Automatic Design of Convolutional Neural Network for Hyperspectral Image Classification

Yushi Chen, *Member, IEEE*, Kaiqiang Zhu, Lin Zhu, Xin He, Pedram Ghamisi, *Senior Member, IEEE*, and Jón Atli Benediktsson, *Fellow, IEEE*

***Abstract*— Hyperspectral image (HSI) classification is a core task in the remote sensing community, and recently, deep learning-based methods have shown their capability of accurate classification of HSIs. Among the deep learning-based methods, deep convolutional neural networks (CNNs) have been widely used for the HSI classification. In order to obtain a good classification performance, substantial efforts are required to design a proper deep learning architecture. Furthermore, the manually designed architecture may not fit a specific data set very well. In this paper, the idea of automatic CNN for the HSI classification is proposed for the first time. First, a number of operations, including convolution, pooling, identity, and batch normalization, are selected. Then, a gradient descent-based search algorithm is used to effectively find the optimal deep architecture that is evaluated on the validation data set. After that, the best CNN architecture is selected as the model for the HSI classification. Specifically, the automatic 1-D Auto-CNN and 3-D Auto-CNN are used as spectral and spectral–spatial HSI classifiers, respectively. Furthermore, the cutout is introduced as a regularization technique for the HSI spectral–spatial classification to further improve the classification accuracy. The experiments on four widely used hyperspectral data sets (i.e., Salinas, Pavia University, Kennedy Space Center, and Indiana Pines) show that the automatically designed data-dependent CNNs obtain competitive classification accuracy compared with the state-of-the-art methods. In addition, the automatic design of the deep learning architecture opens a new window for future research, showing the huge potential of using neural architectures' optimization capabilities for the accurate HSI classification.**

***Index Terms*— Convolutional neural network (CNN), deep learning, hyperspectral image (HSI) classification, neural architecture search (NAS).**

I. Introduction

HYPERSPECTRAL images (HSIs) simultaneously capture the spectral and spatial information of an observing target. The detailed spectral information of HSIs increases the power of differentiating land-cover materials, while the fine spatial resolution provides rich spatial structural information [1]. Due to the rich spectral–spatial information provided by HSIs, such data play an important role in a wide range of applications, including urban development, land change monitoring, scene interpretation, and resource management [2]. Among HSI processing techniques, the HSI classification that attempts to allocate a specific class to each pixel in the scene is a fundamental approach and is recognized as a hot topic in the remote sensing community [3].

In the last two decades, a huge number of HSI classification methods have been proposed and, in particular, many supervised machine learning methods have been explored for the HSI classification. Although unsupervised classification only relies on the data to categorize each pixel in the scene, supervised classification usually achieves a much better classification accuracy due to the availability of labeled training samples. In general, there are two categories of supervised HSI classification methods: shallow classifiers and deep classifiers. HSIs provide rich spectral information as well as spatial information. Therefore, in each HSI classification category, spectral feature extraction and spectral–spatial feature extraction are two important research directions [4], [5].

Shallow classifiers for HSI classification mainly contain two steps: feature extraction and training a classifier [6]. In the early studies of HSI classification, the remote sensing community mainly focused on spectral feature extraction. Hence, only spectral information that is available from the hundreds of narrow bands obtained by hyperspectral sensors was used for pixelwise classification. Many feature extraction methods, including principal component analysis (PCA) [7], independent component analysis (ICA) [8], linear discriminant analysis (LDA) [9], and locally linear embedding (LLE) [10], have been used to extract spectral features. The extracted features were fed into classifiers to produce the final classification result. Many supervised classification methods have been used as classifiers, including the k-nearest neighbor approach [11], multinomial logistic regression [12], decision trees [13], ensemble learning [14], support vector machines (SVMs) [15], and random forest [16]. On the one hand, due to the simplicity and effectiveness of spectral classifiers, the spectral classification of HSIs has been a vibrant research area for a long time [17], [18].

On the other hand, due to the availability of both spectral information and spatial information, spectral–spatial

Manuscript received January 1, 2019; revised March 17, 2019; accepted April 7, 2019. Date of publication April 30, 2019; date of current version August 27, 2019. This work was supported by the Natural Science Foundation of China under Grant 61771171. *(Corresponding author: Yushi Chen.)*

Y. Chen, K. Zhu, L. Zhu, and X. He are with the School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China (e-mail: chenyushi@hit.edu.cn; zhukq1995@163.com; 17s105139@stu.hit.edu.cn; 1091636421@qq.com).

P. Ghamisi is with the Helmholtz-Zentrum Dresden-Rossendorf, Helmholtz Institute Freiberg for Resource Technology, 09599 Freiberg, Germany (e-mail: p.ghamisi@gmail.com).

J. A. Benediktsson is with the Faculty of Electrical and Computer Engineering, University of Iceland, IS-107 Reykjavik, Iceland (e-mail: benedikt@hi.is).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2019.2910603

0196-2892 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

classification techniques achieve a better classification performance compared with spectral classifiers [5], [19]. In [20], opening and closing morphological transformation approaches with a structuring element (SE) of increasing size were used to produce morphological profiles (MPs). Subsequently, MPs with partial reconstruction (MPPRs) were proposed to extract spatial features that better model the attributes of different objects and further boosted the HSI classification performances [21], [22]. In order to deal with the computationally inefficient and redundant high-dimensional features available in MPs, maximally stable extremal region-guided MPs (MSER-MPs) were proposed [23]. The fixed SEs of MPs dramatically influence the power of spatial feature extraction. To address this issue, Dalla Mura *et al*. [24] proposed morphological attribute profiles (APs) that apply a sequence of morphological attribute filters to a gray-level image and can be used to model different kinds of structural information [25]. In [26], extinction profiles (EPs) were proposed to extract spatial features from HSIs and demonstrated a superior spatial feature extraction power and better recognition performance than APs [27]. The extracted spectral–spatial features were, then, fed into a classifier (e.g., SVM and random forest) to obtain the final classification result [5]. As the extension of SVM, multiple kernel learning is another mainstream of the spectral–spatial HSI classification, which has a powerful capability to handle the heterogeneous features from spectral–spatial hyperspectral data [28].

Based on the literature on shallow classifiers, it can be observed that the feature extraction is a key factor that affects the performance of HSI classifiers. Deep learning-based methods, which have been proven to be very useful for invariant and discriminant feature extraction, have achieved a significant progress in many research areas, including image classification [29], object detection [30], semantic segmentation [31], and natural language processing [32]. Deep learning models are believed to have the ability of characterizing spectral–spatial features of HSIs in an effective way and yield higher classification accuracies than traditional shallow classifiers. Deep learning-based methods have attracted increased attention and achieved a remarkable progress in recent years in the HSI classification.

In a similar fashion to shallow classifiers, spectral and spectral–spatial classifiers based on deep learning methods have also been explored in recent years. For HSI spectral classification, deep learning models, including stacked autoencoder [33], deep belief network [34], convolutional neural network (CNN) [35], and recurrent neural network (RNN) [36], have been used as deep spectral classifiers. Compared with traditional spectral classifiers, deep learning-based classifiers use multiple nonlinear networks to hierarchically extract the abstract and discriminative features of spectral inputs and yield a good classification performance.

For HSI spectral–spatial classification, due to the joint use of spectral and spatial information, deep learning-based spectral–spatial classifiers usually obtain better classification accuracies compared with spectral classifiers. In recent years, an increasing number of deep learning models have been proposed as deep spectral–spatial classifiers, which are usually based on the stacked autoencoder [37], the deep belief network [38], and the CNN [39]. A survey of deep learning approaches for remote sensing data processing can be found in [40].

Among the aforementioned deep learning models, deep CNN-based methods usually outperform other deep learning models in terms of classification accuracy due to their unique and useful characteristics (e.g., local connections and shared weights) [41]. CNNs and their modifications have been widely used for HSI spectral–spatial classification [42], [43]. In [41], the first spectral–spatial CNN-based classifier was proposed, which was a combination of PCA, deep CNN, and logistic regression. Since the inputs of deep learning models are usually 3-D data, 3-D CNNs were used to extract spectral–spatial features for HSI spectral–spatial classification [44], [45]. Some new types of CNNs, including deep residual network and the densely connected CNN, have been used for spectral–spatial classification [46], [47]. Furthermore, CNNs have been combined with other techniques, such as sparse representation, MPs, and discriminant embedding to further improve the classification performance [48]–[51].

Although CNN-based methods have made a remarkable progress in the HSI classification task in recent years, there are two major problems that should be carefully considered when applying CNN-based methods for HSI classification. First, the deep CNN along with other deep models faces a serious overfitting issue when the number of training samples is limited [52]. Second, the architecture of CNN is manually designed by experts. The design of a proper neural architecture is the key aspect of CNN-based classifiers, which is a time-consuming and error-prone process. In the last decade, a great effort has been dedicated to the proper design of neural architectures [53].

In order to overcome the overfitting problem, a lot of regularization methods have been adopted in the remote sensing community for HSI classification. These regularization techniques include L2 regularization, batch normalization (BN), and dropout [44], [54], [55]. In this paper, we introduce a simple yet effective regularization technique named cutout for CNN-based HSI accurate classification [56]. In contrast to dropout-based regularization which randomly removes neurons in the intermediate feature layer of a CNN, cutout randomly removes continuous square regions in the input layer.

In order to design a proper CNN architecture, there is now a growing interest in designing neural architectures in an automatic manner [57]–[62]. Many different search strategies, including random search, Bayesian optimization, reinforcement learning (RL), evolutionary methods, and gradient-based methods, have been used for neural architecture search (NAS). Among those search strategies, RL and evolutionary algorithms have attracted most attention and achieved remarkable progress. Zoph and Le [59] proposed an RL-based search strategy to search the whole CNN architecture. In the RL-based method, an RNN is built as a controller to generate architectural hyperparameters of neural networks. A CNN with the generated architecture is developed and evaluated. The performance on the validation set is used as the reward to optimize the RNN controller [63]. Evolutionary-based algorithms

have also been used for optimizing the neural architecture and achieved comparable results compared with RL-based methods [60], [64]. In evolutionary-based search methods, each neural network architecture (i.e., model or individual) is encoded as a sequence of numbers. Each model is trained and the performance on a validation set is used as the fitness of the model. According to the fitness, reproductions and mutations are performed, and then, new high-performance "children" (i.e., model) are generated. Recently, Liu *et al.* [62] proposed a method that transforms a discrete search space to a continuous space, which enables the gradient-based optimization method to search for a suitable neural architecture. Due to the simplicity and time-saving property of the gradient-based method, gradient-based search strategies have become a hot research topic in NAS [65], [66].

In this paper, we explore a new paradigm of designing CNN architectures for HSI classification. The automatic design of CNN for HSI classification is introduced and investigated. The main contributions of this paper are summarized as follows.

1) The automatic design of CNN for HSI classification is explored for the first time.
2) Two frameworks, 1-D Auto-CNN and 3-D Auto-CNN, are automatically designed as spectral and spectral–spatial classifiers for HSI classification.
3) In order to mitigate the overfitting problem in CNN and further improve the classification performance, the cutout is investigated as a regularization technique.
4) The 1-D Auto-CNN and 3-D Auto-CNN are tested on four well-known hyperspectral data sets with limited training samples.

The rest of this paper is organized as follows. Sections II and III explain the details of 1-D Auto-CNN and 3-D Auto-CNN, respectively. The experimental results are reported in Section IV. In Section V, conclusions and discussions are presented.

II. Automatic Design of CNN as Spectral Classifier

In this section, the automatically designed CNN for HSI spectral classification is presented. Here, 1-D CNN architectures in the search space A are evaluated and the optimal one is selected as a spectral classifier.

A. CNN for HSI Classification

CNN is a special case of deep learning models, which is inspired by the visual system [67]. There are three parts in CNN: convolution operations, nonlinear operations, and pooling operations. A CNN can be established by the stacking of many aforementioned operations.

A convolution operation can be defined as follows:

$$v_j^l = \sum_{i=1}^{M} v_i^{l-1} * w_{ij}^l + b_j^l \tag{1}$$

where matrix v_i^{l-1} is the ith feature map of the previous $(l-1)$th layer, v_j^l is the jth feature map of the current (l)th layer, and M is the number of input feature maps. w_{ij}^l and b_j^l are randomly initialized and set to zero, respectively. Furthermore, $*$ is the convolution operator.

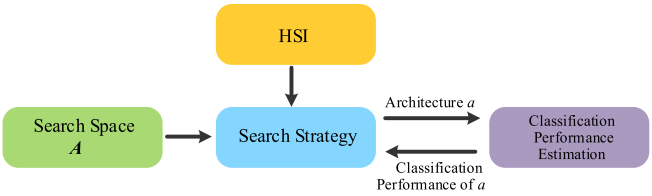


Fig. 1. Illustration of automatic architecture design for HSI classification.

There are several types of nonlinear operations. A widely used nonlinear function is the rectified linear unit (ReLU), which is defined as follows:

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

The pooling operation combines the outputs of a small $N \times N$ (e.g., $N = 2$) patch at one layer into a single output in the next layer. A pooling operation offers invariance by reducing the resolution of the feature maps. The common pooling operations include max pooling and average pooling.

BN is a widely used technique to train deep learning models [54]. $B = \{x_1, x_2, \ldots, x_m\}$ contains values over a minibatch. BN can be formulated as follows:

$$y_i = BN_{\gamma,\beta}(x_i) = \gamma \frac{x_i - \frac{1}{m}\sum_{i=1}^{m} x_i}{\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(x_i - \frac{1}{m}\sum_{i=1}^{m} x_i\right)^2 + \varepsilon}} + \beta. \tag{3}$$

The normalized value is scaled and shifted by learnable parameters γ and β to get the final result y_i.

Deep CNN is a combination of different convolutions, nonlinearity, and pooling operations. Furthermore, there are some special combinations, including multiple branches' structure, shortcut connections, and concatenate connections [68]–[70].

On the one hand, the stacking of different operations brings a flexible CNN, which leads to a powerful feature extraction capability. On the other hand, it brings difficulty in designing a proper architecture for a specific classification task.

B. Principles of Automatic Architecture Design

The available CNN architectures for HSI classification have been manually designed by experts in a trial and error way. In recent years, much progress has been achieved and CNN-based HSI classifiers have demonstrated a good classification performance. However, the manual design of a CNN classifier usually takes a long time. Moreover, deep architectures are data set dependent and they need to be changed and adapted from one data set to another. Therefore, how to automatically design a proper CNN architecture is an important direction in the HSI classification.

Designing a suitable architecture can be seen as a search problem. Fig. 1 shows the procedure of designing an architecture in an automatic way. There are three parts in the search system: search space, search strategy, and classification performance estimation.

Search space is a collection of CNN architectures that can be represented in principle. Appropriate search space

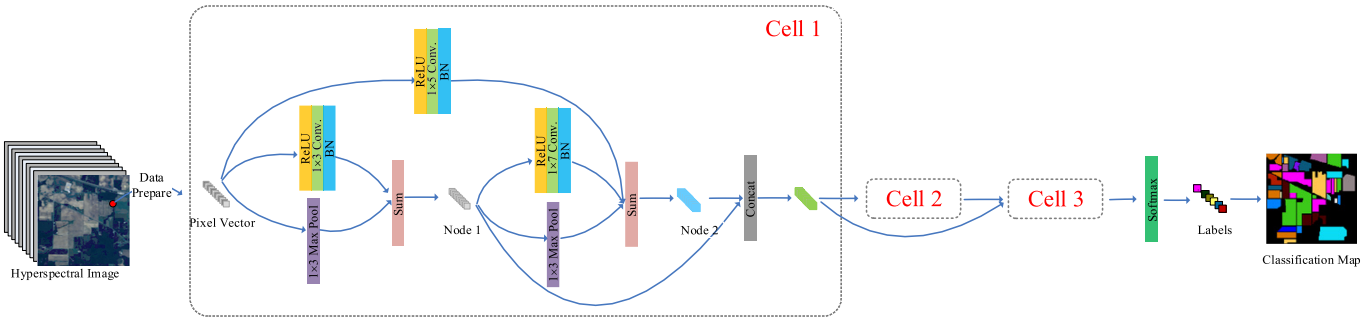


Fig. 2. Framework of 1-D Auto-CNN for HSI classification.

definition can reduce the size of search space and also leads to novel CNN architectures. For HSI spectral classification based on CNN, 1-D convolution is the core part. There are several 1-D convolution filters, including filters with 1×1 convolution, 1×3 convolution, and 1×5 convolution. ReLU is used as nonlinear operation and max pooling is used as pooling operation. Furthermore, multibranch designs and shortcut connections are used to develop the 1-D CNN architecture.

The search strategy is the core part of devising a well-performing CNN architecture. In general, it is a tradeoff between speed and performance. For the task of HSI classification, here, we use a single graphical processing unit (GPU) card to train the classifier. Therefore, a fast algorithm is important.

Classification performance estimation is the last part of the search system in Fig. 1. In this paper, overall classification accuracy on validation samples is used to evaluate the performance of a specific CNN architecture.

C. 1-D Auto-CNN as Spectral Classifier

Fig. 2 shows the framework of the automatically designed CNN for HSI spectral classification. From the framework, one can see that a 1-D Auto-CNN uses pixel vector of HSI as an input. Then, through several computing cells, the features are extracted for the subsequent classification step. Finally, one can obtain the predicted label of each input pixel.

A computing cell is the building block of the final CNN architecture. In such a cell, there are several nodes that are intermediate feature extractor. In Fig. 2, the designed CNN consists of three cells and each cell has two nodes with the sequence.

The details of the cell are described in the following. The inputs of the cell are the output of the previous two cells. Let $n^{(i)}$ be a node in the cell and $o^{(i,j)}$ be an operation set (e.g., convolution, pooling, and skip connection) between $n^{(i)}$ and $n^{(j)}$. Thus, the output of node $n^{(i)}$ is computed based on all the previous nodes in the same cell. The output of the cell h is obtained by concatenating each intermediate node's output

$$n^{(i)} = \sum_{j<i} o^{(i,j)}(n^{(j)}) \tag{4}$$

$$h = \sum_{i} \text{concat}(n^{(i)}). \tag{5}$$

Let O be a set of operations (e.g., convolution and pooling operations). In order to accelerate the search procedure, gradient descent-based methods, which are useful in optimization problems (e.g., a neural network), are considered for the search. Due to the continuity of the search space that is required by the gradient descent-based method, the softmax operation is used over all possible operations to make the categorical choice of an operation continuous. Here, the output of an operation set is the weighted sum of the outputs of each operation in the operation set [62]

$$c_o^{(i,j)} = \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o'\in O}\exp\left(\alpha_{o'}^{(i,j)}\right)} \tag{6}$$

$$o^{(i,j)}(n^{(j)}) = \sum_{o\in O} c_o^{(i,j)} o(n^{(j)}) \tag{7}$$

where $c_o^{(i,j)}$ is the coefficient of operation o in the operation set between node $n^{(i)}$ and $n^{(j)}$. The coefficients $c_o^{(i,j)}$ in the operation set sum to 1 and are obtained by the softmax operation. The coefficients $\alpha_o^{(i,j)}$ are randomly initialized and optimized by the gradient descent method. Through the above-described method, the architecture search space changes from discrete to continuous, which can be solved using gradient descent. At the end of the search, the most likely operation, according to $o^{(i,j)} = \text{argmax}_{o\in O}\alpha_o^{(i,j)}$, is selected as the final operation, and then, the network architecture is determined.

Analogous to the procedure of the manually designed neural architecture where the performance on the validation data set is used to guide the architecture design, Auto-CNN uses the gradient descent method to update the architecture parameter α based on the validation data set. The difference between the two architecture design methods is that the former needs lots of prior experience on behalf of the analyst to adjust the neural structure, while the latter is fully automatic and data-dependent. Now, let L_{train} and L_{val} be the training and validation losses, respectively. The architecture optimization and the training of a CNN are a bilevel optimization problem with α as the architecture variables and w as the weights in CNN

$$\min_{\alpha} L_{\text{val}}(w,\alpha) \tag{8}$$

$$\text{s.t. } w = \underset{w}{\text{argmin}}\, L_{\text{train}}(w,\alpha). \tag{9}$$

Algorithm 1 1-D-Auto-CNN for HSI Spectral Classification

1. **begin**
2. **initialize** number of training samples *Tr* and validation samples *Va*, operation set $\boldsymbol{O}$.
3. feed all pixel vectors into pixel vector set (PVS).
4. for each pixel vector in the PVS:
5. add the pixel vector to the training dataset until the number of training samples is *Tr*.
6. for each pixel vector in the remaining PVS:
7. add the pixel vector to the validation dataset until the number of validation samples is *Va*.
8. add the remaining pixel vectors in PVS to the test dataset.
9. **Architecture Search:**
10. **initialize** learning rate $\boldsymbol{\xi}$ and $\boldsymbol{\varepsilon}$, search epochs, the weight of CNN ($\boldsymbol{w}$), the architecture variable $\boldsymbol{\alpha}$.
11. for every search epoch:
12. for every training and validation mini_batch:
13. $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{\xi}\nabla_{\boldsymbol{w}}\boldsymbol{L}_{\boldsymbol{train}}(\boldsymbol{w}, \boldsymbol{\alpha})$
14. $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \boldsymbol{\varepsilon}\nabla_{\boldsymbol{\alpha}}\boldsymbol{L}_{\boldsymbol{val}}(\boldsymbol{w} - \boldsymbol{\xi}\nabla_{\boldsymbol{w}}\boldsymbol{L}_{\boldsymbol{train}}(\boldsymbol{w}, \boldsymbol{\alpha}), \boldsymbol{\alpha})$
15. choose the best $\boldsymbol{\alpha}^*$ according to the performance on the validation dataset.
16. $\boldsymbol{o}^{(i,j)} = \boldsymbol{argmax}_{\boldsymbol{o}\in\boldsymbol{O}}\boldsymbol{\alpha}_{\boldsymbol{o}}^{*(i,j)}$, obtain the optimal 1-D-Auto-CNN.
17. **Train the optimal 1-D-Auto-CNN and test:**
18. **initialize** the weight ($\boldsymbol{w}^*$) of optimal 1-D-Auto-CNN, learning rate $\boldsymbol{\xi}$, and training epochs.
19. for every training epoch:
20. for every training mini_batch:
21. $\boldsymbol{w}^* \leftarrow \boldsymbol{w}^* - \boldsymbol{\xi}\nabla_{\boldsymbol{w}^*}\boldsymbol{L}_{\boldsymbol{train}}(\boldsymbol{w}^*)$
22. for every test mini_batch:
23. predict = 1-D-Auto-CNN(test min_batch).
24. OA, AA, Kappa = evaluate(predict, test labels).
25. **end**

After the optimization, the obtained α^* and w^* minimize the training and validation losses (i.e., L_{val} and L_{train}), which are used for the architecture design and CNN, respectively.

When the architecture search process is finished, only one most likely operation on the connection between two nodes is preserved. Moreover, the output of each node is computed based on only the two strongest previous nodes. Here, the strength of the connection between two nodes is defined by the coefficient $\alpha_o^{(i,j)}$. Then, the architecture of 1-D Auto-CNN is determined. We train the 1-D Auto-CNN from scratch based on the training data set. The whole process of 1-D Auto-CNN for HSI classification is shown in Algorithm 1.

III. Automatic Design of CNN as Spectral–Spatial Classifier

This section introduces the automatic version of 3-D CNN (i.e., 3-D Auto-CNN) as a spectral–spatial classifier. Furthermore, to mitigate the overfitting problem in classification, a cutout is introduced as a regularization technique.

A. 3-D Auto-CNN as Spectral–Spatial Classifier

Instead of only utilizing the spectral information of HSI in 1-D Auto-CNN, the 3-D Auto-CNN also considers the abundant spatial information of HSI.

Fig. 3 shows the framework of the 3-D Auto-CNN for HSI classification. Different from the 1-D Auto-CNN which only utilizes spectral information, spatial information is also taken into consideration. We choose $K \times K \times B$ neighborhoods of a pixel as an input to the 3-D CNN model, in which K is the width and height of the 3-D input and B is the number of HSI bands. Then, through the designed CNN, the spectral–spatial features are extracted for classification.

In this paper, a 1×1 convolution layer is used as the bottleneck layer, which reduces the number of input data channels. Due to the usage of the bottleneck layer, the time cost and complexity in automatically searching and evaluating the best network architecture are dramatically reduced. Furthermore, it contributes to the stability of the training process for the 3-D Auto-CNN.

In contrast to the 1-D Auto-CNN, in 3-D architectures, the node in each computing cell can leverage spectral–spatial features together from the previous nodes. The output of each cell is defined as the depthwise concatenation of all the intermediate nodes qualified with abundant spatial features in addition to the spectral information as in the 1-D Auto-CNN.

B. Cutout-Based Regularization

Due to the high dimensionality of HSI inputs, together with a large number of trainable parameters in deep learning models, deep learning-based HSI classification methods always face a serious overfitting problem. Overfitting is a situation where the models do not generalize, i.e., training error is low but the test error is high. In the HSI classification, overfitting is even more serious when the number of training samples is limited. Therefore, deep learning-based HSI classification methods need a proper regularization technique in order to mitigate overfitting.

In this paper, we introduce a simple yet effective regularization technique named cutout for CNN-based HSI accurate classification. Another technique, dropout, is popular to handle overfitting and is widely used in many research areas, including HSI classification [44], [55]. It should be noted that the cutout approach can be combined with dropout to further mitigate overfitting in the HSI classification under the condition of limited training samples.

Unlike dropout-based regularization, cutout randomly removes regions at the input layer rather than in the feature layers and masks the inputs with continuous square regions rather than individual pixels. Furthermore, the cutout is a method that is quite easy to implement.

For a specific data set such as HSI, cutout randomly removes the contiguous regions of the bands which are randomly selected. The cutout strategy applied to the HSI Indiana data set is shown in Fig. 4, which demonstrates the results of the cutout operation on the same scene at different bands and positions. The whole process of 3-D Auto-CNN-Cutout for HSI classification is shown in Algorithm 2.

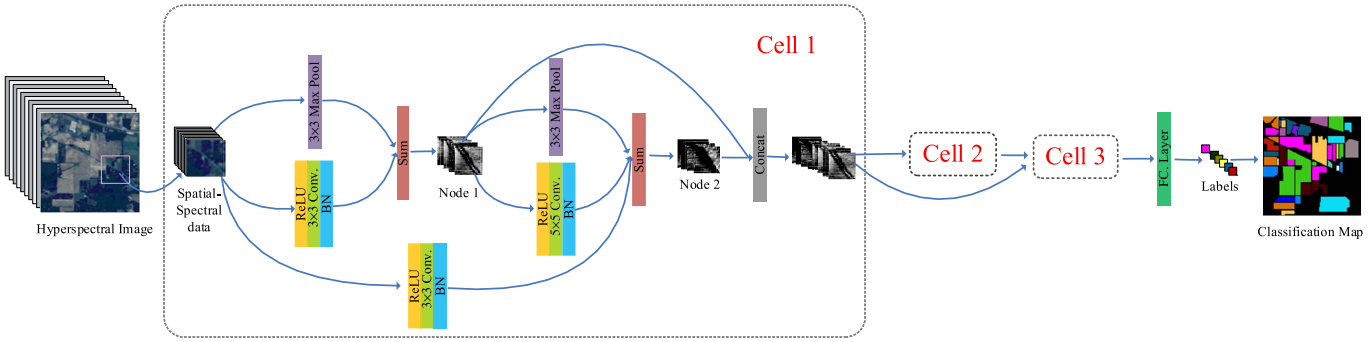


Fig. 3. Framework of 3-D Auto-CNN for HSI classification.

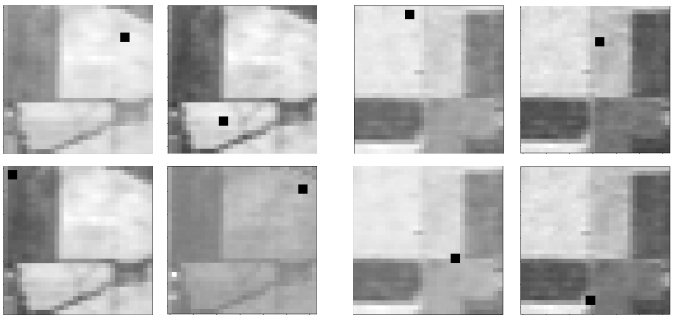


Fig. 4. Illustration of two samples with a cutout operation on the Indian Pines data set. (a) Sample 1. (b) Sample 2.

IV. Experimental Results

A. Data Description

In this paper, the effectiveness of the proposed methods was validated on four widely used hyperspectral data sets with different settings. From Fig. 5, the four data sets are Salinas Valley in California (Salinas), an urban site over the University of Pavia, Italy (Pavia), Kennedy Space Center (KSC) in Florida, and a mixed vegetation site over the Indian Pines test area in Northwestern Indiana (Indian Pines).

The first data set, Salinas, was collected by the AVIRIS instrument with 224-band over Salinas Valley, California. The available data set is composed of 204 bands with 512 × 217 pixels after the low signal-to-noise ratio (SNR) bands were removed. This HSI is of high spatial resolution (3.7 mpixels) and the ground reference map covered 16 classes of interest.

The second data set is an urban site over the University of Pavia, which was collected by the ROSIS-3 sensor. The data set is of 1.3-m spatial resolution and consists of 610 × 340 pixel vectors. The HSI is composed of 115 spectral bands ranging from 430 to 860 nm. In the experiment, some bands have been removed due to the low SNR, and the remaining 103 channels were used for the classification. Nine different land-cover classes were provided in the ground reference map.

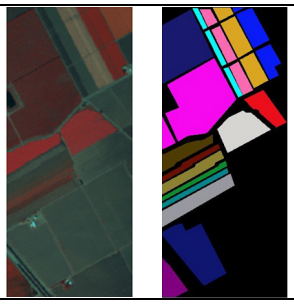
The third data set was captured by the airborne AVIRIS sensor over KSC, Florida. The KSC data set, which has an altitude of approximately 20 km, is composed of 512 × 614 pixels with a spatial resolution of 18 m. After removing water absorption and low SNR bands, 176 bands were used for the analysis. Thirteen classes were selected in the ground reference map for the classification task.

The fourth data set was captured by the AVIRIS sensor over the Indian Pines test area, which was a mixed

Algorithm 2 3-D-Auto-CNN With Cutout for HSI Spectral–Spatial Classification

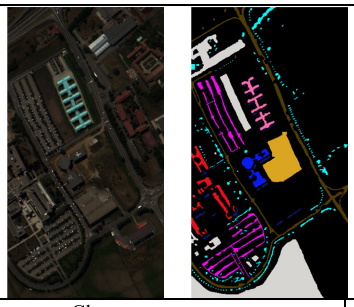
begin
  initialize number of training samples *Tr* and validation samples *Va*, neighborhood size $\boldsymbol{K}$, operation set $\mathbf{O}$.
  for each pixel:
   crop the $\boldsymbol{K} \times \boldsymbol{K}$ neighboring region of each pixel and add it to the 3-D sample set.
  split the 3-D sample set into the training dataset, the validation dataset and the test dataset according to *Tr* and *Va*.
  Architecture Search:
    initialize learning rate $\boldsymbol{\xi}$ and $\boldsymbol{\varepsilon}$, search epochs, the weight of CNN ($\boldsymbol{w}$), the architecture variable $\boldsymbol{\alpha}$, number of cutout bands N, cutout length L.
    for each sample in the training dataset:
      randomly select N bands.
      for each selected bands:
        set the pixel values of $\boldsymbol{L} \times \boldsymbol{L}$ region (randomly select) to zero.
    for every search epoch:
      for every training and validation mini_batch:
        $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{\xi} \nabla_{\boldsymbol{w}} \boldsymbol{L}_{\boldsymbol{train}}(\boldsymbol{w}, \boldsymbol{\alpha})$
        $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \boldsymbol{\varepsilon} \nabla_{\boldsymbol{\alpha}} \boldsymbol{L}_{\boldsymbol{val}}(\boldsymbol{w} - \boldsymbol{\xi} \nabla_{\boldsymbol{w}} \boldsymbol{L}_{\boldsymbol{train}}(\boldsymbol{w}, \boldsymbol{\alpha}), \boldsymbol{\alpha})$
    choose the best $\boldsymbol{\alpha}^*$ according to the performance on validation dataset.
    $\boldsymbol{o}^{(i,j)} = \boldsymbol{argmax}_{o \in \boldsymbol{O}} \boldsymbol{\alpha}_o^{*(i,j)}$, obtain the best 3-D-Auto-CNN architecture $\boldsymbol{\mathcal{N}}$.
  Train the optimal 3-D-Auto-CNN and test:
    initialize the weight ($\boldsymbol{w}^*$) of $\boldsymbol{\mathcal{N}}$, learning rate $\boldsymbol{\xi}$, and training epochs.
    for every training epoch:
      for every training mini_batch:
        $\boldsymbol{w}^* \leftarrow \boldsymbol{w}^* - \boldsymbol{\xi} \nabla_{\boldsymbol{w}^*} \boldsymbol{L}_{\boldsymbol{train}}(\boldsymbol{w}^*)$
    for every test mini_batch:
      predict = 3-D-Auto-CNN(test min_batch).
    OA, AA, Kappa = evaluate(predict, test labels).
end

Salinas



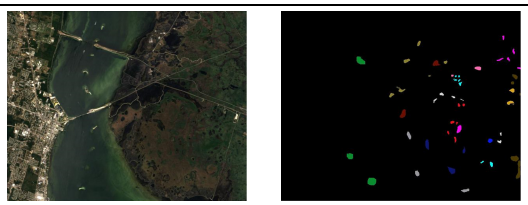
Class			Samples
No.	Color	Name	Numbers
1		Brocoli_green_weeds_1	1977
2		Brocoli_green_weeds_2	3726
3		Fallow	1976
4		Fallow_rough_plow	1394
5		Fallow_smooth	2678
6		Stubble	3959
7		Celery	3579
8		Grapes_untrained	11213
9		Soil_vinyard_develop	6197
10		Corn_senesced_green_weeds	3249
11		Lettuce_romaine_4wk	1058
12		Lettuce_romaine_5wk	1908
13		Lettuce_romaine_6wk	909
14		Lettuce_romaine_7wk	1061
15		Vinyard_untrained	7164
16		Vinyard_vertical_trellis	1737
Total			53785

Pavia



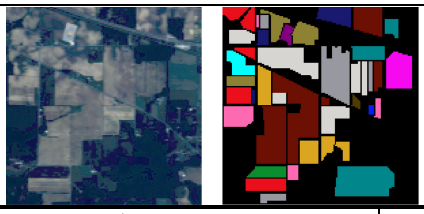
Class			Samples
No.	Color	Name	Numbers
1		Asphalt	6851
2		Meadows	18686
3		Gravel	2207
4		Trees	3436
5		Metal sheets	1378
6		Bare soil	5104
7		Bitumen	1356
8		Bricks	3878
9		Shadow	1026
Total			43922

KSC



Class			Samples
No.	Color	Name	Numbers
1		Scrub	761
2		Willow swamp	243
3		CP hammock	256
4		Slash pine	252
5		Oak/Broadleaf	161
6		Hardwood	229
7		Swamp	105
8		Graminoid marsh	431
9		Spartina marsh	520
10		Cattail marsh	404
11		Salt marsh	419
12		Mud flats	503
13		Water	927
Total			5211

Indian Pines



Class			Samples
No.	Color	Name	Numbers
1		Alfalfa	46
2		Corn-notill	1428
3		Corn-min	830
4		Corn	237
5		Grass-pasture	483
6		Grass-trees	730
7		Grass-pasture-mowed	28
8		Hay-windrowed	478
9		Oats	20
10		Soybean-notill	972
11		Soybean-mintill	2455
12		Soybean-clean	593
13		Wheat	205
14		Woods	1265
15		Buildings-Grass-Trees	386
16		Stone-Steel-Towers	93
Total			10249

Fig. 5. False color composite, ground reference map, and a number of available samples of the four widely used HSI data sets (Salinas, Pavia, KSC, and Indian Pines).

vegetation site in Northwestern Indiana (Indian Pines). The data set consists of 145 × 145 pixels with 220 spectral bands in the wavelength range of 400–2500 nm. After removing water absorption bands, the number of bands was reduced to 200. The ground reference map covers 16 classes of interest.

TABLE I
CLASSIFICATION WITH SPECTRAL FEATURES ON THE SALINAS DATA SET USING 200 TRAINING SAMPLES

Method	RF-200	MLP	L-SVM	RBF-SVM	RNN	1-D CNN	1-D DCNN	**1-D Auto-CNN**
OA(%)	83.11±0.97	84.13±0.50	85.62±2.35	86.89±1.20	79.67±2.42	84.93±0.73	83.63±2.41	**87.15±1.37**
AA(%)	85.87±1.77	88.71±0.60	89.54±2.04	89.88±0.98	83.22±1.25	89.97±0.58	86.40±4.36	**91.32±1.63**
K×100	81.12±1.11	82.21±0.57	83.88±2.67	85.35±1.34	77.38±2.71	83.20±0.82	81.74±2.74	**85.70±1.50**
Brocoli_green_weeds_1	94.42±2.21	97.32±0.68	95.63±2.41	98.41±1.38	94.70±4.40	96.67±1.85	97.95±1.36	96.75±1.56
Brocoli_green_weeds_2	99.91±0.03	98.98±0.33	97.78±1.40	95.93±2.27	98.88±0.61	98.22±1.18	98.41±1.21	99.26±0.45
Fallow	86.09±9.15	84.49±4.30	90.11±6.89	91.24±10.68	59.06±16.51	85.25±3.30	86.12±13.77	79.46±21.58
Fallow_rough_plow	93.08±2.65	99.82±0.06	98.76±1.31	99.08±0.61	99.03±0.32	99.65±0.25	96.18±8.76	99.09±0.56
Fallow_smooth	96.99±0.91	98.37±0.37	92.25±8.36	95.45±1.33	95.67±0.41	95.32±0.51	91.17±5.28	97.21±1.46
Stubble	97.06±2.25	99.64±0.15	98.73±1.02	98.68±0.87	98.81±0.43	97.48±1.10	97.52±1.58	99.68±0.17
Celery	97.83±0.79	98.96±0.22	99.10±0.62	98.21±1.18	96.95±2.31	97.81±0.90	99.01±0.32	99.35±0.25
Grapes_untrained	79.57±1.47	88.22±3.14	91.14±4.06	84.95±4.66	60.90±4.89	77.09±2.92	75.95±11.48	75.82±10.02
Soil_vinyard_develop	98.70±0.19	98.25±0.28	97.29±2.11	97.82±1.41	96.28±0.21	97.69±0.46	97.77±1.11	99.05±0.75
Corn_senesced_green_weeds	81.46±4.80	85.18±1.48	80.39±7.45	74.15±11.75	78.21±9.94	84.69±2.44	78.37±14.45	87.54±3.78
Lettuce_romaine_4wk	71.68±11.60	81.57±4.53	87.56±8.47	85.65±8.76	77.31±7.63	89.69±1.24	71.92±23.32	89.15±2.30
Lettuce_romaine_5wk	87.87±7.02	94.43±4.24	99.41±0.48	99.41±0.41	78.84±13.95	97.12±1.48	88.34±17.31	96.99±4.20
Lettuce_romaine_6wk	88.54±10.74	98.67±0.25	96.63±2.92	93.83±9.09	90.36±7.67	98.50±0.36	80.92±28.25	98.36±1.08
Lettuce_romaine_7wk	84.24±9.62	88.14±1.52	89.88±3.98	88.04±6.86	91.26±3.56	88.37±1.69	88.61±6.92	90.61±5.26
Vinyard_untrained	43.76±11.84	25.68±7.43	35.31±20.90	57.42±4.56	62.62±4.82	49.89±5.60	54.38±19.32	63.47±10.73
Vinyard_vertical_trellis	72.79±6.51	81.63±4.14	82.67±16.19	79.79±12.37	52.67±0.83	86.01±5.20	79.78±16.25	89.26±5.84

TABLE II
CLASSIFICATION WITH SPECTRAL FEATURES ON THE PAVIA DATA SET USING 200 TRAINING SAMPLES

Method	RF-200	MLP	L-SVM	RBF-SVM	RNN	1-D CNN	1-D DCNN	**1-D Auto-CNN**
OA(%)	77.21±0.81	82.32±1.26	80.52±0.91	83.55±1.55	76.86±2.10	81.47±1.47	81.78±1.56	**84.63±1.80**
AA(%)	68.67±3.74	76.44±2.75	69.51±1.63	78.26±2.63	71.47±4.04	77.88±1.62	77.28±2.38	**80.87±1.64**
K×100	69.06±1.23	76.47±1.71	73.60±1.33	78.02±2.16	69.11±2.68	75.49±1.94	75.72±2.09	**79.70±2.26**
Asphalt	81.77±3.73	84.76±4.99	88.93±2.87	81.82±3.20	78.51±4.47	79.83±3.89	83.76±2.74	83.40±3.40
Meadows	94.66±2.09	93.06±1.15	96.08±2.21	95.45±2.47	90.16±4.99	91.57±2.17	92.96±3.47	93.32±3.18
Gravel	30.08±12.15	50.18±14.35	39.75±13.65	40.66±17.03	36.66±15.34	56.97±12.10	48.97±20.97	61.52±5.05
Trees	63.91±10.41	81.47±5.41	75.29±5.68	77.66±4.26	70.29±8.16	79.38±4.50	79.20±6.52	78.86±4.38
Metal sheets	80.83±21.94	92.27±8.05	98.46±0.81	92.31±12.69	82.65±29.81	95.27±7.08	99.20±0.29	98.25±0.98
Bare soil	36.01±4.94	57.36±8.15	44.19±10.32	62.30±11.60	46.68±8.34	58.15±7.29	57.56±10.49	73.34±5.42
Bitumen	53.59±22.67	54.33±17.58	3.01±4.89	74.19±11.21	65.83±13.14	62.41±13.33	61.52±17.47	64.56±9.15
Bricks	79.50±9.33	81.70±4.36	82.76±8.36	83.76±6.94	74.64±9.31	79.66±5.23	73.69±11.16	76.86±7.29
Shadow	97.71±0.54	92.79±9.25	97.14±4.01	96.20±5.92	97.77±3.68	97.68±2.50	98.63±0.57	97.69±2.30

B. Experimental Setup

For the 1-D Auto-CNN, the operation set O contains separable convolution with different kernel sizes (i.e., 1×3, 1×5, 1×7, and 1×9) [71], 1×3 average pooling, 1×3 max pooling, zero, and identity. The spectral vector of each pixel is used as the input of 1-D CNN model. We first use a 1×1 bottleneck convolution layer to reduce the spectral dimension from B (i.e., the number of HSI bands) to 32, which can alleviate that the overfitting problem under the circumstance of the number of available training samples is limited.

For the 3-D Auto-CNN, the operation set O is composed of 3×3 and 5×5 separable convolution, 3×3 and 5×5 dilated separable convolution [72], 3×3 average pooling, 3×3 max pooling, and identity. For these data sets, we choose $32 \times 32 \times B$ neighborhoods of a pixel as an input to the 3-D CNN model, in which B is the number of HSI bands. To stabilize the training process and reduce the processing time of the architecture search and evaluation, a 1×1 bottleneck convolution layer is first used, and therefore, the size of input HSI data can be condensed from $32 \times 32 \times B$ to $32 \times 32 \times b$, in which b stands for the number of output feature maps of the *bottleneck* layer. Here, we set b to 10.

As mentioned in Section III, the computing cell is the building block of the Auto-CNN architecture. In both 1-D and

TABLE III
CLASSIFICATION WITH SPECTRAL FEATURES ON THE KSC DATA SET USING 200 TRAINING SAMPLES

Method	RF-200	MLP	L-SVM	RBF-SVM	RNN	1-D CNN	1-D DCNN	**1-D Auto-CNN**
OA(%)	81.43±0.17	85.15±0.81	85.67±1.76	86.11±1.55	82.03±1.33	84.80±0.97	81.33±2.12	**88.40±2.02**
AA(%)	74.78±0.91	78.78±0.98	79.37±1.94	79.14±2.10	74.68±1.21	79.32±1.48	72.27±3.71	**82.34±2.63**
K×100	79.30±0.19	83.45±0.90	84.04±1.94	84.51±1.73	79.97±1.48	83.08±1.08	79.16±2.38	**87.08±2.23**
Scrub	92.66±2.23	91.92±1.43	91.03±5.04	92.21±2.99	91.91±1.71	90.14±2.29	90.48±3.21	88.96±7.77
Willow swamp	79.65±4.29	74.60±4.83	83.59±5.12	80.25±8.22	68.67±12.32	60.96±5.48	83.96±8.50	89.38±3.42
CP hammock	81.63±7.46	71.90±3.38	86.43±6.91	87.37±5.93	87.79±2.99	74.73±7.71	72.87±21.60	83.74±6.75
Slash pine	55.94±12.02	41.13±7.33	44.79±14.62	54.01±14.56	45.55±12.10	52.54±6.48	41.01±20.92	60.08±10.93
Oak/Broadleaf	49.68±9.12	64.93±4.54	58.88±10.42	50.87±13.14	41.09±12.65	73.10±1.74	37.84±14.33	55.35±9.88
Hardwood	41.62±9.11	41.67±4.86	52.36±9.66	45.27±7.62	37.47±7.14	34.86±7.75	33.64±12.40	56.79±9.95
Swamp	71.89±4.95	86.76±4.97	69.95±14.64	69.71±10.66	80.79±7.99	95.94±4.78	53.11±18.88	68.58±19.96
Graminoid marsh	62.80±7.34	85.40±2.50	83.33±6.75	86.28±4.65	70.17±5.30	85.50±3.66	73.60±6.53	88.40±4.58
Spartina marsh	85.18±7.78	91.82±2.85	89.42±13.08	91.60±5.41	89.23±2.40	91.08±2.07	86.27±8.84	94.30±4.69
Cattail marsh	80.22±2.87	91.70±2.20	93.61±1.82	89.78±4.60	79.41±3.08	94.74±1.74	92.22±3.05	95.28±2.37
Salt marsh	92.35±0.30	97.95±0.60	93.32±4.73	92.91±4.85	95.70±3.44	95.40±2.20	93.80±3.84	95.40±4.37
Mud flats	78.75±5.38	84.43±0.86	85.14±5.30	88.56±3.84	83.31±4.25	82.30±1.52	80.84±6.70	94.17±2.25
Water	99.81±0.14	100.00±0.00	99.94±0.06	99.94±0.06	99.71±0.37	99.86±0.01	99.92±0.05	100.00±0.00

TABLE IV
CLASSIFICATION WITH SPECTRAL FEATURES ON THE INDIAN PINES DATA SET USING 200 TRAINING SAMPLES

Method	RF-200	MLP	L-SVM	RBF-SVM	RNN	1-D CNN	1-D DCNN	**1-D Auto-CNN**
OA(%)	60.82±1.51	61.13±1.63	62.57±1.62	64.78±2.42	57.51±2.67	60.30±1.41	61.14±3.92	**65.35±2.94**
AA(%)	43.22±2.35	49.44±2.71	51.03±2.30	52.97±5.04	44.52±4.25	46.90±1.75	50.42±5.11	**54.73±4.22**
K×100	54.40±1.89	55.03±1.96	56.75±2.08	59.42±3.12	51.24±3.28	54.38±1.59	55.14±4.65	**60.41±3.23**
Alfalfa	0.22±0.67	16.52±17.93	10.55±12.12	19.71±19.77	10.87±17.61	13.78±12.91	21.69±27.63	19.58±19.42
Corn-notill	51.55±4.94	50.38±9.62	50.36±6.15	54.98±8.95	49.76±12.18	55.30±3.05	54.15±9.77	60.16±8.24
Corn-min	25.90±11.03	35.52±9.79	32.32±11.73	44.86±9.82	30.93±14.52	43.09±4.04	50.26±9.33	44.12±12.07
Corn	14.23±9.90	17.06±9.41	28.16±16.48	30.02±22.26	24.03±11.39	3.46±2.01	30.89±11.69	25.35±15.82
Grass-pasture	51.44±16.49	58.18±10.89	71.48±13.31	63.40±18.01	67.24±13.23	55.49±10.68	46.16±21.31	77.80±4.54
Grass-trees	91.13±5.79	86.09±8.34	87.97±6.20	92.31±5.77	72.03±13.57	86.07±3.28	82.95±10.75	90.99±4.86
Grass-pasture-mowed	6.11±10.47	6.15±18.46	14.21±28.45	33.62±41.51	8.35±14.80	0.00±0.00	24.84±26.35	35.63±36.89
Hay-windrowed	97.58±2.39	87.94±7.85	96.87±2.93	96.60±3.34	89.75±10.46	89.42±4.71	94.33±6.75	95.87±5.24
Oats	0.53±1.58	5.79±10.11	3.16±9.47	2.19±5.07	9.39±17.77	0.00±0.00	1.05±3.16	5.31±9.74
Soybean-notill	43.34±10.43	44.48±12.54	44.39±6.95	53.75±10.95	49.62±18.81	49.57±3.47	41.17±21.43	55.93±9.41
Soybean-mintill	77.47±4.10	73.64±6.44	74.99±4.45	72.96±4.41	66.49±5.86	66.07±3.30	69.44±8.23	68.73±9.50
Soybean-clean	25.79±11.62	23.02±9.54	24.98±8.62	28.08±13.46	24.02±15.77	31.34±3.58	30.64±17.15	36.96±7.91
Wheat	66.25±23.07	82.75±21.77	82.95±12.06	76.77±21.28	66.35±18.40	88.99±6.48	78.80±18.87	87.33±10.85
Woods	90.55±5.00	89.01±3.64	89.24±6.39	92.17±4.19	85.09±5.86	96.81±4.51	85.59±7.73	84.90±4.19
Buildings-Grass-Trees	22.60±7.30	32.39±9.73	29.02±10.77	23.76±10.58	21.23±10.01	17.41±4.23	30.20±15.28	39.02±17.05
Stone-Steel-Towers	26.85±26.51	82.13±4.81	75.74±24.71	62.37±32.01	37.17±39.95	63.67±24.15	64.60±31.64	48.02±42.87

3-D Auto-CNNs, there are two cell types (i.e., a normal cell and reduction cell). In the normal cell, the feature maps are padded and the stride of all operations is set to 1, while in the reduction cell, the feature maps are also padded but the stride is set to 2 for all operations in order to reduce the resolution of the feature map. In the architecture search process, the number of computing cells is set to 2 for both the 1-D Auto-CNN and the 3-D Auto-CNN in an attempt to reduce the time cost and accelerate the architecture search process. The best architecture used for evaluation is picked based on the classification performance on the validation data set, while in the process of architecture evaluation, we build a bigger network to extract more robust features for classification. The 1-D Auto-CNN and 3-D Auto-CNN contain three and four cells, respectively.

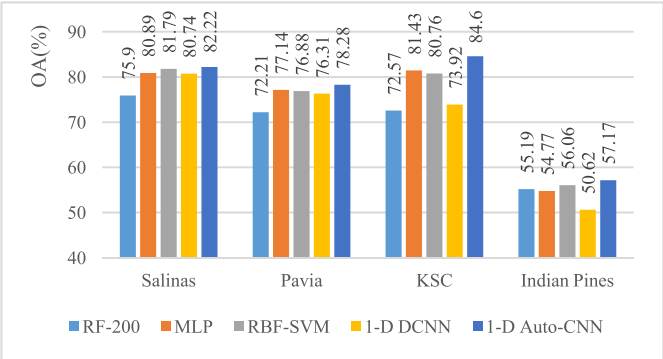


Fig. 6. Classification results of 1-D Auto-CNN and other state-of-the-art methods on four HSI data sets with 100 training samples.

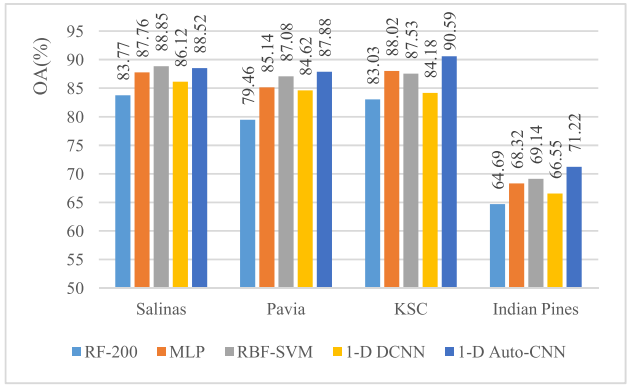


Fig. 7. Classification results of 1-D Auto-CNN and other state-of-the-art methods on four HSI data sets with 300 training samples.

Each cell consists of four nodes. The number of reduction cells is set to 1 and 2 in the architecture search and evaluation process for both 1-D and 3-D Auto-CNNs.

Due to the fact that the number of available training samples is limited, L2 regularization and cutout are used to solve the overfitting problem. L2 regularization is used for both 1-D Auto-CNN and 3-D Auto-CNN and the selected weight decay is 0.0003. The 3-D Auto-CNN also uses cutout to further boost the classification performance. This regularization is only used in the architecture search but not in the evaluation process, and the cut length is set to 2. In more detail, we randomly choose 10% of the total number of B bands and different positions in the input data to perform cutout, which can more robustly contribute to the search procedure.

Furthermore, for the 1-D Auto-CNN, the training epochs in the architecture search and evaluation process are 300 and 500, respectively. The learning rate for the weights of the CNN in the search and evaluation process is both initialized to 0.004. For 3-D Auto-CNN, the training epochs are set to 100 and 120 for the architecture search and evaluation process, respectively. The learning rate for the weight of CNN in the search and evaluation process is initialized to 0.025 and 0.05, respectively. In 1-D Auto-CNN and 3-D Auto-CNN, the learning rate for architecture variables is 0.0003. We use the Adam optimizer to train the model [73]. The learning rate for the weights of CNN decreases dynamically along the training process.

For all experiments in this study, we split the labeled samples of the HSI data sets into three subsets, i.e., training, validation, and test data sets. In detail, we randomly chose 200 and 100 samples from HSI labeled samples as training and validation data sets. The training data set is used to train the weights and biases of each neuron in the model, while the architecture variables are optimized based on the validation data set. After the optimal architecture is determined and trained, all the remaining labeled samples serve as the test data set to evaluate the capability of the network and obtain the final classification results. Furthermore, in order to demonstrate the effectiveness of the proposed model, all experiments with 100 and 300 training samples are also implemented. Three evaluation criteria are investigated, i.e., overall accuracy (OA), average accuracy (AA), and Kappa coefficients (K).

C. Classification Results for the 1-D Auto-CNNs

In this section, some classical classification methods that are only based on spectral information are conducted to give a comprehensive comparison with the proposed 1-D Auto-CNN. Random forest with 200 decision trees (RF-200) is used for HSI classification. Multiple-layer perceptron (MLP) is used in this experiment, which is a fully connected neural network with one hidden layer composed of 64 hidden neural units. SVM, which is famous for its good performance for HSI classification, provides a competitive classification performance. L-SVM is a linear SVM with no kernel function, while radial basis function (RBF)-SVM uses the radial basis function as the kernel. In the RBF-SVM, a grid search method is used to define the most appropriate C and γ. In this experiment, the search range is the exponentially growing sequences of C and γ ($C = 10^{-3}, 10^{-2}, \ldots, 10^{3}$ and $\gamma = 10^{-3}, 10^{-2}, \ldots, 10^{3}$). We try pairs of (C, γ) and select the one with the best classification accuracy on the validation set. A single-layer RNN with a gated recurrent unit and a hyperbolic tangent activation function are also adopted. The architecture for the 1-D CNN is designed as in [35], which contains an input layer, a convolutional layer, a max-pooling layer, a fully connected layer, and an output layer. Furthermore, to give a competitive comparison, deep CNN with multiple convolutional layers (1-D DCNN) has also been used for comparison [44].

Tables I–IV show the classification results obtained with the aforementioned experimental settings. All the experiments have been run ten times with different random training samples. The classification accuracy is given in the form of mean $\pm$ standard deviation. The 1-D Auto-CNN shows better performances in terms of accuracies on all four data sets. For the Salinas data set, the 1-D Auto-CNN improves the OA, AA, and K of the RBF-SVM by 0.26%, 1.44%, and 0.0035, respectively (see Table I). For the Pavia data set, the OA of 1-D Auto-CNN is 84.63%, which is increased by 1.08% and 2.85% compared with RBF-SVM and 1-D-DCNN, respectively (see Table II). For the KSC data set, as can be seen, the 1-D Auto-CNN exhibits the best OA, AA, and K with the improvements of 2.29%, 3.2%, and 0.0257 over RBF-SVM, respectively (see Table III). The Indian Pines data set has the same classification results as the aforementioned three data sets. In addition,

TABLE V
CLASSIFICATION WITH SPECTRAL–SPATIAL FEATURES ON THE SALINAS DATA SET USING 200 TRAINING SAMPLES

Method	3-D RBF-SVM	EMP-RF	EMP-SVM	EMP-CNN	3-D CNN	SSRN [74]	**3-D Auto-CNN**	**3-D Auto-CNN-Cutout**
OA(%)	87.85±2.36	91.64±0.82	89.09±1.44	92.71±0.66	90.68±1.28	91.16±1.64	93.37±1.79	**94.65±0.70**
AA(%)	86.74±4.54	92.05±2.14	91.90±1.01	93.22±0.62	88.18±1.25	93.59±2.35	92.95±2.33	**93.99±2.01**
K×100	86.37±2.66	90.68±0.92	87.82±1.64	91.83±0.74	89.56±1.45	90.16±1.82	92.57±2.00	**93.87±0.81**
Brocoli_green_weeds_1	65.82±23.38	99.39±0.48	96.38±0.92	93.91±6.37	81.76±6.68	99.59±0.72	70.04±4.17	94.15±6.02
Brocoli_green_weeds_2	92.14±5.93	98.64±2.55	99.30±0.57	85.93±8.49	94.88±6.07	99.54±1.35	97.08±4.17	98.36±2.09
Fallow	93.77±8.03	86.65±7.69	78.77±13.20	93.31±11.47	91.35±4.62	82.03±20.05	92.89±7.52	93.59±7.15
Fallow_rough_plow	94.93±4.93	98.91±1.40	99.59±0.37	97.94±1.57	88.31±7.49	98.61±1.57	88.28±3.21	98.56±2.30
Fallow_smooth	96.75±3.07	91.71±13.54	94.02±4.59	94.90±4.25	95.60±4.40	97.13±3.98	94.62±4.26	98.61±1.15
Stubble	98.24±2.07	97.71±1.66	99.60±0.29	98.91±1.54	99.76±0.63	99.81±0.49	98.75±1.64	99.67±0.16
Celery	94.68±5.76	99.21±0.58	97.27±1.41	97.42±1.66	97.54±2.03	99.62±0.58	98.03±2.16	97.96±2.39
Grapes_untrained	84.69±5.68	89.18±2.53	86.15±3.90	93.71±2.40	88.48±3.13	81.13±8.86	92.09±2.51	90.98±1.13
Soil_vinyard_develop	99.19±1.31	99.56±0.36	98.96±0.50	98.41±0.86	98.76±1.70	98.96±2.30	98.66±1.13	99.74±0.16
Corn_senesced_green_weeds	90.61±3.31	92.34±6.93	91.33±3.01	95.93±3.25	95.20±5.57	91.58±3.72	99.60±0.51	99.78±0.31
Lettuce_romaine_4wk	87.48±15.70	82.34±17.84	91.12±5.64	93.25±7.93	77.33±8.43	83.38±28.67	95.72±3.14	79.51±4.37
Lettuce_romaine_5wk	96.57±3.17	92.30±9.95	99.84±0.17	92.98±9.99	89.66±9.04	98.72±3.02	98.83±0.84	99.84±0.22
Lettuce_romaine_6wk	86.11±17.38	93.16±12.10	98.95±0.31	91.48±11.01	87.92±9.43	95.30±5.09	90.48±0.69	90.86±0.98
Lettuce_romaine_7wk	89.87±21.48	85.10±28.67	96.41±2.53	98.07±2.78	90.53±8.71	96.95±3.15	82.39±1.14	87.17±3.14
Vinyard_untrained	68.89±14.36	78.36±8.13	63.02±14.82	78.57±5.36	81.78±11.24	79.72±7.47	94.44±2.62	97.40±2.02
Vinyard_vertical_trellis	48.04±27.41	88.30±6.08	79.74±8.85	86.80±7.39	61.12±14.38	95.37±2.84	85.52±5.64	77.63±4.31

TABLE VI
CLASSIFICATION WITH SPECTRAL–SPATIAL FEATURES ON THE PAVIA DATA SET USING 200 TRAINING SAMPLES

Method	3-D RBF-SVM	EMP-RF	EMP-SVM	EMP-CNN	3-D CNN	SSRN [74]	**3-D Auto-CNN**	**3-D Auto-CNN-Cutout**
OA(%)	80.06±1.52	88.29±2.24	89.30±1.15	91.65±1.22	91.79±0.98	91.65±2.64	93.67±1.10	**93.88±1.86**
AA(%)	69.48±3.03	81.87±5.41	83.97±4.27	83.13±3.57	84.59±2.97	**89.00±3.27**	87.17±1.59	88.19±2.72
K×100	74.19±2.02	84.36±3.09	85.78±1.58	89.42±1.56	89.58±1.26	89.16±3.37	92.01±1.29	**92.15±2.29**
Asphalt	88.75±2.70	94.22±2.45	92.42±2.05	91.99±2.81	92.11±2.40	95.08±2.46	94.65±0.22	94.14±2.53
Meadows	94.94±1.47	96.21±1.71	96.36±2.54	97.95±1.32	98.88±0.72	96.20±4.20	98.68±0.13	92.78±1.89
Gravel	34.89±15.46	70.14±10.05	61.42±8.18	49.02±10.07	59.99±13.85	69.92±18.36	91.13±1.78	80.60±2.23
Trees	64.11±9.76	85.59±10.22	92.30±6.88	89.25±4.96	93.35±4.35	90.84±6.93	64.65±0.31	83.42±1.12
Metal sheets	89.35±8.20	85.26±24.14	85.09±24.39	95.31±5.13	97.10±4.30	99.25±2.01	98.10±2.61	99.13±0.11
Bare soil	66.54±6.15	65.94±10.99	72.64±7.98	98.45±1.47	93.97±3.24	89.34±10.88	99.33±0.04	95.62±0.05
Bitumen	55.77±22.79	52.93±22.35	72.66±7.22	73.50±13.36	54.23±12.48	82.36±14.01	95.33±1.01	87.31±1.25
Bricks	73.39±7.31	94.29±3.25	91.72±3.30	93.24±3.25	88.77±5.64	79.64±18.68	97.82±1.09	98.39±4.52
Shadow	57.61±21.51	92.24±16.31	91.11±8.76	59.49±21.87	82.93±21.60	98.34±2.11	46.94±6.89	63.02±3.36

additional experiments with 100 and 300 training samples are also implemented. The detailed classification results are shown in Figs. 6 and 7. The proposed 1-D Auto-CNN exhibits the best classification results compared with other classical classification methods, which demonstrates the effectiveness of the proposed methods.

D. Classification Results for the 3-D Auto-CNNs

Here, we investigate the 3-D Auto-CNNs, which extract both spectral and spatial features, will lead to a better performance in terms of classification accuracy than that obtained by the 1-D Auto-CNNs. Moreover, due to the fact that the number of training samples is limited for HSI classification, a simple regularization method called cutout is here used in the 3-D AutoCNNs (3-D Auto-CNN-Cutout) to alleviate the overfitting problem and further boost the classification performance.

Since SVM and CNN have previously demonstrated, in HSI classification, accurate classification results, several spectral–spatial classification approaches based on SVM and CNN are here adopted for comparison. The SVM with the radial basis kernel function (3-D RBF-SVM) receives the spectral–spatial

TABLE VII
CLASSIFICATION WITH SPECTRAL–SPATIAL FEATURES ON THE KSC DATA SET USING 200 TRAINING SAMPLES

Method	3-D RBF-SVM	EMP-RF	EMP-SVM	EMP-CNN	3-D CNN	SSRN [74]	**3-D Auto-CNN**	**3-D Auto-CNN-Cutout**
OA(%)	90.12±1.40	94.06±1.40	93.41±1.15	96.56±1.24	95.98±0.85	96.77±0.83	96.90±0.90	**97.61±0.39**
AA(%)	83.95±2.22	89.48±3.11	90.31±1.89	94.65±2.48	94.07±1.39	94.70±0.86	96.29±1.19	**97.29±0.58**
K×100	89.00±1.56	93.37±1.56	92.66±1.28	96.18±1.38	95.53±0.95	96.41±0.93	96.54±1.10	**97.33±0.44**
Scrub	83.47±11.04	98.49±1.26	96.60±2.68	99.24±1.11	99.72±0.29	97.72±2.16	98.77±2.45	99.24±0.71
Willow swamp	66.16±21.30	80.95±10.67	83.65±10.95	86.81±10.09	82.18±9.58	90.35±9.95	88.80±3.21	91.20±6.91
CP hammock	78.27±22.26	88.83±11.21	83.01±7.37	91.25±8.89	92.80±7.60	95.88±2.52	86.78±1.16	94.19±5.53
Slash pine	66.87±20.04	86.16±12.13	84.27±8.73	87.92±10.60	78.38±7.19	91.68±5.97	89.18±3.25	92.38±4.67
Oak/Broadleaf	76.17±16.51	70.45±17.18	74.52±8.01	91.39±9.03	91.57±7.51	81.88±9.97	95.40±2.57	99.76±0.43
Hardwood	78.91±22.62	73.62±10.55	76.87±12.42	94.70±7.09	88.02±8.54	91.55±8.02	98.22±3.56	95.57±6.12
Swamp	53.30±30.70	75.31±26.62	96.09±2.94	87.29±9.32	96.64±7.55	89.91±10.76	98.63±2.75	100.00±0.00
Graminoid marsh	93.55±3.87	98.15±2.02	90.07±11.33	95.14±7.42	95.81±5.42	96.96±4.31	94.99±3.62	96.13±3.04
Spartina marsh	98.58±2.86	98.38±0.59	95.09±6.40	97.54±4.89	98.46±3.19	98.16±3.77	100.00±0.00	100.00±0.00
Cattail marsh	98.61±1.99	98.59±1.69	96.86±3.59	100.00±0.00	100.00±0.00	99.35±1.19	100.00±0.00	100.00±0.00
Salt marsh	99.97±0.08	99.67±1.00	98.96±1.90	99.95±0.16	100.00±0.00	98.28±2.83	100.00±0.00	100.00±0.00
Mud flats	97.53±2.94	95.10±3.74	98.03±3.09	99.29±1.39	99.33±1.07	99.37±1.63	99.23±0.08	96.43±5.32
Water	100.00±0.00	99.59±0.51	99.99±0.03	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00

TABLE VIII
CLASSIFICATION WITH SPECTRAL–SPATIAL FEATURES ON THE INDIAN PINES DATA SET USING 200 TRAINING SAMPLES

Method	3-D RBF-SVM	EMP-RF	EMP-SVM	EMP-CNN	3-D CNN	SSRN [74]	**3-D Auto-CNN**	**3-D Auto-CNN-Cutout**
OA(%)	79.75±1.89	85.10±1.59	81.28±1.68	87.22±1.37	85.29±1.45	80.66±2.91	87.79±1.42	**89.01±2.02**
AA(%)	60.11±2.56	**74.85±2.35**	74.34±5.31	74.77±5.50	69.94±4.83	65.35±6.85	72.15±6.00	72.88±8.22
K×100	75.82±2.35	82.96±1.84	78.61±1.92	85.03±1.62	82.70±1.75	77.83±3.37	85.73±1.64	**87.16±2.37**
Alfalfa	38.67±26.66	60.88±31.15	64.76±42.53	44.81±30.09	35.98±32.52	12.46±22.28	35.24±43.81	33.14±40.99
Corn-notill	80.01±9.17	78.10±7.99	76.63±5.52	87.81±7.01	82.49±6.84	79.05±10.59	85.80±4.88	88.28±3.21
Corn-min	52.50±11.83	83.43±11.56	77.40±6.25	73.58±11.10	64.88±14.94	73.92±16.97	76.16±16.34	79.86±14.67
Corn	23.85±22.71	59.23±16.02	62.55±23.71	64.95±37.09	45.67±20.51	38.96±21.69	53.60±36.17	56.57±29.64
Grass-pasture	61.00±16.10	74.90±10.75	77.95±11.24	71.72±21.07	58.91±16.72	75.46±16.00	73.48±19.75	71.13±16.66
Grass-trees	91.76±4.49	94.34±6.50	88.56±6.70	88.79±5.09	90.86±8.48	94.84±5.20	94.98±3.42	95.58±4.27
Grass-pasture-mowed	16.58±23.12	32.15±39.56	30.00±45.83	63.95±42.30	48.26±42.02	33.70±43.84	54.62±45.25	48.18±44.87
Hay-windrowed	87.89±8.56	99.85±0.17	96.70±6.34	96.50±3.02	94.54±7.28	99.53±0.69	99.53±0.62	99.77±0.46
Oats	42.21±37.64	5.26±15.79	59.47±48.58	20.00±40.00	24.29±31.89	8.48±13.13	10.67±21.33	20.00±40.00
Soybean-notill	74.57±8.69	74.18±10.67	76.59±6.93	88.60±4.97	86.89±4.65	75.52±8.84	91.98±2.32	91.74±3.80
Soybean-mintill	92.74±3.23	90.84±4.55	83.89±3.38	92.43±3.42	92.72±1.57	82.76±6.90	92.11±4.26	93.70±2.22
Soybean-clean	48.60±13.61	71.13±15.48	51.89±12.07	71.39±17.31	68.58±16.91	68.04±19.14	71.97±20.25	73.70±21.19
Wheat	61.59±25.50	99.50±0.90	87.48±29.21	91.08±7.05	92.99±13.36	95.41±8.34	93.64±5.28	94.75±4.65
Woods	94.26±2.15	97.29±1.35	96.91±2.45	95.60±4.29	98.89±1.28	96.38±3.15	97.15±1.41	98.20±0.96
Buildings-Grass-Trees	52.78±28.23	82.12±17.62	84.56±14.70	74.28±14.20	75.44±13.80	60.64±19.60	51.11±31.32	50.99±34.23
Stone-Steel-Towers	42.78±34.42	94.46±2.12	74.05±36.80	70.88±27.37	57.62±26.55	50.40±42.30	72.40±11.42	70.57±29.54

vector (i.e., pixel and its neighborhoods) as inputs. The extended MP (EMP) [20] is used to extract spatial information. In the EMP, the morphological opening and closing operations are used to extract spatial information on the first three principal components of HSIs, which are obtained by the PCA. In the experiments, the shape of the SE is set as a disk and the radius of the disk increases from 2 to 8 with the interval of 2. Therefore, 27 spatial features are generated. Then, the extracted robust spectral–spatial features are fed to some classical classifiers (i.e., RF-200 and RBF-SVM) to obtain the final classification results (EMP-RF and EMP-SVM). Furthermore, the CNN-based method is also implemented for

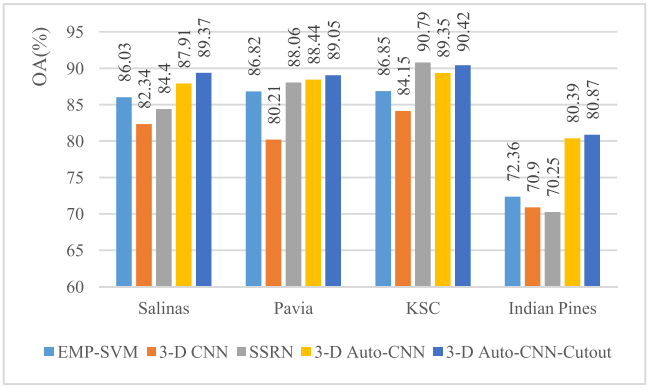


Fig. 8. Classification results of 3-D Auto-CNN and other state-of-the-art methods on four HSI data sets with 100 training samples.

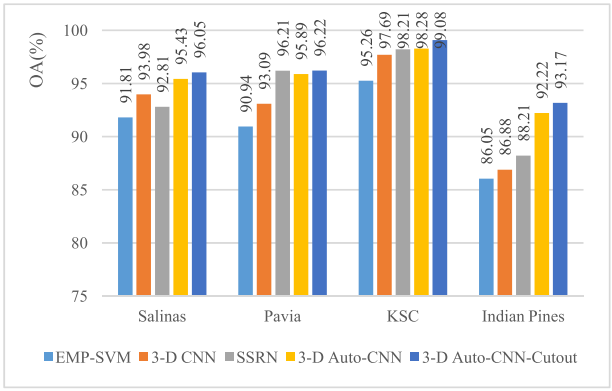


Fig. 9. Classification results of 3-D Auto-CNN and other state-of-the-art methods on four HSI data sets with 300 training samples.

comparison. The 3-D CNN and EMP-CNN are conducted and the architecture of CNN is designed according to [44]. Moreover, to provide a comprehensive comparison, a unique CNN architecture named spectral-spatial residual network (SSRN) recently proposed in [74] for HSIs classification is used for comparison.

From Tables V–VIII, one can see that for the Salinas data set, the 3-D Auto-CNN-Cutout improves the OA, AA, and K of EMP-CNN by 1.94%, 0.77%, and 0.0204, respectively. On the other hand, our 3-D Auto-CNN approach also performs better than EMP-CNN in terms of OA, AA, and K (see Table V). For the Pavia data set, the OA of 3-D Auto-CNN-Cutout is increased by 2.09% and 2.23% compared with 3-D CNN and spectral-spatial residual network (SSRN) (see Table VI). For the KSC data set, both the 3-D Auto-CNN and 3-D Auto-CNN-Cutout show higher classification accuracies than other classical methods. The 3-D Auto-CNN-Cutout exhibits the best OA, AA, and K, with the improvement of 0.84%, 2.59%, and 0.0092 over SSRN, respectively (see Table VII). For the Indian Pines data set, the similar results to the other data sets are obtained (see Table VIII). The detailed classification results are shown in Tables V–VIII. Moreover, experiments with 100 and 300 training samples are investigated as well to evaluate the robustness of the proposed methods. The classification results are illustrated in Figs. 8 and 9. Compared with other state-of-the-art methods, the 3-D Auto-CNN-Cutout demonstrates the best performance

TABLE IX
TOTAL NUMBER OF TRAINABLE PARAMETERS IN DIFFERENT CNN-BASED MODELS

Methods / Data sets	3-D CNN	SSRN	3-D Auto-CNN-Cutout
Salinas	193.5K	370.3K	76.7K
Pavia	141.3K	216.5K	83.8K
KSC	178.9K	327.2K	61.6K
Indian	191.4K	364.2K	103.5K

under a different number of training samples. Furthermore, for all four data sets, the 3-D Auto-CNN-Cutout shows a better classification accuracy than the 3-D Auto-CNN, which shows the effectiveness of using the cutout.

E. Optimal Architecture

As illustrated in Algorithms 1 and 2, there are two major steps, i.e., architecture search and architecture evaluation, in both 1-D Auto-CNN and 3-D Auto-CNN. The learning curves (i.e., loss and accuracy of training and validation data sets) of the architecture search process with 200 training samples and 100 validation samples are shown in Fig. 10. The architecture variable $\boldsymbol{\alpha}$ that determines the CNN architecture is optimized along the search process.

The training epoch in the architecture search process for the 1-D Auto-CNN and 3-D Auto-CNN-Cutout is set to 300 and 100. The experiments are conducted on a personal computer equipped with an Intel Core i5-7300H processor with 2.5 GHz, 8 GB of DDR4 RAM, an NVIDIA GeForce GTX 1050Ti GPU. Regarding our software environment, it is composed by Windows 10 as an operating system, CUDA 9.0 and cuDNN 7.1, and Pytorch framework and with Python 3.6 as the programing language. The learning curves of the architecture search process in Auto-CNN on the four HSI data sets are shown in Fig. 10. The architecture search process, which is a significant time-demanding step in the automatic CNN, only takes 12 and 14 min in 1-D Auto-CNN and 3-D Auto-CNN-Cutout to find the optimal CNN architecture.

At the end of the architecture search, the variable $\boldsymbol{\alpha}$ is selected according to the classification performance on the validation data set. Then, the optimal architecture (i.e., the architecture of normal cell and reduction cell) is obtained. Finally, the 1-D Auto-CNN and 3-D Auto-CNN are trained from scratch on the training data set, and we evaluate the classification accuracy of the network on the test data set. Here, we take the Pavia data set and the Indian Pines data set as examples for 1-D Auto-CNN and 3-D Auto-CNN respectively. Figs. 11–14 show the detailed architecture of each cell. From the figure, one can see that each cell consists of four nodes. The input to the cell is the previous two output cells and the output of the cell is feature concatenation of four nodes.

F. Number of Trainable Parameters

The total number of trainable parameters is an important aspect of deep learning-based models. Here, we discussed the number of trainable parameters in different CNN-based models. The detailed results were reported in Table IX. Due to

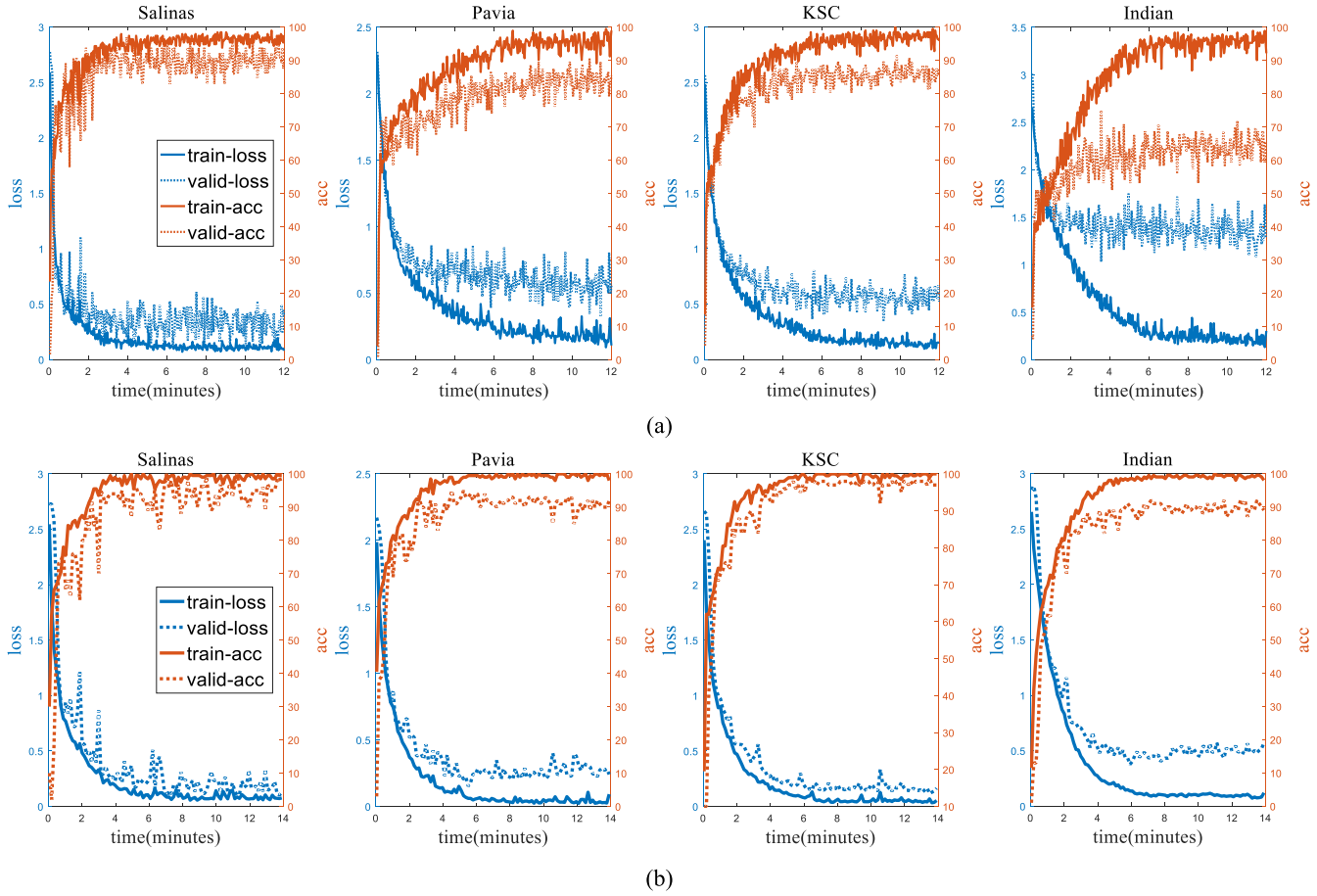


Fig. 10. Learning curves of the architecture search process in Auto-CNN on the four HSI data sets. (a) 1-D Auto-CNN. (b) 3-D Auto-CNN-Cutout.

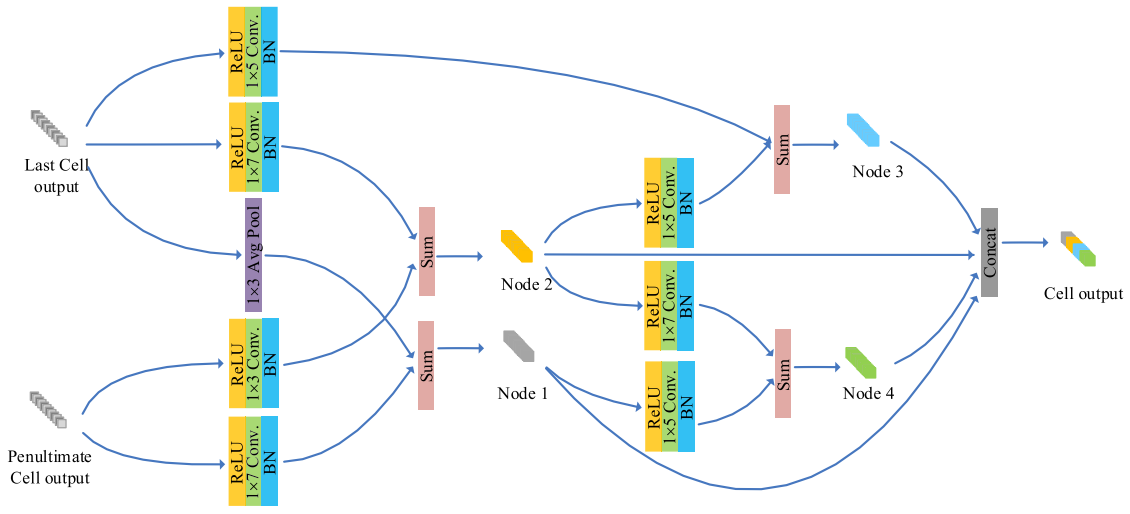


Fig. 11. Optimal architecture of normal cell in 1-D Auto-CNN for the Pavia data set.

the fact that the architecture of CNN in 3-D Auto-CNN-Cutout was automatically designed, the optimal CNN architectures were different in each run (experiments in this paper were run ten times). We picked one optimal architecture and calculated the number of trainable parameters.

From Table IX, it is easy to observe that the number of trainable parameters in the proposed 3-D Auto-CNN-Cutout is significantly fewer than the other studied methods (i.e., 3-D CNN and SSRN). There are mainly two reasons why 3-D Auto-CNN-Cutout achieved a superior classification performance with a smaller number of trainable parameters. First, a model with a large number of trainable parameters tends to overfit when the number of training samples is limited. Second, the architecture of our 3-D Auto-CNN-Cutout

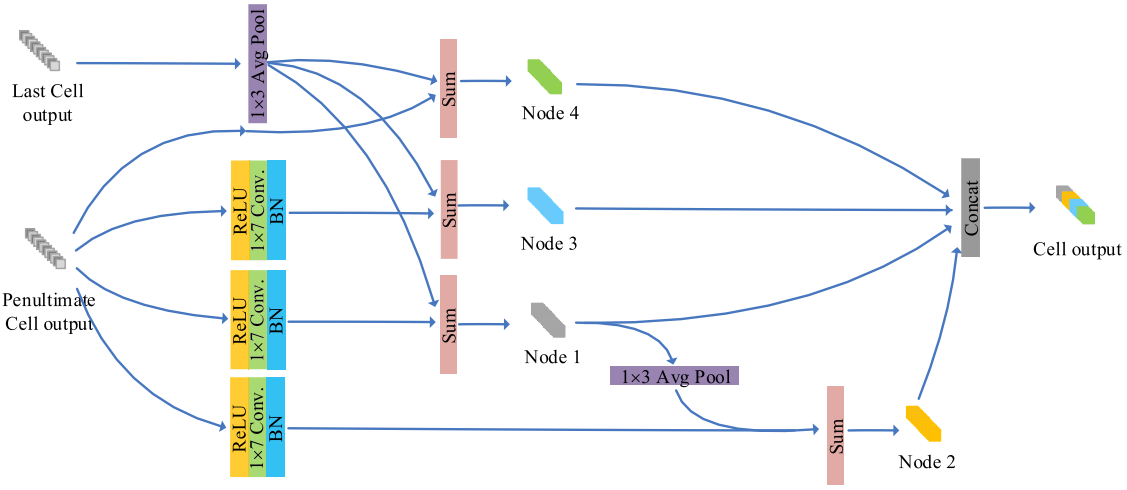


Fig. 12. Optimal architecture of reduction cell in 1-D Auto-CNN for the Pavia data set.

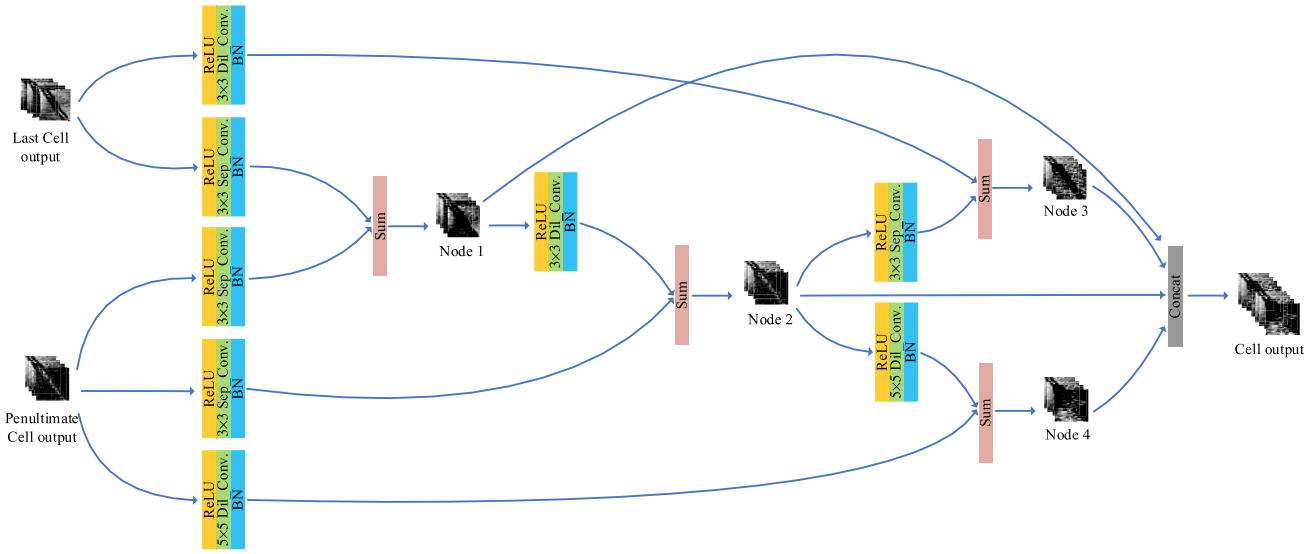


Fig. 13. Optimal architecture of normal cell in 3-D Auto-CNN-Cutout for the Indian Pines data set.

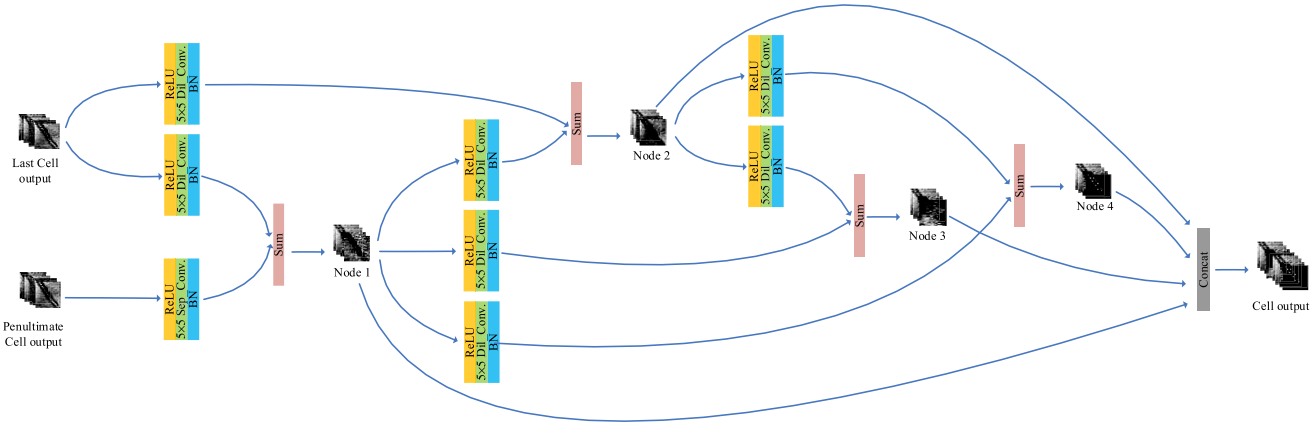


Fig. 14. Optimal architecture of reduction cell in 3-D Auto-CNN-Cutout for the Indian Pines data set.

is automatically designed based on the training and validation samples. The data-dependent CNN architecture is suitable for the specific HSI data set and further boosts the classification accuracy.

G. Classification Maps

In this section, we evaluate the classification accuracies from a visual perspective. The trained models, including RBF-SVM, 1-D Auto-CNN, 3-D CNN, SSRN, and 3-D

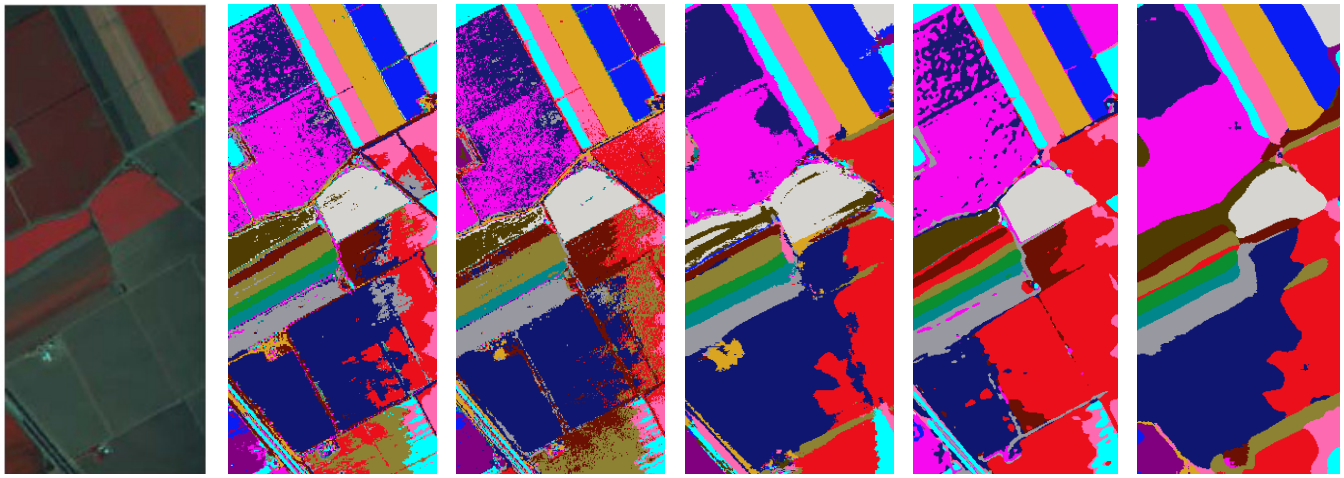


Fig. 15. Salinas. (a) False color image. Classification maps of different classifiers. (b) RBF-SVM. (c) 1-D Auto-CNN. (d) 3-D CNN. (e) SSRN. (f) 3-D Auto-CNN-Cutout.

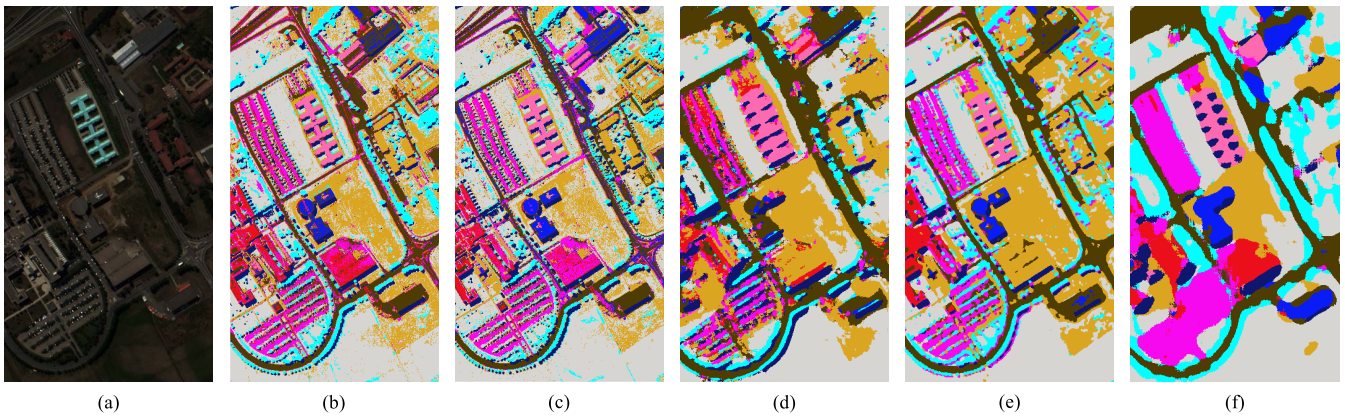


Fig. 16. Pavia. (a) False color image. Classification maps of different classifiers. (b) RBF-SVM. (c) 1-D Auto-CNN. (d) 3-D CNN. (e) SSRN. (f) 3-D Auto-CNN-Cutout.

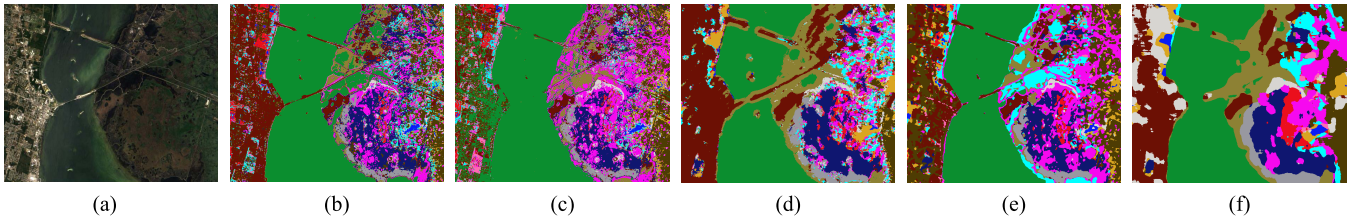


Fig. 17. KSC. (a) False color image. Classification maps of different classifiers. (b) RBF-SVM. (c) 1-D Auto-CNN. (d) 3-D CNN. (e) SSRN. (f) 3-D Auto-CNN-Cutout.

Auto-CNN-Cutout, are selected to classify the whole images. All parameters in these models are optimized. The classification maps obtained by different models using the four data sets are shown in Figs. 15–18. From the resulting images, we can figure out how the different classification methods affect the classification results. It is obvious that spectral-based models, which only utilize spectral features, always result in noisy scatter points in the classification map and depict more errors compared with spectral–spatial-based methods for the four data sets [see Figs. 15(b) and (c), 16(b) and (c), 17(b) and (c), and 18(b) and (c)]. Spectral–spatial methods overcome this shortcoming. Obviously, 3-D CNN, SSRN, and 3-D Auto-CNN-Cutout directly use the neighbor information as the model input, resulting in smoother classification maps [see Figs. 15(d)–(f), 16(d)–(f), 17(d)–(f), and 18(d)–(f)]. By comparing the true ground reference with the classification maps, one can see that the 3-D Auto-CNN-Cutout can obtain more precise classification results, which demonstrates the effectiveness of the autodesign neural network for HSI classification.

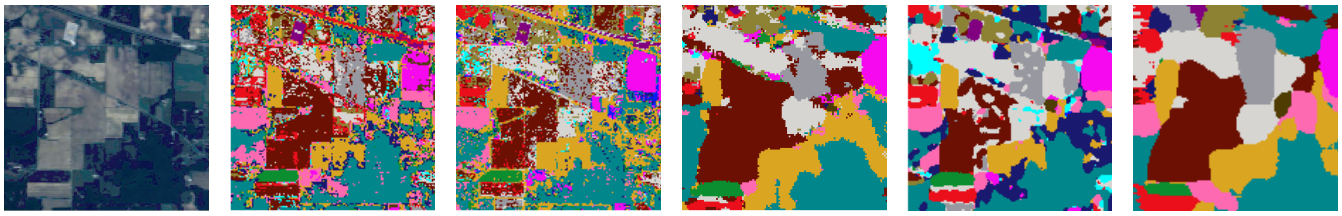


Fig. 18. Indian Pines. (a) False color image. Classification maps of different classifiers. (b) RBF-SVM. (c) 1-D Auto-CNN. (d) 3-D CNN. (e) SSRN. (f) 3-D Auto-CNN-Cutout.

V. Conclusion

In this paper, the automatic design of CNN for HSI classification is proposed for the first time. The automatically designed CNNs achieve better classification accuracies compared with the state-of-the-art deep learning models, including CNN, RNN, 3-D CNN, and SSRN, which were designed by human experts.

Specifically, two automatic HSI classification frameworks, the 1-D Auto-CNN and 3-D Auto-CNN are proposed as spectral and spectral–spatial HSI classifiers, respectively. An optimization technique, which is based on gradient descent, is used to determine the optimal architecture in the search space. The two selected architectures demonstrate the excellent abilities for the task of HSI classification. Furthermore, a modified cutout approach is used in the 3-D Auto-CNN and it mitigates the overfitting phenomenon and improves the classification performance.

Experiments are carried out on four popular hyperspectral data sets with limited training samples (e.g., amount to 100 training samples for each data set). The Auto-CNNs (i.e., 1-D Auto-CNN and 3-D Auto-CNN) always obtain better classification results.

The optimization of CNN architecture takes time (e.g., tens of minutes), but it does not need a human effort. Moreover, the designed CNN fits a specific data set automatically, and more importantly, it opens a new research path for future research, showing that the autodesign of deep learning models has a huge potential for accurate HSI classification.

References

[1] G. Camps-Valls, D. Tuia, L. Bruzzone, and J. A. Benediktsson, "Advances in hyperspectral image classification: Earth monitoring with statistical learning methods," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 45–54, Jan. 2014.

[2] J. A. Richards and X. Jia, *Remote Sensing Digital Image Analysis*. New York, NY, USA: Springer-Verlag, 1999.

[3] C.-I. Chang, *Hyperspectral Data Exploitation: Theory and Applications*. Hoboken, NJ, USA: Wiley, 2007.

[4] P. Ghamisi, J. Plaza, Y. Chen, J. Li, and A. J. Plaza, "Advanced spectral classifiers for hyperspectral images: A review," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 1, pp. 8–32, Mar. 2017.

[5] P. Ghamisi *et al.*, "New frontiers in spectral-spatial hyperspectral image classification: The latest advances based on mathematical morphology, Markov random fields, segmentation, sparse representation, and deep learning," *IEEE Geosci. Remote Sens. Mag.*, vol. 6, no. 3, pp. 10–43, Sep. 2018.

[6] X. Jia, B.-C. Kuo, and M. M. Crawford, "Feature mining for hyperspectral image classification," *Proc. IEEE*, vol. 101, no. 3, pp. 676–697, Mar. 2013.

[7] G. Licciardi, P. R. Marpu, J. Chanussot, and J. A. Benediktsson, "Linear versus nonlinear PCA for the classification of hyperspectral data based on the extended morphological profiles," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 3, pp. 447–451, May 2011.

[8] T. M. Tu, "Unsupervised signature extraction and separation in hyperspectral images: A noise-adjusted fast independent component analysis approach," *Opt. Eng.*, vol. 39, no. 4, pp. 897–906, Apr. 2000.

[9] T. V. Bandos, L. Bruzzone, and G. Camps-Valls, "Classification of hyperspectral images with regularized linear discriminant analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 3, pp. 862–873, Mar. 2009.

[10] H. Yang *et al.*, "LLE-PLS nonlinear modeling method for near infrared spectroscopy and its application," *Spectrosc. Spectral Anal.*, vol. 27, no. 10, pp. 1955–1958, Oct. 2007.

[11] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[12] J. Ledolter, "Multinomial logistic regression," in *Data Mining and Business Analytics With R*. Hoboken, NJ, USA: Wiley, 2013, pp. 109–124.

[13] S. Delalieux, B. Somers, B. Haest, T. Spanhove, J. V. Borre, and C. A. Mücher, "Heathland conservation status mapping through integration of hyperspectral mixture analysis and decision tree classifiers," *Remote Sens. Environ.*, vol. 126, pp. 222–231, Nov. 2012.

[14] B. Waske, S. van der Linden, J. Benediktsson, A. Rabe, and P. Hostert, "Sensitivity of support vector machines to random feature selection in classification of hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 7, pp. 2880–2889, Jul. 2010.

[15] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[16] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[17] L. Zhang, Y. Zhong, B. Huang, J. Gong, and P. Li, "Dimensionality reduction based on clonal selection for hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 12, pp. 4172–4186, Dec. 2007.

[18] X. Lu, B. Wang, X. Zheng, and X. Li, "Exploring models and data for remote sensing image caption generation," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 2183–2195, Apr. 2018.

[19] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral-spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 2, pp. 844–856, Feb. 2013.

[20] 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.*, vol. 43, no. 3, pp. 480–491, Mar. 2005.

[21] R. Bellens, S. Gautama, L. Martinez-Fonte, W. Philips, J. C.-W. Chan, and F. Canters, "Improved classification of VHR images of urban areas using directional morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 10, pp. 2803–2813, Oct. 2008.

[22] W. Liao *et al.*, "Taking optimal advantage of fine spatial resolution: Promoting partial image reconstruction for the morphological analysis of very-high-resolution images," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 2, pp. 8–28, Jun. 2017.

[23] A. Samat, C. Persello, S. Liu, E. Li, Z. Miao, and J. Abuduwaili, "Classification of VHR multispectral images using extratrees and maximally stable extremal region-guided morphological profile," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 9, pp. 3179–3195, Sep. 2018.

[24] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone, "Morphological attribute profiles for the analysis of very high resolution images," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 10, pp. 3747–3762, Oct. 2010.

[25] P. Ghamisi, M. D. Mura, and J. A. Benediktsson, "A survey on spectral-spatial classification techniques based on attribute profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2335–2353, May 2015.

[26] P. Ghamisi, R. Souza, J. A. Benediktsson, X. X. Zhu, L. Rittner, and R. A. Lotufo, "Extinction profiles for the classification of remote sensing data," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 5631–5645, Oct. 2016.

[27] L. Fang, N. He, S. Li, P. Ghamisi, and J. A. Benediktsson, "Extinction profiles fusion for hyperspectral images classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 3, pp. 1803–1815, Mar. 2018.

[28] Y. Gu, T. Liu, X. Jia, and J. A. Benediktsson, "Nonlinear multiple kernel learning with multiple-structure-element extended morphological profiles for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3235–3247, Jun. 2016.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[31] J. Long, E. Shelhamer, and T. Darrell. (Mar. 2015). "Fully convolutional networks for semantic segmentation." [Online]. Available: https://arxiv.org/abs/1411.4038

[32] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *Proc. Interspeech*, Aug. 2013, pp. 2524–2528.

[33] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.

[34] 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.

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

[36] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, Jul. 2017.

[37] X. Ma, H. Wang, and J. Geng, "Spectral-spatial classification of hyperspectral image based on deep auto-encoder," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 9, pp. 4073–4085, Sep. 2016.

[38] Y. Chen, X. Zhao, and X. Jia, "Spectral-spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2381–2392, Jun. 2015.

[39] W. Li, G. Wu, F. Zhang, and Q. Du, "Hyperspectral image classification using deep pixel-pair features," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 2, pp. 844–853, Feb. 2017.

[40] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016.

[41] J. Yue, W. Zhao, S. Mao, and H. Liu, "Spectral–spatial classification of hyperspectral images using deep convolutional neural networks," *Remote Sens. Lett.*, vol. 6, no. 6, pp. 468–477, May 2015.

[42] H. Zhang, Y. Li, Y. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery using a dual-channel convolutional neural network," *Remote Sens. Lett.*, vol. 8, no. 5, pp. 438–447, May 2017.

[43] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 120–147, Nov. 2018.

[44] 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.

[45] Y. Li, H. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery with 3D convolutional neural network," *Remote Sens.*, vol. 9, no. 1, p. 67, 2017

[46] L. Mou, P. Ghamisi, and X. X. Zhu, "Unsupervised spectral–spatial feature learning via deep residual Conv–Deconv network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 1, pp. 391–406, Jan. 2018.

[47] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "Deep&Dense convolutional neural network for hyperspectral image classification," *Remote Sens.*, vol. 10, no. 9, p. 1454, 2018.

[48] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3973–3985, Oct. 2011.

[49] H. Liang and Q. Li, "Hyperspectral imagery classification using sparse representations of convolutional neural network features," *Remote Sens.*, vol. 8, no. 2, p. 99, 2016.

[50] E. Aptoula, M. C. Ozdemir, and B. Yanikoglu, "Deep learning with attribute profiles for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1970–1974, Dec. 2016.

[51] W. Shao and S. Du, "Spectral-spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4544–4554, Oct. 2016.

[52] J. Kukačka, V. Golkov, and D. Cremers. (Oct. 2017). "Regularization for deep learning: A taxonomy." [Online]. Available: https://arxiv.org/abs/1710.10686

[53] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. (Jan. 2019). "A survey of the recent architectures of deep convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1901.06032

[54] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, Feb. 2015, pp. 448–456.

[55] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *Comput. Sci.*, vol. 3, no. 4, pp. 212–223, Jul. 2012.

[56] T. DeVries and G. W. Taylor. (Aug. 2017). "Improved regularization of convolutional neural networks with cutout." [Online]. Available: https://arxiv.org/abs/1708.04552

[57] T. Elsken, J. H. Metzen, and F. Hutter. (Aug. 2018). "Neural architecture search: A survey." [Online]. Available: https://arxiv.org/abs/1808.05377

[58] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter. (Dec. 2016). "Towards automatically-tuned neural networks." [Online]. Available: http://proceedings.mlr.press/v64/mendoza_towards_2016.html

[59] B. Zoph and Q. V. Le. (Nov. 2016). "Neural architecture search with reinforcement learning." [Online]. Available: https://arxiv.org/abs/1611.01578

[60] E. Real *et al.* (Mar. 2017). "Large-scale evolution of image classifiers." [Online]. Available: https://arxiv.org/abs/1703.01041

[61] C. Liu *et al.* (Dec. 2017). "Progressive neural architecture search." [Online]. Available: https://arxiv.org/abs/1712.00559

[62] H. Liu, K. Simonyan, and Y. Yang. (Jun. 2018). "DARTS: Differentiable architecture search." [Online]. Available: https://arxiv.org/abs/1806.09055

[63] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. (Jul. 2017). "Learning transferable architectures for scalable image recognition." [Online]. Available: https://arxiv.org/abs/1707.07012

[64] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. (Feb. 2018). "Regularized evolution for image classifier architecture search." [Online]. Available: https://arxiv.org/abs/1802.01548

[65] X. Zhang, Z. Huang, and N. Wang. (Nov. 2018). "You only search once: Single shot neural architecture search via direct sparse optimization." [Online]. Available: https://arxiv.org/abs/1811.01567

[66] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei. (Jan. 2019). "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation." [Online]. Available: https://arxiv.org/abs/1901.02985

[67] N. Kruger *et al.*, "Deep hierarchies in the primate visual cortex: What can we learn for computer vision?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1847–1871, Aug. 2013.

[68] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[69] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[70] S. Zagoruyko and N. Komodakis. (May 2016). "Wide residual networks." [Online]. Available: https://arxiv.org/abs/1605.07146

[71] F. Chollet. (Apr. 2017). "Xception: Deep Learning with depthwise separable convolutions." [Online]. Available: https://arxiv.org/abs/1610.02357

[72] F. Yu and V. Koltun. (Nov. 2015). "Multi-scale context aggregation by dilated convolutions." [Online]. Available: https://arxiv.org/abs/1511.07122
[73] D. P. Kingma and J. Ba. (Dec. 2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[74] Z. Zhong, J. Li, Z. Luo, and M. Chapman, "Spectral-spatial residual network for hyperspectral image classification: A 3-D deep learning framework," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 847–858, Feb. 2018.

Yushi Chen (M'11) received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2008.

He is currently an Associate Professor with the School of Electronics and Information Engineering, Harbin Institute of Technology. His research interests include remote sensing data processing and machine learning.

Kaiqiang Zhu received the B.S. degree from the Harbin Institute of Technology, Harbin, China, in 2017, where he is currently pursuing the M.S. degree with the School of Electronics and Information Engineering.

His research interests include remote sensing image classification, machine learning, and deep learning.

Lin Zhu is currently pursuing the master's degree with the Department of Information Engineering, School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin, China.

Her research interests include hyperspectral image classification, machine learning, and deep learning.

Xin He received the M.S. degree from the Harbin University of Science and Technology, Harbin, China. She is currently pursuing the Ph.D. degree with the School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin.

Her research interest includes remote sensing image processing based on deep learning methods.

Pedram Ghamisi (S'12–M'15–SM'18) is currently the Head of the Machine Learning Group, Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany. His research interests include interdisciplinary research on remote sensing and machine (deep) learning, image, signal processing, and multisensory data fusion.

Jón Atli Benediktsson (S'84–M'90–SM'99–F'04) received the Cand.Sci. degree in electrical engineering from the University of Iceland, Reykjavik, Iceland, in 1984, and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1987 and 1990, respectively.

He is the President and Rector of the University of Iceland, where he is also a Professor of electrical and computer engineering. He is a Co-Founder of Oxymap, biomedical start-up company. His research interests include remote sensing, biomedical analysis of signals, pattern recognition, image processing, and signal processing. He has published extensively in those fields.

Dr. Benediktsson was a member of the 2014 IEEE Fellow Committee, the Association of Chartered Engineers in Iceland (VFI), Societas Scientiarum Islandica, and Tau Beta Pi and a fellow of the SPIE. He was a recipient of the Icelandic Research Council's Outstanding Young Researcher Award in 1997 and the IEEE Third Millennium Medal in 2000. He received the Stevan J. Kristof Award from Purdue University as an outstanding graduate student in remote sensing, in 1991, the Yearly Research Award from the Engineering Research Institute of the University of Iceland in 2006, the Outstanding Service Award from the IEEE Geoscience and Remote Sensing Society in 2007, the IEEE/VFI Electrical Engineer of the Year Award in 2013, and the OECE Award from the School of Electrical and Computer Engineering, Purdue University, in 2016. He was co-recipient of the University of Iceland's Technology Innovation Award in 2004, the 2012 IEEE Transactions on Geoscience and Remote Sensing Paper Award, the IEEE Geoscience and Remote Sensing Society (GRSS) Highest Impact Paper Award in 2013, and the *International Journal of Image and Data Fusion* Best Paper Award in 2014. He was the Chairman of the Steering Committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (J-STARS) from 2007 to 2010. He was the President of the IEEE GRSS from 2011 to 2012. He has been on the GRSS AdCom since 2000. He is on the Editorial Board of the PROCEEDINGS OF THE IEEE, the International Editorial Board of the *International Journal of Image and Data Fusion*, and the Editorial Board of *Remote Sensi*ng. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING (TGRS) from 2003 to 2008. He has been serving as an Associate Editor for TGRS since 1999, the *IEEE Geoscience and Remote Sensing Letters* since 2003, and the IEEE ACCESS since 2013.