above were obtained regarding testing time. I-NAS requires less time than the other classification algorithms for reasons mentioned above, i.e., lower data redundancy and computational burden. We obtained similar results on the UP dataset.

In this study, the number of trainable parameters (MB) was used to discuss the extent to which different algorithms occupy computer resources. The detailed results are listed in Table 5. Because the CNN architecture based on an image is searched for automatically, the optimal architecture found by each search could be different (the experiment in this study was run 10 times). We chose an optimal structure and calculated the number of trainable parameters.

As shown in Table 5, the number of trainable parameters of I-NAS is significantly less than that of other methods (such as SSRN, CapsNet, and P-NAS) for the IN dataset. The number of trainable parameters of I-NAS in UP was greater than that for the IN dataset. The reason may be that I-NAS takes entire image as input, while the spatial size of UP is larger than that of IN. In addition, I-NAS can achieve better classification performance with fewer trainable parameters for two reasons. First, the limited number of training samples and the considerable number of trainable parameters easily overfits the model. Second, I-NAS, which has a strong generalization ability, is an automatically designed architecture that depends on the training and validation samples.

For NAS, an architecture search is a very time-consuming step. Tables 6 and 7 show the time consumption and number of parameters of P-NAS and I-NAS in the architecture search phase on the two datasets, respectively. As shown in Tables 6 and 7, the time consumption and the number of required parameters in the architecture search phase of I-NAS are significantly lower than those of P-NAS.

Table 6 Time consumption and number of parameters for P-NAS and I-NAS on the IN dataset within the architecture search phase.

Method	Architecture search time (s)	Number of parameters (MB)
P-NAS	3090.35	0.288820
I-NAS	453.85	0.188084

Table 7 Time consumption and number of parameters for P-NAS and I-NAS on the UP dataset within the architecture search phase.

Method	Architecture search time (s)	Number of parameters (MB)
P-NAS	2453.18	0.286947
I-NAS	1262.30	0.185987

5 Conclusions

This study proposes a new image-based NAS method. Compared with the 3DCNN and SSRN designed by human experts, I-NAS has better classification accuracy. I-NAS uses a masking operation to eliminate the interference of irrelevant pixels in each index set according to the position information of selected pixels in I^1, I^2, and I^3. The image groups of X^1, X^2, and X^3 are generated within mutual independent spectral pixel sets. Then, image groups X^1 and X^2 are used as the input for architecture search and determines the optimal cell structure in the search space by the gradient-based search strategy. To improve the classification performance, an end-to-end connection method was adopted in the cell to merge the input nodes into the output node to reduce the data loss caused by convolution and pooling.

This paper discusses the temporal and spatial complexity of the experiments, in which I-NAS spent less time (∼7 min) during the architecture search process on the IN dataset than P-NAS (∼51 min). In addition, the CNN architecture obtained by the I-NAS method can automatically matched specific datasets and has good generalization ability. However, I-NAS also has many limitations. I-NAS used the entire image as a network input, which inevitably resulted in data loss. In addition, I-NAS consumed considerable memory because the input was an entire image and its size directly affected the memory of the graphics card being used. In future work, the image will be further processed to reduce the loss of feature information and the burden on the graphics card memory and more efficient search algorithms will be found to improve further the classification performance.

Acknowledgments

This work was supported by the Natural Science Foundation of Ningxia Province of China (Project No. 2021AAC03179) and the Natural Science Foundation of Ningxia Province of China (Project No. 2020AAC02028). The authors would like to thank the Key Laboratory of Images & Graphics Intelligent Processing of State Ethnic Affairs Commission of China for their support. The authors declare no conflict of interest.

References

1. G. Camps-Valls et al., "Advances in hyperspectral image classification: Earth monitoring with statistical learning methods," *IEEE Signal Process. Mag.* **31**(1), 45–54 (2013).
2. A. Vali, S. Comai, and M. Matteucci, "Deep learning for land use and land cover classification based on hyperspectral and multispectral earth observation data: a review," *Remote Sens.* **12**(15), 2495 (2020).
3. R. R. Pullanagari et al., "Assessing the performance of multiple spectral–spatial features of a hyperspectral image for classification of urban land cover classes using support vector machines and artificial neural network," *J. Appl. Remote Sens.* **11**(2), 026009 (2017).
4. C. Hu, M. Xv, and Y. Fan, "Hyperspectral image classification method combining fast guided filtering and spatial neighborhood information," in *Int. Conf. Comput. Commun. and Network Secur. (CCNS)*, IEEE, pp. 55–58 (2020).
5. X. Shen, W. Bao, and K. Qu, "Subspace-based preprocessing module for fast hyperspectral endmember selection," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **14**, 3386–3402 (2021).
6. M. Kycko et al., "*In situ* hyperspectral remote sensing for monitoring of alpine trampled and recultivated species," *Remote Sens.* **11**(11), 1296 (2019).
7. Y. Xu et al., "Advanced multi-sensor optical remote sensing for urban land use and land cover classification: outcome of the 2018 IEEE GRSS data fusion contest," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **12**(6), 1709–1724 (2019).
8. R. Hänsch and O. Hellwich, "Fusion of multispectral lidar, hyperspectral, and RGB data for urban land cover classification," *IEEE Geosci. Remote Sens. Lett.* **18**, 366–370 (2020).
9. C. Li et al., "Hyperspectral image classification with robust sparse representation," *IEEE Geosci. Remote Sens. Lett.* **13**(5), 641–645 (2016).

10. C. Yu et al., "Class signature-constrained background-suppressed approach to band selection for classification of hyperspectral images," *IEEE Trans. Geosci. Remote Sens.* **57**(1), 14–31 (2018).
11. H. Yu et al., "Locality sensitive discriminant analysis for group sparse representation-based hyperspectral imagery classification," *IEEE Geosci. Remote Sens. Lett.* **14**(8), 1358–1362 (2017).
12. K. Zhang, W. Zuo, and L. Zhang, "FFDNet: toward a fast and flexible solution for CNN-based image denoising," *IEEE Trans. Image Process.* **27**(9), 4608–4622 (2018).
13. H. Huang and K. Xu, "Combing triple-part features of convolutional neural networks for scene classification in remote sensing," *Remote Sens.* **11**(14), 1687 (2019).
14. H. Huang et al., "Local linear spatial–spectral probabilistic distribution for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.* **58**(2), 1259–1272 (2019).
15. S. K. Roy et al., "Morphological convolutional neural networks for hyperspectral image classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **14**, 8689–8702 (2021).
16. W. Hu et al., "Deep convolutional neural networks for hyperspectral image classification," *J. Sens.* **2015**, 258619 (2015).
17. C. Yu et al., "Hyperspectral image classification method based on CNN architecture embedding with hashing semantic feature," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **12**(6), 1866–1881 (2019).
18. S. Li et al., "Fusing hyperspectral and multispectral images via coupled sparse tensor factorization," *IEEE Trans. Image Process.* **27**(8), 4118–4130 (2018).
19. J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, "Classification of hyperspectral data from urban areas based on extended morphological profiles," *IEEE Trans. Geosci. Remote Sens.* **43**(3), 480–491 (2005).
20. M. Dalla Mura et al., "Morphological attribute profiles for the analysis of very high resolution images," *IEEE Trans. Geosci. Remote Sens.* **48**(10), 3747–3762 (2010).
21. X. Kang, S. Li, and J. A. Benediktsson, "Spectral–spatial hyperspectral image classification with edge-preserving filtering," *IEEE Trans. Geosci. Remote Sens.* **52**(5), 2666–2677 (2013).
22. Y. LeCun et al., "Gradient-based learning applied to document recognition," *Proc. IEEE* **86**(11), 2278–2324 (1998).
23. W. Li et al., "Hyperspectral image classification using deep pixel-pair features," *IEEE Trans. Geosci. Remote Sens.* **55**(2), 844–853 (2016).
24. W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: a dimension reduction and deep learning approach," *IEEE Trans. Geosci. Remote Sens.* **54**(8), 4544–4554 (2016).
25. S. Mei et al., "Unsupervised spatial–spectral feature learning by 3D convolutional autoencoder for hyperspectral classification," *IEEE Trans. Geosci. Remote Sens.* **57**(9), 6808–6820 (2019).
26. S. K. Roy et al., "HybridSN: exploring 3-D–2-D CNN feature hierarchy for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.* **17**(2), 277–281 (2019).
27. Z. Zhong et al., "Spectral–spatial residual network for hyperspectral image classification: a 3-D deep learning framework," *IEEE Trans. Geosci. Remote Sens.* **56**(2), 847–858 (2017).
28. W. Wang et al., "A fast dense spectral–spatial convolution network framework for hyperspectral images classification," *Remote Sens.* **10**(7), 1068 (2018).
29. K. Zhu et al., "Deep convolutional capsule network for hyperspectral image spectral and spectral-spatial classification," *Remote Sens.* **11**(3), 223 (2019).
30. I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM* **63**(11), 139–144 (2020).
31. H. Liang, W. Bao, and X. Shen, "Adaptive weighting feature fusion approach based on generative adversarial network for hyperspectral image classification," *Remote Sens.* **13**(2), 198 (2021).
32. J. Wang et al., "Adaptive dropblock-enhanced generative adversarial networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.* **59**(6), 5040–5053 (2020).
33. Y. Chen et al., "Automatic design of convolutional neural network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.* **57**(9), 7048–7066 (2019).

34. C. Zhang et al., "Particle swarm optimization based deep learning architecture search for hyperspectral image classification," in *IGARSS 2020-2020 IEEE Int. Geosci. Remote Sens. Symp.*, IEEE, pp. 509–512 (2020).
35. X. Cui et al., "Multiscale spatial-spectral convolutional network with image-based framework for hyperspectral imagery classification," *Remote Sens.* **11**(19), 2220 (2019).
36. H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *Int. Conf. Learn. Represent.* (2019).
37. B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Int. Conf. Learn. Represent.* (2017).
38. B. Zoph et al., "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 8697–8710 (2018).
39. E. Real et al., "Regularized evolution for image classifier architecture search," *Proc. AAAI Conf. Artif. Intell.* **33**(01), 4780–4789 (2019).
40. X. Liu et al., "Continuous particle swarm optimization-based deep learning architecture search for hyperspectral image classification," *Remote Sens.* **13**(6), 1082 (2021).
41. Z. I. Botev et al., "The cross-entropy method for optimization," in *Handbook of Statistics*, Vol. **31**, pp. 35–59, Elsevier (2013).
42. D. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (iclr'15)*, San Diego (2015).
43. A. B. Hamida et al., "3-D deep learning approach for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.* **56**(8), 4420–4434 (2018).
44. M. Ahmad et al., "Spatial prior fuzziness pool-based interactive classification of hyperspectral images," *Remote Sens.* **11**(9), 1136 (2019).

Zhonggang Hu received his BEng degree in network engineering from Maths and Information Technology School of Yuncheng University, Shanxi, China, 2019. He is currently pursuing his MS degree at the School of Computer Science and Engineering, North Minzu University, Yinchuan, China. His research interests include hyperspectral image processing and deep learning.

Wenxing Bao received his BEng degree in industrial automation from Xidian University, Xi'an, China, in 1993, his MSc degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2001, and his PhD in electronic science and technology from Xi'an Jiaotong University, Xi'an, China, in 2006. He is currently a professor. His current research interests include digital image processing, remote sensing image classification, and fusion.

Kewen Qu received his BEng degree in computer science from Henan University, Kaifeng, China, in 2008, his MSc degree in computer science from North Minzu University, Yinchuan, China, in 2011, and his PhD degree in computer science and technology from Hefei University of Technology, Hefei, China, in 2020. His current research interests include hyperspectral image processing, machine learning, and optimization algorithm.

Hongbo Liang received his BS degree in computer science and technology from North Minzu University, Yinchuan, China, 2018, his MEng degree from the School of Computer Science and Engineering, North Minzu University, Yinchuan, China, in 2020. His research interests include hyperspectral image processing, remote sensing image classification, and deep learning.