Learning and Transferring Deep Joint Spectral–Spatial Features for Hyperspectral Classification

Article *in* IEEE Transactions on Geoscience and Remote Sensing · May 2017

DOI: 10.1109/TGRS.2017.2698503

3 authors, including:

Jingxiang Yang
Nanjing University of Science and Technology
31 PUBLICATIONS **1,770** CITATIONS

SEE PROFILE

Yongqiang Zhao
Northwestern Polytechnical University
194 PUBLICATIONS **3,791** CITATIONS

SEE PROFILE

Learning and Transferring Deep Joint Spectral–Spatial Features for Hyperspectral Classification

Jingxiang Yang, *Student Member, IEEE*, Yong-Qiang Zhao, *Member, IEEE*, and Jonathan Cheung-Wai Chan

***Abstract*— Feature extraction is of significance for hyperspectral image (HSI) classification. Compared with conventional hand-crafted feature extraction, deep learning can automatically learn features with discriminative information. However, two issues exist in applying deep learning to HSIs. One issue is how to jointly extract spectral features and spatial features, and the other one is how to train the deep model when training samples are scarce. In this paper, a deep convolutional neural network with two-branch architecture is proposed to extract the joint spectral–spatial features from HSIs. The two branches of the proposed network are devoted to features from the spectral domain as well as the spatial domain. The learned spectral features and spatial features are then concatenated and fed to fully connected layers to extract the joint spectral–spatial features for classification. When the training samples are limited, we investigate the transfer learning to improve the performance. Low and mid-layers of the network are pretrained and transferred from other data sources; only top layers are trained with limited training samples extracted from the target scene. Experiments on Airborne Visible/Infrared Imaging Spectrometer and Reflective Optics System Imaging Spectrometer data demonstrate that the learned deep joint spectral–spatial features are discriminative, and competitive classification results can be achieved when compared with state-of-the-art methods. The experiments also reveal that the transferred features boost the classification performance.**

***Index Terms*— Convolutional neural network (CNN), deeplearning, feature extraction, hyperspectral classification, transfer learning.**

Manuscript received August 30, 2016; revised February 21, 2017; accepted April 14, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61371152, Grant 61071172, and Grant 61374162, in part by the National Natural Science Foundation of China and South Korean National Research Foundation Joint Funded Cooperation Program under Grant 61511140292, in part by the New Century Excellent Talents Award Program from the Ministry of Education of China under Grant NCET-12-0464, in part by the Ministry of Education Scientific Research Foundation for the Returned Overseas, in part by the Fundamental Research Funds for the Central Universities under Grant 3102015ZY045, in part by the China Scholarship Council for joint Ph.D. students under Grant 201506290120, and in part by the Innovation Foundation of Doctor Dissertation of Northwestern Polytechnical University under Grant CX201621. *(Corresponding author: Yong-Qiang Zhao.)*

J. Yang is with the Key Laboratory of Information Fusion Technology (Ministry of Education of China), School of Automation, Northwestern Polytechnical University, Xi'an 710072, China, and also with the Department of Electronics and Informatics, Vrije Universiteit Brussel, 1050 Brussel, Belgium (e-mail: yang123jx@mail.nwpu.edu.cn).

Y.-Q. Zhao is with the Key Laboratory of Information Fusion Technology (Ministry of Education of China), School of Automation, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: zhaoyq@nwpu.edu.cn).

J. C.-W. Chan is with the Department of Electronics and Informatics, Vrije Universiteit Brussel, 1050 Brussel, Belgium (e-mail: jcheungw@etro.vub.ac.be).

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

Digital Object Identifier 10.1109/TGRS.2017.2698503

I. Introduction

HYPERSPECTRAL imagery (HSI) contains rich spectral information and spatial information, which is beneficial to discriminate different materials in the observed scene. Due to the discriminative ability, HSI has been applied to many fields, such as agriculture [1], environment management [2], and mineral exploitation [3]. Land cover classification is one of the important applications of hyperspectral remote sensing.

In the past decade, different kinds of hyperspectral classification methods were proposed. Most of the traditional hyperspectral classification methods rely on hand-crafted features [4]–[15]. Different hand-crafted features are designed and then fed to classifiers such as support vector machine (SVM) or sparse representation classifier [7], [8]. For example, mathematical morphology can exploit the spatial structure information of HSI. Morphological profile (MP) features were first proposed for the HSI classification in [4]. The MP features can also be combined with spectrum through linearly concatenation [5] or composite kernel learning [6]. Other than morphology, spatial structure information of HSI can also be extracted by multiscale analysis, such as 3-D wavelet [9] and 3-D Gabor filtering [10]. Different types of features complementary in describing the scene can be integrated to improve the classification performance. Zhang *et al.* [12] unified spectrum, Gabor features, and pixel-shaped features, and then learned a low-dimensional representation of the unified features via manifold learning. In [13], a multiple feature learning framework was proposed to combine the spectrum, MP features, and nonlinear kernel-based features. HSI can be naturally represented as a tensor, extracting features and performing classification in multilinear algebra framework, is an another option [15]. In [16], feature tensor was extracted via multilinear principal component analysis (PCA) and then fed to a support tensor machine for classification.

The above classification methods can be regarded as shallow models with the hand-crafted features (e.g., MP features or spectrum), exploiting only low-level information [17], [18]. Compared with shallow models, deep learning model has a hierarchical architecture, in which both low-level and high-level features can be learned automatically from the data [19]. In the deep learning framework, low-level features can be extracted by the bottom layers, and then fed to top layers to generate high-level features, which reveal information that is more abstract and discriminative than that of low-level features [20]. Furthermore, compared with the low-level hand-crafted features, the high-level features are more robust and

invariant to deal with the challenge of decreased interclass variability and increased intraclass variability [18], [25], [26], so the classification performance would be boosted by employing the high-level features.

Typical deep learning model includes stacked autoencoders (SAEs) [21], convolutional neural network (CNN) [22], and restricted Boltzmann machines (RBMs) [20]. Some deep learning-based HSI classification algorithms have been proposed recently [24]–[31]. Deep spectral–spatial joint features of HSI were learned by combining SAE and PCA in [24]. SAE could also be used to extract multiscale spatial features and spectral features for HSI classification [27]. Similar to the network in [24], RBM was combined with PCA to learn spectral–spatial joint features for HSI classification in [28] and [35]. However, as both SAE and RBM are originally designed for 1-D signal, HSI had to be vectorized for appropriate uses, which may lead to missing spatial correlation. In order to exploit the spatial correlation of HSI, a spatial regularizer that connected neighboring pixels with similar features was introduced to the objective function of SAE in [29]. CNN can naturally extract image spatial features without any vectorization. An unsupervised CNN model was built for the feature extraction of HSI through greedy layer-wise training in [17]. Spatial features were extracted by CNN and directly fed to a sparse representation classifier in [30]. In [26], CNN was used to extract spatial features from HSI, and manifold learning was used to extract spectral features. The spatial features and the spectral features were then stacked to form a spectral–spatial joint feature vector. Similarly, multiscale spatial features were extracted by applying CNN to the Laplacian pyramid of HSI, and then stacked with dimension-reduced spectrum to generate a spectral–spatial joint feature vector in [31]. It is worth noting that compared with the unsupervised CNN in [17], the supervised CNN could learn features that are more discriminative by exploiting the class-specific information from the training data. The supervised CNN in [26] and [31] learned spectral features and spatial features separately, as such correlation between the spectral and spatial domains would not be exploited. Furthermore, the spectral features in [26] and [31] were obtained by shallow dimensionality reduction features that are more abstract and discriminative would be obtained if deep model is applied to the spectral domain.

In order to exploit the correlation between the spectral and spatial domains, it is better to learn the spectral features and spatial features jointly in an end-to-end framework. In this paper, we propose a two-branche deep CNN (Two-CNN) to learn the joint spectral–spatial features for the classification. The proposed deep learning model is composed of two branches of CNN, extracting information from both the spectral and spatial domains. Spectrum of each pixel and its neighboring pixels are used as input for the two branches. The spectral features and the spatial features extracted by the two branches of CNN are then concatenated and fed to the fully connected layers to learn higher-level joint spectral–spatial features. Finally, a softmax layer is used to predict the class.

It is always expensive to obtain a large number of labeled samples for classification purposes. If the labeled training samples are limited, overfitting may happen. The performance of classification could be satisfying on the training data but poor on the testing data. In order to achieve a good performance in cases with scarce training samples, we propose to train the Two-CNN through the transfer learning strategy. Transfer learning is motivated by the fact that humans intelligently apply their previous knowledge to solve new problems [36]. Useful information from other related data is extracted and transferred to target task in the transfer learning framework [40], [52]. It has been successfully applied in many fields where sufficient training samples are not available [37], [38]. Transfer learning could boost the Two-CNN as well, especially when the training samples are scarce. Because the bottom and mid-layers of deep learning network are generic, they extract the low-level features that can be utilized by different tasks. Transferring the low and mid-layers to the new task would reduce the requirement for training samples and result in fast training [39], [45]. Recently, transferring pretrained deep learning network has shown its potential in remote-sensed scene classification [41]–[43]. In this paper, we test the use of a pretrained Two-CNN with sufficient training samples from other hyperspectral scenes, and transfer the pretrained layers to the network for the target HSI. This transfer learning approach will further improve the performance of Two-CNN even in the scarce training sample case.

We consider four main contributions of this paper. First, we propose a two-branch CNN to automatically learn the deep joint spectral–spatial features from HSI. Compared with hand-crafted features, the deep joint spectral–spatial features are more discriminative. Second, we propose to train the deep model through transfer learning, which boosts the robustness of the proposed method, especially in scarce training sample cases. Third, the spectral–spatial joint features are trained for classification task in an end-to-end framework, which means that the extracted spectral information and spatial information are complementary to each other, and the final joint spectral–spatial features are more suitable for the following classification task. Finally, we analyze the discrimination of the learned features by the feature visualization, in terms of the transferability of features, and we also analyze the optimal architecture for transfer learning of Two-CNN.

This paper is organized as follows. In Section II, we introduce the basics of CNN. In Section III, we present the architecture and training strategy of the proposed Two-CNN model. Experimental results and discussions are given in Section IV. The conclusions are drawn in Section V.

II. Basics Of CNN

As shown in Fig. 1, a typical CNN architecture is alternatively stacked by convolutional layers and pooling layers, and then followed by fully connected layers. Each layer is cascaded with its previous layer; in this way, the feature extracted by the higher layer is an abstraction of the lower layer. Usually, the more of the layers, the more discriminative and semantic information is contained in the features [22], [23]. The convolutional layer computes convolution of the input feature maps with convolutional kernels (also called filters),

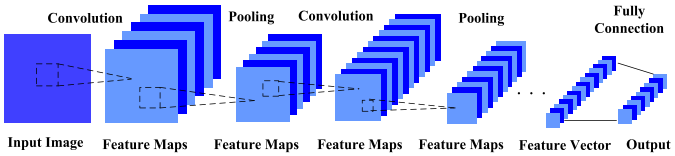


Fig. 1. Typical architecture of CNN.

followed by elementwise nonlinear activation. Activity of the ith feature map in the lth layer is [32]

$$F_i^l = \sum_j g(w_{i,j}^l * F_j^{l-1} + b_i^l) \tag{1}$$

where $\boldsymbol{F}_j^{l-1} \in \mathbb{R}^{p\times q}$ is the jth feature map in the $(l-1)$th layer that connects to the feature map $\boldsymbol{F}_i^l$ in the lth convolutional layer. $\boldsymbol{w}_{i,j}^l \in \mathbb{R}^{w\times w}$ is the convolutional kernel for $\boldsymbol{F}_i^{l-1}$. $\boldsymbol{b}_i^l$ is bias. $*$ denotes the convolutional operator. Size of $\boldsymbol{F}_i^l$ is $(p-w+1)\times (q-w+1)$. $g(\cdot)$ is a nonlinear activation function, such as the rectified linear unit function $g(x)=\max(0,x)$ [22].

The convolutional layer is often followed by a pooling layer. The most common pooling type is *max pooling*, which computes the maximum in local window of the feature map. Through the pooling, the feature map would be more robust to distortion of the data and obtain higher invariance. The pooling layer could reduce the size of the feature map, and thus decreases the computation burden [33].

On the top of a CNN, there are often several fully connected layers, which learn the high-level features and output the final prediction of the network. The feature vector learned by the lth fully connected layer is expressed as

$$\boldsymbol{F}^l = g(\boldsymbol{W}^l \cdot \boldsymbol{F}^{l-1} + \boldsymbol{b}^l) \tag{2}$$

where $\boldsymbol{W}^l$ is the weight matrix connecting the $(l-1)$th layer and the lth fully connected layer. $\boldsymbol{F}^{l-1}$ is the feature vector in the $(l-1)$th layer, and $\boldsymbol{b}^l$ is the bias vector of the lth layer.

Multiple convolutional layers, pooling layers, and fully connected layers are stacked to form a deep CNN network. Features learned by the stacked network are hierarchical. Bottom layers capture low-level features such as edges or texture, and top layers extract high-level features with more discriminative information, which is useful for classification tasks [34]. Compared with other deep learning models, CNN has much fewer connections and parameters due to its weight sharing and local connection scheme, which makes it easier to train [22].

III. Proposed Two-CNN For HSI Classification

Previous classification methods proposed for HSI can be treated as shallow models, and these methods mainly extract low-level features, such as edge, texture, and rotation angle. However, the invariance and robustness of the low-level features are limited and can hardly handle the challenge of high intraclass variability and low interclass variability [25], [26]. For example, in Fig. 2(a), the center pixels of patches 1 and 2 belong to the same class (i.e., asphalt), but their neighboring pixels have different shapes. The shape of asphalt in patch 1

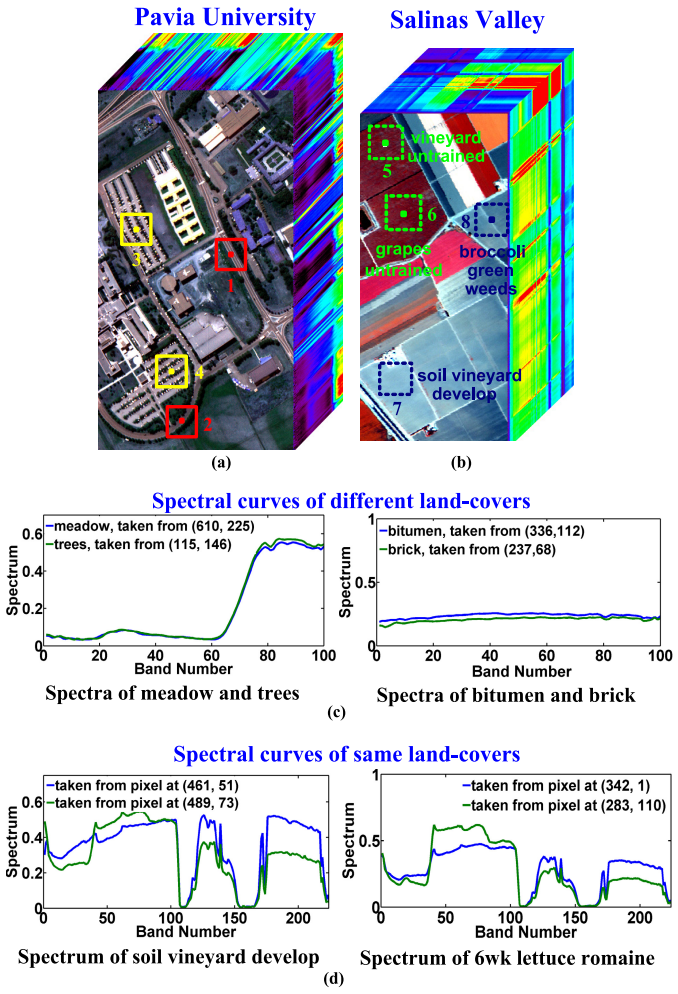


Fig. 2. Illustration of inter- and intravariability of different classes.

is straight, but the shape of asphalt in patch 2 is curving; thus, the difference of their low-level features would be distinct. The centers pixel in patches 3 and 4 belong to the same class, but there is a rotation between patches 3 and 4, which would lead to distinct low-level features as well. As shown in Fig. 2(b), even though the pixels in patches 5 and 6, and patches 7 and 8 have similar texture, they belong to different classes. Traditional low-level feature extraction methods may not lead to optimal classification result in the above cases. On the other hand, different land covers may generate spectra with similar shapes, and the spectra of the same land cover may be rather different. As shown in Fig. 2(c) and (d), meadow and trees have similar spectra, but soil of vineyard may present distinct spectra.

Therefore, extracting high-level features from the spatial and spectral domains jointly will improve classification accuracy [14], [17]. In this section, we propose a Two-CNN for the joint spectral–spatial feature learning.

A. Architecture of the Proposed Model

As shown in Fig. 3, the deep learning model has two branches of CNN, designed for spectral and spatial feature extractions. The spectral branch takes spectrum s_n of the nth

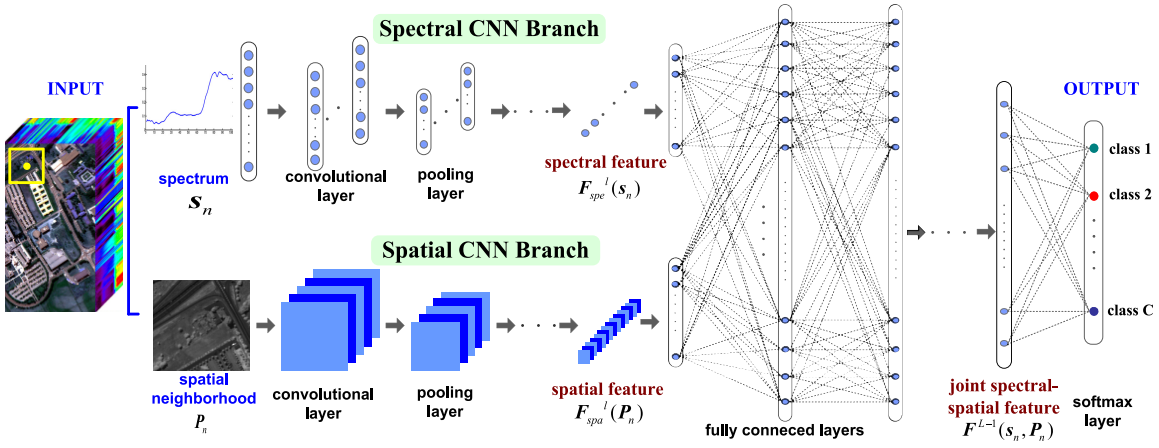


Fig. 3. Architecture of the proposed two branches of CNN for hyperspectral classification. (a) Pavia University data. (b) Salinas Valley data. (c) Spectral curves of different land-covers. (d) Spectral curves of same land-covers.

pixel as input, after l layers of the convolutional operation in (1) and *max* pooling, and we get the output $\boldsymbol{F}^{l}_{\text{spe}}(s_n)$ of the spectral branch, which can be regarded as spectral features. It should be noted that input s_n is 1-D signal, and all the convolutional and pooling operations in this branch belong to 1-D computation.

Hierarchical features extracted from the spatial domain are beneficial for discriminating land cover [14], [18]. Our proposed Two-CNN takes neighboring pixels of spectrum s_n as the input of spatial branch. HSI has contiguous spectral bands, and noise contaminated bands are often found. The noise can be modeled as additive Gaussian noise with zero mean [46], [47]. In order to fuse the spatial information of all bands and suppress the noise, we average all the images over the spectral bands, and then the spatial neighboring patch $\boldsymbol{P}(n) \in \mathbb{R}^{r\times r}$ (r is the patch size, and we fix the patch size 21×21 in the experiment) of the nth pixel is used as the input for the spatial branch; after l convolutional and pooling layers in this branch, the output $\boldsymbol{F}^{l}_{\text{spa}}(\boldsymbol{P}_n)$ can be regarded as spatial features.

In order to exploit both the spectral correlation and spatial correlation and obtain the joint spectral–spatial features, we feed simultaneously $\boldsymbol{F}^{l}_{\text{spe}}(s_n)$ and $\boldsymbol{F}^{l}_{\text{spa}}(P_n)$ to the fully connected layers. The output of $(l+1)$th layer is

$$\boldsymbol{F}^{l+1}(\boldsymbol{s}_n, \boldsymbol{P}_n) = g\{\boldsymbol{W}^{l+1} \cdot [\boldsymbol{F}^{l}_{\text{spe}}(\boldsymbol{s}_n) \oplus \boldsymbol{F}^{l}_{\text{spa}}(\boldsymbol{P}_n)] + \boldsymbol{b}^{l+1}\} \quad (3)$$

where $\boldsymbol{W}^{l+1}$ and $\boldsymbol{b}^{l+1}$ are the weight matrix and bias of the fully connected layer, respectively. $\oplus$ means concatenating the spectral features and spatial features. After several fully connected layers, the output of the last fully connected layer can be regarded as the final joint spectral–spatial features, which is followed by a softmax regression layer to predict the probability distribution of each class. The probability distribution is

$$\boldsymbol{p}(n) = \frac{1}{\sum_{k=1}^{C} e^{\boldsymbol{W}_k^L \boldsymbol{F}^{L-1}(\boldsymbol{s}_n, \boldsymbol{P}_n)}} \begin{bmatrix} e^{\boldsymbol{W}_1^L \boldsymbol{F}^{L-1}(\boldsymbol{s}_n, \boldsymbol{P}_n)} \\ e^{\boldsymbol{W}_2^L \boldsymbol{F}^{L-1}(\boldsymbol{s}_n, \boldsymbol{P}_n)} \\ \vdots \\ e^{\boldsymbol{W}_C^L \boldsymbol{F}^{L-1}(\boldsymbol{s}_n, \boldsymbol{P}_n)} \end{bmatrix} \quad (4)$$

where $\boldsymbol{W}_k^L (k = 1, 2, \ldots, C)$ is the kth row of weight matrix of L-h softmax regression layer. C is the number of classes. $\boldsymbol{p}(n) \in \mathbb{R}^C$ is a vector with C elements, and it is the probability of assigning the nth pixel to each class. $\boldsymbol{F}^{L-1}(\boldsymbol{s}_n, \boldsymbol{P}_n)$ is the final output of feature extraction; it conveys both spectral information and spatial information, and can be treated as the joint spectral–spatial features of the nth pixel.

The proposed classification method for HSI is designed based on CNN. Compared with methods based on other deep learning models (e.g., SAE in [24] and RBM in [35]), spatial correlation of HSI is preserved and exploited through the convolution on local image patches. In addition, CNN is less prone to overfitting because it has fewer connections and parameters to be tuned.

B. Training and Transfer Learning Strategy

All the convolutional kernels, weight matrices, and bias values in the model need to be trained. The set of training samples is denoted as $\{\boldsymbol{s}_n, \boldsymbol{P}_n, c(n)\}, (n = 1, 2, \ldots N)$. The loss function of the softmax regression layer is

$$J(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{C} 1\{k = c(n)\} \log \boldsymbol{p}_k(n) \quad (5)$$

where N is the number of samples. $c(n)$ is the class label of the nth training sample. $\boldsymbol{p}_k(n)$ is the kth element of $\boldsymbol{p}(n)$, and it is the probability of assigning the nth sample to class k. $1\{.\}$ is an indicative function, and its value would be one if the condition in the bracket is met; otherwise, it would be zero. θ denotes the set of convolutional kernels, weight matrices, and bias values to be trained. The loss function is optimized using the stochastic gradient descent method with standard back propagation [48].

Sufficient training samples are required to achieve high classification accuracy. However, in reality, labeled training samples are limited, which would result in overfitting. One solution is to train the deep learning network based on transfer learning strategy. Transfer learning extracts the knowledge from the source data and applies the knowledge to the target data. Knowledge used to transfer can be encoded in the

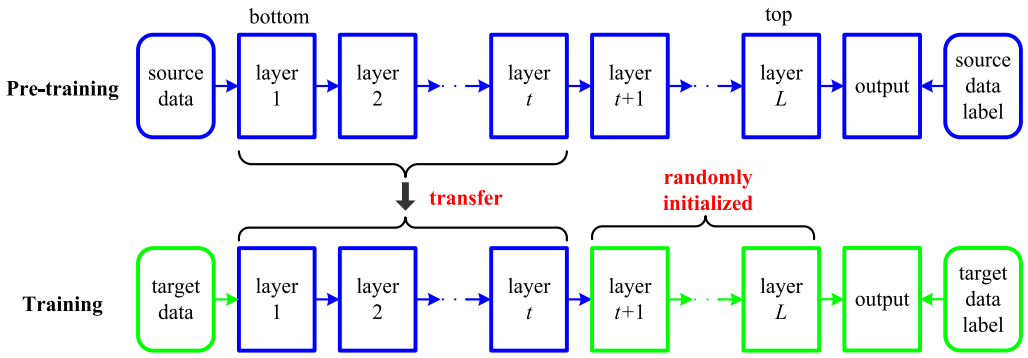


Fig. 4. Illustration of the transfer-learning-based training of the deep learning model.

features learned from the source data, with the transferred features, and the target task is expected to be improved [36]. Transfer learning can be applied to the training of a deep learning model as the network has hierarchical structure. Bottom layers capture low-level features such as edges and corners; these features are generic and could be transferred to the target scene. Top layers extract high-level features, which are specific to the target scene, so the top layers cannot be transferred and should be trained on the target data [39].

The transfer-learning-based training flowchart is shown in Fig. 4. We first pretrain a base Two-CNN network on source HSI (other hyperspectral scenes obtained using the same sensor), which is assumed to contain sufficient labeled samples. Then, we transfer the bottom layers of the pretrained network to the target network as initialization. To train the Two-CNN model for the target scene, the top layers are randomly initialized and trained together with the transferred layers using the limited labeled samples of target scene. It is worth noting that the data for pretraining should be collected by the same sensor as the target data, because data come from a common sensor have the same spectral configuration. The pretrained filters using the source data would have the same physical meaning as the target network and can be applied to the target data.

IV. Experimental Results

In this section, the performance of the proposed Two-CNN is evaluated. We present the parameter setting of the network architecture and analyze the sensitivity of the parameters. Transferability of the prelearned features is analyzed as well. We also demonstrate the advantages of extracting the joint spectral–spatial features in an integrated model. Then, the proposed method is compared with other state-of-the-art deep-learning-based classification methods.

Two data sets are used in the experiment. One data set is *Salinas Valley*, which was collected by Airborne Visible/Infrared Imaging Spectrometer sensor over Salinas, California. Size of the data is $512 \times 217 \times 224$. After removing water absorption bands and noisy bands, 200 bands are remained in the experiment. There are 16 classes of land covers in this data. The other data set is *Pavia University*, which was acquired by the Reflective Optics System Imaging Spectrometer sensor over Pavia, Italy. Size of the data is $610 \times 340 \times 103$. There are

(a) (b)

Fig. 5. False color composites of the data sets. (a) (Left) *Pavia University* and (Right) *Pavia Center* (Bands 10, 27, 46). (b) (Left) *Salinas Valley* and (Right) *Indian pines* (Bands 25, 19, 8).

TABLE I
Parameter Setting of the Network Architecture

Number of filters per conv. layer	20 (spectral branch) 30 (spatial branch)
Size of filter per conv. layer	16×1 (spectral branch) 3×3 (spatial branch)
Size of max pooling window	5×1 (spectral branch) 2×2 (spatial branch)
Number of conv. layers	2
Number of max pooling layers	1
Number of FC layers	2
Number of neurons per FC layer	400

100 bands remained after discarding the absorption bands and noisy bands. There are 9 classes of land covers in the data. For both data sets, we randomly choose r% ($r = 5, 10, 15, 20, 25$) of the labeled samples from each class for training. We test the performance on the whole image.

In the transfer learning experiment, the source data sets for pretraining are *Indian pines* and *Pavia Center*, which were collected by the same sensor as *Salinas Valley* and *Pavia University*, respectively. There are 16 classes in *India pines* and 9 classes in *Pavia Center*. Some classes of land covers

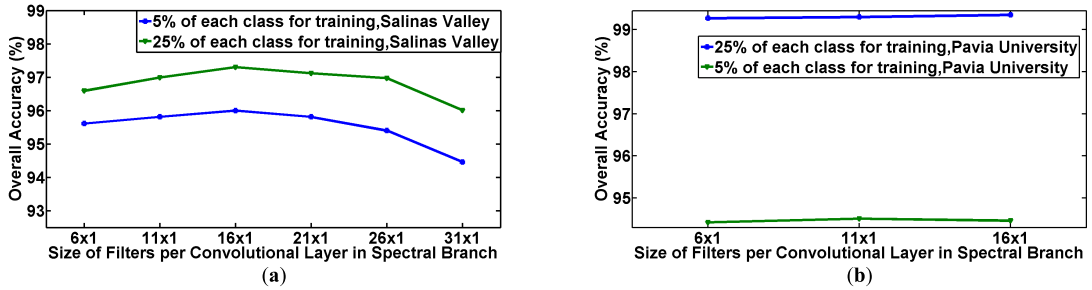


Fig. 6. Sensitivity of Two-CNN over the size of filter per convolutional layer in the spectral branch. (a) *Salinas Valley*. (b) *Pavia University*.

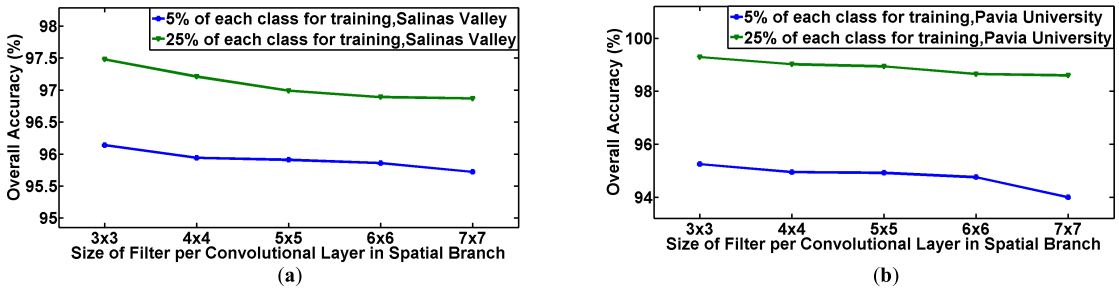


Fig. 7. Sensitivity of Two-CNN over the size of filter per convolutional layer in the spatial branch. (a) *Salinas Valley*. (b) *Pavia University*.

exist in both *Pavia Center* and *Pavia University*, and the land-cover classes in *Indian pines* are different from *Salinas Valley*. We assume that both *Indian pines* and *Pavia Center* contain sufficient labeled samples. The data sets are shown in Fig. 5.

In the experiment, the proposed method is denoted as Two-CNN, and the transfer-learning-based variant is denoted as Two-CNN transfer.

A. Sensitivity Analysis of Network Architecture

The architecture of deep learning network is often complex, and there are many parameters related to the network architecture, such as number of filters and layers. However, it is theoretically hard to determine these parameters. Here, we present the parameter setting of the proposed Two-CNN model in Table I, and then present the sensitivity analysis of these parameters.

In the experiments, we vary one parameter and fix others, and then observe the corresponding performance. We repeat the experiment ten times and then report the mean values. In order to present the curves concisely, we only report the results in 5% and 25% training sample cases in the figures. All of the convolutional kernels and weight matrix of the fully connected layers are initialized from the Gaussian random distribution, with settings: standard variance 0.05 and mean 0. The bias values are initialized to be 0. The parameters involved in the standard stochastic gradient descent method include learning rate, momentum, and batch size [48]. The learning rate is fixed as 0.0001, the momentum is set to 0.9, and the batch size is set to 128. The maximal training iterations are set to 3×10^5.

The sensitivity of Two-CNN over the size of filters in the spectral branch and spatial branch is presented in Figs. 6 and 7. It is noted that in Fig. 6(b), the data set that we used is *Pavia University* with 100 bands. If we set two convolutional layers, the maximal filter size in the spectral branch is 16×1 according to the network architecture in Table I. Proper large filter size in spectral branch is needed to collect enough spectral information. In Fig. 6(a), it is clear that filters of size 16×1 in the spectral branch lead to the best results on *Salinas Valley*. In Fig. 6(b), the performance decreases when filter size is 16×1 in the 5% training samples case, but the extent of decrease is slight. So we fix the filter size in the spectral branch to 16×1 in the experiment. Large filter in the spatial branch results in lower accuracies, as shown in Fig. 7. If the filter in spatial branch is large, more neighboring pixels would be involved in the spatial feature extraction, and the extracted features may contain unnecessary information that belongs to other classes, which imposes the negative effect for classification. Therefore, we set the filter size in spatial branch to 3×3.

Sensitivity over the number of filters is given in Figs. 8 and 9. In order to represent the features effectively, there should be enough feature maps per convolutional layer, so the performance increases with the increase of feature maps. However, more feature maps would result in a larger number of parameters to be trained, which requires more training samples. In the case of 5% training samples, the performance drops when feature maps are more than 20 in the spectral branch and 40 in the spatial branch. In Fig. 9, even though the performance increases monotonously in 25% training samples case, the margin of increase is slight. Therefore, we set the number of filters per convolutional layer in the spectral branch and the spatial branch to 20 and 30, respectively.

Sensitivity of the number of neurons per fully connected layer has the similar pattern with the number of filters,

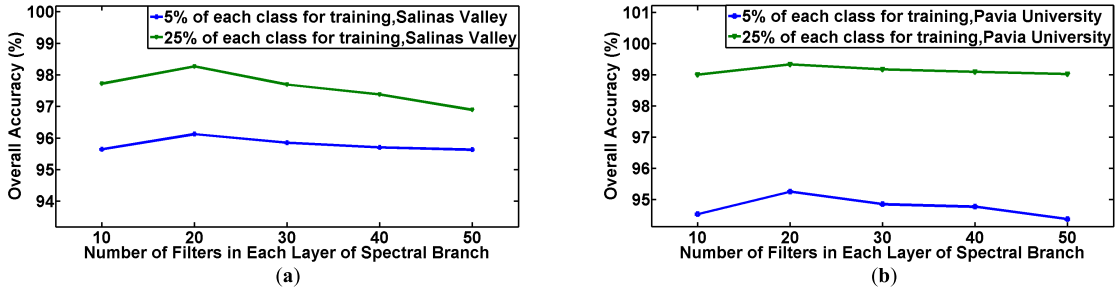


Fig. 8. Sensitivity of Two-CNN over the number of filters per convolutional layer in the spectral branch. (a) *Salinas Valley*. (b) *Pavia University*.

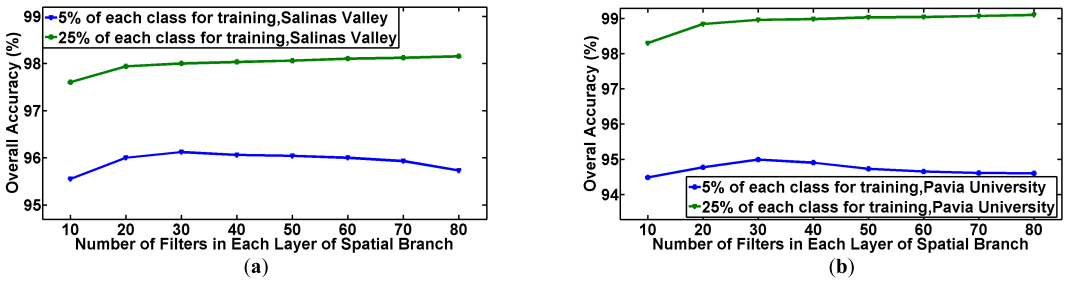


Fig. 9. Sensitivity of Two-CNN over the number of filters per convolutional layer in the spatial branch. (a) *Salinas Valley*. (b) *Pavia University*.

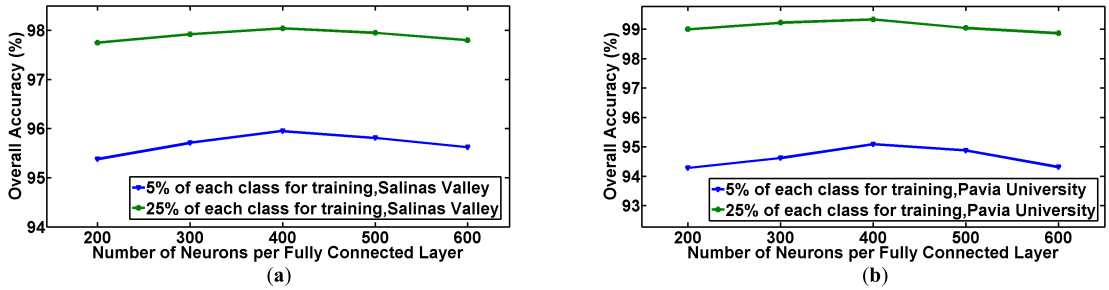


Fig. 10. Sensitivity of Two-CNN over the number of neurons per fully connected layer. (a) *Salinas Valley*. (b) *Pavia University*.

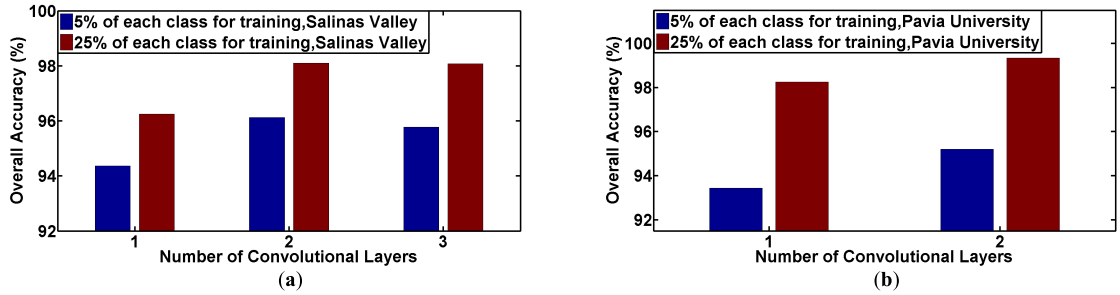


Fig. 11. Sensitivity of Two-CNN over the number of convolutional layers. (a) *Salinas Valley*. (b) *Pavia University*.

as shown in Fig. 10. The performance increases when the number of neurons per fully connected layer varies from 200 to 400. When the number of neurons is higher than 500, the classification performance declines. We therefore set the number of neurons per fully connected layer to 400.

Usually, the more layers of the deep learning network, the more discriminative the feature is, and the higher classification accuracy it could achieve [23]. The sensitivity of the number of convolutional layers to the classification performance is given in Fig. 11, and the sensitivity of the number of fully connected layers is given in Fig. 12. It should be noted that in Fig. 11(b), the data set that we used is *Pavia University* with 100 spectral bands, and the maximal number of convolutional layers we can set is two according to the network architecture in Table I. From Fig. 11(a), we can find that the classification accuracy of Two-CNN with two convolutional

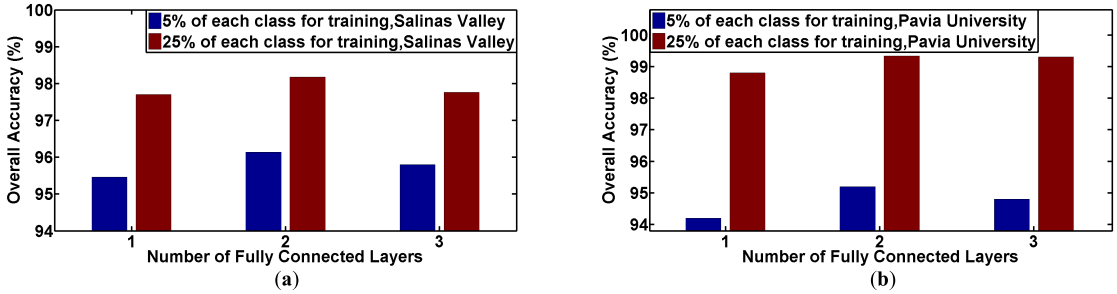


Fig. 12. Sensitivity of Two-CNN over the number of fully connected layers. (a) *Salinas Valley*. (b) *Pavia University*.

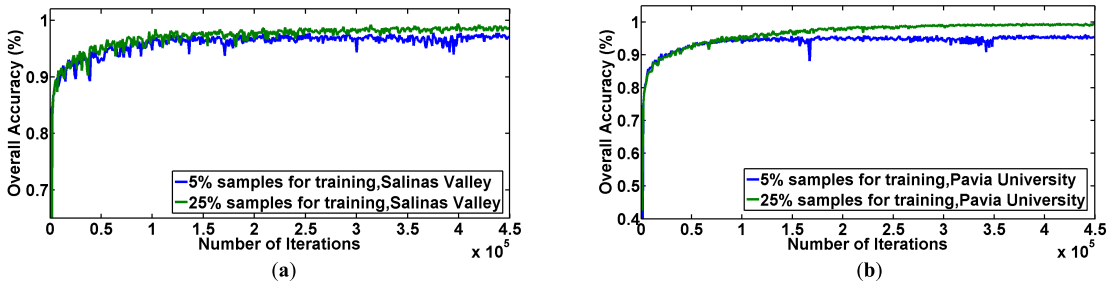


Fig. 13. Evolution of overall accuracy over the iterations of training. (a) *Salinas Valley*. (b) *Pavia University*.

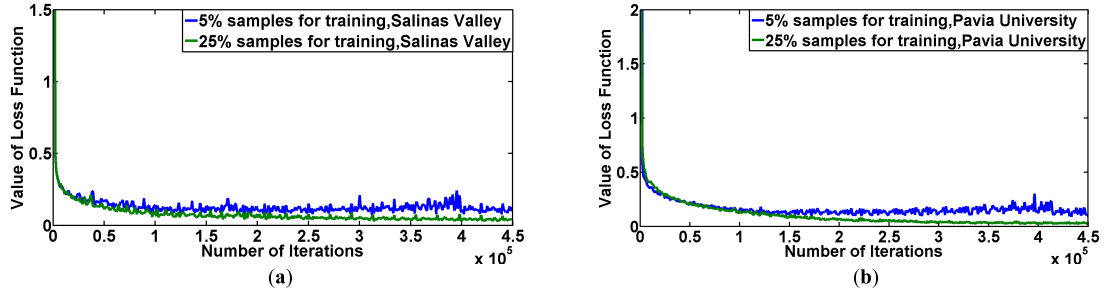


Fig. 14. Evolution of loss function over the iterations of training. (a) *Salinas Valley*. (b) *Pavia University*.

layers is higher than Two-CNN with only one convolutional layer. When the number of convolutional layers is larger than two convolutional layers, the classification accuracy would drop or increase very slowly. The same pattern can also be observed in Fig. 12. So there are two convolutional layers and two fully connected layers in our final proposed network.

In Figs. 13 and 14, we present the evolution of overall accuracy and loss function over iterations. In Fig. 13, the classification performance increases drastically in the first 0.5×10^5 iterations, and then plateaus after 3×10^5 iterations. In Fig. 14, the loss function declines sharply in the first 0.5×10^5 iterations; after the big descend, the rate of decrease becomes smaller, and the loss function tends to be constant after 3×10^5 iterations. Therefore, the number of iterations setting to 3×10^5 is adequate to generate good result. It is worth noting that there is fluctuation in the above curves, which may be caused by the stochastic nature of stochastic gradient descend optimization method [48].

B. Transferability Analysis of Prelearned Features

Transfer learning strategy is introduced in this paper to boost performance in the cases with small training samples. However, how many layers of prelearned network should be transferred to the target network is an open question. In Section IV-B, we analyze the transferability of the prelearned features by comparing the performance of transferring various layers of features.

We first train two base Two-CNN networks on *Indian pines* and *Pavia Center*. For *Indian pines*, we randomly choose 90% samples of each class for training. For *Pavia Center*, we randomly choose 15% samples of each class for training. To obtain sufficient samples, higher percentage of samples is chosen for *Indian pines* because its size is smaller. The pretrained networks are then transferred to the target networks for *Salinas Valley* and *Pavia University,* respectively. Two-CNN$_{tr}$ $\{tr = 1, 2, \ldots, 4\}$ denotes training only the top tr layers on the limited labeled target data, and the remaining bottom layers are transferred from the pretrained source network. For example, Two-CNN$_1$ means transferring all layers but the top softmax classifier layer would be randomly initialized and trained on the target data. In order to validate the transfer learning performance in small training samples cases, we randomly choose $\{25, 50, 75\}$ samples from each class of *Salinas Valley* and *Pavia University* for training. The performance of transferring various layers is reported in Tables II and III.

The effectiveness of the transfer training can be demonstrated in Tables II and III. It is clear that transfer learning improves the classification performance, but the degree of

TABLE II

OVERALL ACCURACY OF TRANSFER LEARNING BY TRANSFERRING VARIOUS NUMBER OF LAYERS FROM THE PRETRAINED TWO-CNN OF INDIAN PINES TO THE TWO-CNN OF SALINAS VALLEY

Number of Training Samples per Class	Without Transfer	Two-CNN$_4$	Two-CNN$_3$	Two-CNN$_2$	Two-CNN$_1$
25	85.84%	88.80%	88.81%	**89.04**%	88.60%
50	88.56%	90.85%	**91.83**%	91.28%	90.89%
75	91.11%	91.44%	**92.41**%	91.51%	91.42%

TABLE III

OVERALL ACCURACY OF TRANSFER LEARNING BY TRANSFERRING VARIOUS NUMBER OF LAYERS FROM THE PRETRAINED TWO-CNN OF PAVIA CENTER TO THE TWO-CNN OF PAVIA UNIVERSITY

Number of Training Samples per Class	Without Transfer	Two-CNN$_4$	Two-CNN$_3$	Two-CNN$_2$	Two-CNN$_1$
25	68.07%	74.58%	77.17%	**77.48**%	77.10%
50	79.75%	84.92%	**85.40**%	85.01%	85.59%
75	85.15%	87.18%	**87.52**%	86.98%	86.19%

improvement varies with the number of layers to be transferred. We can also find that in most cases, the best classification result is achieved when transferring three bottom layers; the last two fully connected layers and the softmax classifier layer should be randomly initialized and trained from training samples from the target scene. Features extracted by the last two fully connected layers are often specific to the data, so it is better to train these layers from the target data [39]. When reducing the training samples to 25 samples per class, transferring four bottom layers achieves the best performance, rather than transferring three layers. This phenomenon may be caused by the target training data, which is too scarce to adequately train two fully connected layers, so transferring four bottom layers and training only one fully connected layer results in higher overall accuracy.

The transfer learning boosts the performance by a larger margin on *Pavia University* than on *Salinas Valley*, as shown in Tables II and III. The improvement of transfer learning depends on the disparity between the source data and the target data. If the disparity is small, the improvement of transferring learning would be high, and vice versa [36], [44]. There are common land covers in *Pavia Center* and *Pavia University* (e.g., asphalt and bricks), but the land covers of *Salinas Valley* are different from those of *Indian pines*. The larger class disparity between *Salinas Valley* and *Indian pines* explains why transfer learning performs better on the *Pavia University* scene.

As presented in Tables II and III, when we extract 25 samples per class for training, the overall accuracy improved by transfer learning is 3.20% for *Salinas Valley* and 9.41% for *Pavia University*. When the number of training samples per class is 75, the improvement reduces to 1.30% for *Salinas Valley* and 2.37% for *Pavia University*. The improvement drops with the increase of training samples, because network with more specific information can be trained directly from the target data. Two-CNN can perform well even without transfer learning.

TABLE IV

OVERALL ACCURACY OF CLASSIFICATION ON SALINAS VALLEY BY EXTRACTING SPECTRAL INFORMATION, SPATIAL INFORMATION, AND SPECTRAL–SPATIAL INFORMATION, RESPECTIVELY

Ratio of training samples	Two-CNN-spe	Two-CNN-spa	Two-CNN
5%	91.03%	93.40%	**95.20**%
10%	91.26%	95.53%	**97.15**%
15%	91.42%	95.58%	**97.40**%
20%	91.71%	96.43%	**97.99**%
25%	92.26%	96.57%	**98.27**%

C. Visualization of Feature Learning

The Two-CNN can extract the joint spectral–spatial features, which is of high dimensions (i.e., 400-D in this experiment) and of high abstractness. In order to visualize the discriminative ability of the high-dimensional features, we reduce the dimensionality of the learned features using t-SNE method [49]. 2-D features are plotted after t-SNE, *Pavia University* in Fig. 15 and *Salinas Valley* in Fig. 16. In Figs. 15(a) and 16(a), concatenation of the raw spectrum and the spatial patch is used as features for visualization. The raw pixels are taken as features without any learning, and their discriminative ability is limited, as we can see points of different classes are overlapped with each other. When joint spectral–spatial features are learned, the learned features of different pixels from the same classes gather into several clusters, making it much easier to separate. With the increase of iterations in training, the clusters would be more separable with each other, as shown in Figs. 15 and 16.

D. Rationality Analysis of Jointly Extracting Spectral and Spatial Features

Spectral information and spatial information are jointly exploited for the feature extraction in the proposed method. The rationality of combining the spectral information and

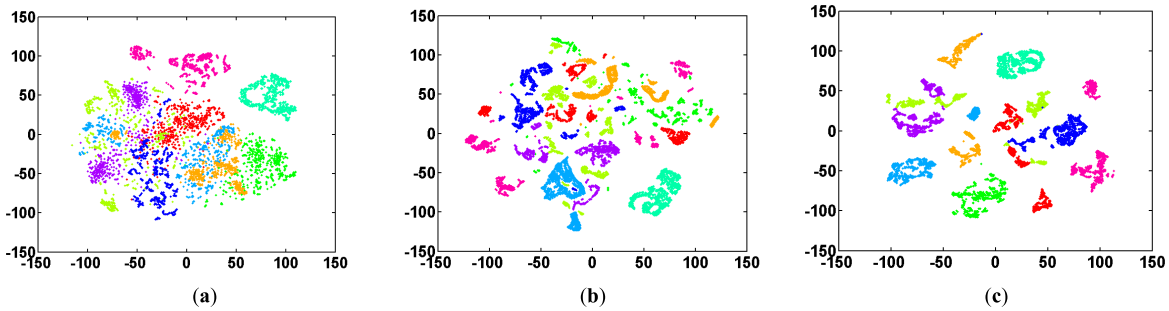


Fig. 15. Two-dimensional feature visualization of the joint spectral–spatial features via t-SNE method. Each point represents one sample, and different colors represent different classes. The 25% samples of each class are used for training Two-CNN on *Pavia University*, and 700 samples are randomly chosen from each class to compute the features. (a) Two-dimensional visualization of raw features stacked by spectrum and spatial patch. (b) Two-dimensional visualization of the joint spectral–spatial features trained with 1000 iterations (OA = 43.39%). (c) Two-dimensional visualization of the joint spectral–spatial features trained with 300 000 iterations (OA = 99.02%).

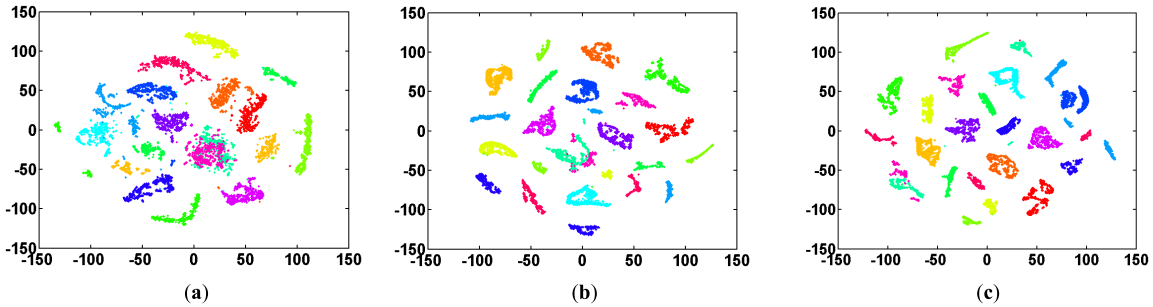


Fig. 16. Two-dimensional feature visualization of the joint spectral–spatial features via t-SNE method. Each point represents one sample, and different colors represent different classes. The 20% samples of each class are used for training Two-CNN on *Salinas Valley*, and 500 samples are randomly chosen from each class to compute the features. (a) Two-dimensional visualization of raw spectrum features stacked by spectrum and spatial patch. (b) Two-dimensional visualization of the joint spectral–spatial features trained with 7000 iterations (OA = 32.73%). (c) Two-dimensional visualization of the joint spectral–spatial features trained with 300 000 iterations (OA = 98.57%).

spatial information is analyzed here. In order to demonstrate the effectiveness of jointly utilizing spectral information and spatial information, we remove the spectral branch or spatial branch from Two-CNN and keep other architecture parameters fixed; thus; only spectral information or spatial information is exploited in the extracted features. The classification result with spectral information only is denoted as Two-CNN-spe, and the classification result with spatial information only is denoted as Two-CNN-spa. The results are presented in Tables IV and V. Exploiting either spectral information or spatial information only is inferior to exploiting spectral information and spatial information jointly. The experimental result in Tables IV and V demonstrates the rationality of jointly extracting spectral and spatial features, and it also demonstrates that the spectral information and spatial information are complementary to each other for the classification.

E. Comparison With Other State-of-the-Art Methods

In this section, the proposed Two-CNN and Two-CNN-transfer methods are compared with other state-of-the-art classification methods. The compared methods are SVM method [7], spectral–spatial joint sparse representation classification (JSRC) method [8], SAE method [24], and CNN combined with manifold learning (CNN + ML) method [26]. SVM can achieve stable performance and is often used as a benchmark in HSI classification. Both of SAE and CNN + ML methods are in the framework of deep learning, it leads to a fair comparison.

TABLE V

OVERALL ACCURACY OF CLASSIFICATION ON PAVIA UNIVERSITY BY EXTRACTING SPECTRAL INFORMATION, SPATIAL INFORMATION, AND SPECTRAL–SPATIAL INFORMATION, RESPECTIVELY

Ratio of training samples	Two-CNN-spe	Two-CNN-spa	Two-CNN
5%	88.85%	91.34%	**94.57%**
10%	89.24%	95.16%	**96.72%**
15%	90.13%	97.05%	**98.08%**
20%	90.44%	97.20%	**98.64%**
25%	91.51%	98.29%	**98.83%**

The parameters setting of compared methods first follow suggestions from the original authors, and then, we empirically tune them to achieve the best performance. The SVM classification method is implemented with LIBSVM toolbox [50], the type of kernel function of SVM is radial basis function, and the regularization parameters are determined by fivefold crossvalidation in the range of $[2^{-10}, 2^{-9}, \ldots, 2^{19}, 2^{20}]$. The parameters in JSRC method include sparsity parameter and the size of window for joint sparse coding. The sparsity parameter is set to 50, and the window size is set to 9×9. In order to compare fairly, size of the spatial neighborhood in SAE method is set to 21×21, the same with ours. There are two hidden layers in SAE, and the number of neurons per layer is set to 60. The learning rate for unsupervised pretraining and supervised fine-tuning is 0.02 and 0.008, respectively. The momentum for unsupervised pretraining and supervised fine-tuning is

TABLE VI
OVERALL ACCURACY (%) OF DIFFERENT CLASSIFICATION METHODS ON SALINAS VALLEY DATA

Ratio of training samples	SVM [7]	JSRC [8]	SAE [24]	CNN+ML [26]	Two-CNN	Two-CNN-transfer
5%	93.04±0.20	89.77±0.35	91.71±0.18	89.13±1.24	95.96±0.44	**96.94**±0.39
10%	94.46±0.12	95.17±0.42	93.45±0.23	96.48±0.81	97.12±0.30	**97.51**±0.45
15%	94.89±0.06	96.55±0.33	93.75±0.39	97.03±0.47	98.01±0.32	**98.19**±0.29
20%	95.43±0.20	97.51±0.29	95.70±0.36	97.19±1.02	98.13±0.43	**98.38**±0.34
25%	95.68±0.18	97.99±0.34	96.22±0.33	97.81±0.52	98.27±0.28	**98.51**±0.39

TABLE VII
OVERALL ACCURACY (%) OF DIFFERENT CLASSIFICATION METHODS ON PAVIA UNIVERSITY DATA

Ratio of training samples	SVM [7]	JSRC [8]	SAE [24]	CNN+ML [26]	Two-CNN	Two-CNN-transfer
5%	93.99±0.29	82.80±0.25	90.53±0.21	91.28±2.35	94.40±0.39	**95.31**±0.30
10%	94.85±0.08	87.97±0.39	93.50±0.32	92.51±1.05	96.99±0.32	**97.17**±0.53
15%	95.54±0.05	92.55±0.28	95.21±0.28	93.01±0.62	98.17±0.29	**98.62**±0.27
20%	95.76±0.10	94.01±0.41	95.93±0.14	94.28±1.13	98.72±0.25	**99.06**±0.18
25%	95.95±0.22	96.43±0.36	96.37±0.41	96.20±0.79	99.10±0.18	**99.21**±0.23

TABLE VIII
STATISTICAL SIGNIFICANCE OF DIFFERENCE (Z VALUE) BETWEEN DIFFERENT METHODS ON SALINAS VALLEY

Ratio of training samples	Methods	SVM [7]	JSRC [8]	SAE [24]	CNN+ML [26]
5%	Two-CNN	34.04	45.13	46.91	52.92
	Two-CNN-transfer	42.96	50.39	52.37	61.56
10%	Two-CNN	36.87	27.10	47.51	8.83
	Two-CNN-transfer	42.55	29.74	51.16	23.00
15%	Two-CNN	39.12	21.26	43.89	14.89
	Two-CNN-transfer	42.18	22.41	46.56	25.17
20%	Two-CNN	36.55	20.41	50.41	18.75
	Two-CNN-transfer	41.04	23.04	53.51	25.30
25%	Two-CNN	29.11	23.70	29.70	8.57
	Two-CNN-transfer	38.22	28.41	38.10	9.28

0.98 and 0.9, respectively. The number of iterations for unsupervised pretraining and supervised fine-tuning of SAE is set to 2000 and 20000, respectively. There are five convolutional layers in CNN + ML method, the number of feature maps per layer is set to 20, and the other parameters of manifold learning for spectral feature extraction in CNN + ML are set according to the suggestion in [26]. The number of transferred layers of Two-CNN-transfer method is three, and the two fully connected layers and the top classifier layer are trained on the target data.

We repeat the experiment ten times and report the mean and standard deviation of overall accuracy in Tables VI and VII. In order to demonstrate that the differences between our proposed method and the compared methods are statistically significant, we also perform McNemars testing, which is computed as [53], [54]

$$Z = \frac{f_{12} - f_{21}}{\sqrt{f_{12} + f_{21}}}$$

where Z is the statistical significance value of two methods, f_{12} is the number of samples that can be correctly classified by method 1, but that can be incorrectly classified by method 2, and f_{21} is the number of samples that can be correctly classified by method 2, but that can be incorrectly classified

TABLE IX
STATISTICAL SIGNIFICANCE OF DIFFERENCE (Z VALUE) BETWEEN DIFFERENT METHODS ON PAVIA UNIVERSITY

Ratio of training samples	Methods	SVM [7]	JSRC [8]	SAE [24]	CNN+ML [26]
5%	Two-CNN	13.19	77.87	44.72	36.66
	Two-CNN-transfer	20.24	79.01	46.73	41.76
10%	Two-CNN	32.33	61.75	41.44	45.28
	Two-CNN-transfer	34.62	63.07	43.10	46.87
15%	Two-CNN	33.81	48.34	33.46	46.65
	Two-CNN-transfer	37.29	51.20	37.13	49.22
20%	Two-CNN	34.42	43.56	35.44	43.41
	Two-CNN-transfer	35.02	44.03	36.00	43.87
25%	Two-CNN	36.16	39.40	32.62	35.28
	Two-CNN-transfer	37.34	41.07	33.89	36.33

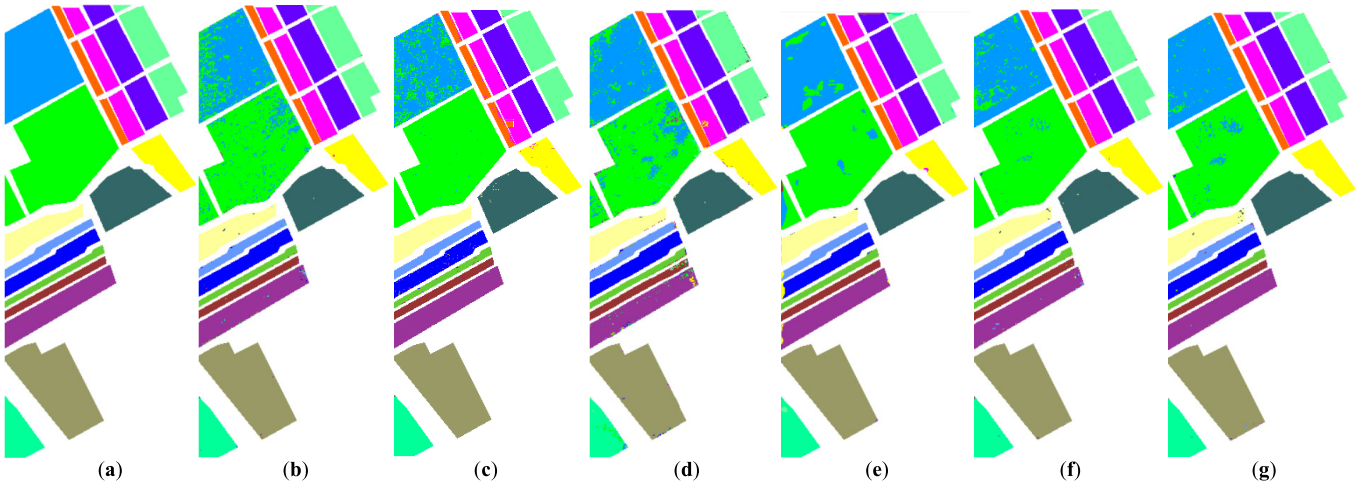


Fig. 17. Classification maps of different methods on *Salinas Valley* data, 15% samples per class are chosen for training. (a) Ground truth. (b) Result of SVM method. (c) Result of JSRC method. (d) Result of SAE method. (e) Result of CNN + ML method. (f) Result of Two-CNN method. (g) Result of Two-CNN-transfer method.

by method 1. If $|Z| > 1.96$, the difference of the two methods is said to be statistically significant; $Z > 0$ means that method 1 is more accurate than method 2, and $Z < 0$ means that method 2 is more accurate than method 1 [53], [54]. The statistical significance values between our proposed method and the compared methods are presented in Tables VIII and IX.

The overall accuracy of different classification methods is given in Tables VI and VII. The proposed Two-CNN and Two-CNN-transfer outperform other methods in different training samples cases. It is noted that the transfer learning can still boost Two-CNN even though the training samples are not scarce. SVM takes raw pixels as input, even though it can achieve stable performance, and it is inferior to other deep learning-based methods when training samples are sufficient, which also demonstrates the importance of depth of the model. JSRC performs better on *Salinas Valley* than on *Pavia University*, and the performance of JSRC on *Salinas Valley* is competitive with SAE and CNN + ML, but it is still lower than the proposed method. From Tables VIII and IX, it can be found that the Z values between our proposed method and the compared methods are higher than 1.96, which means that our method outperforms the compared methods, and the differences are significant in statistical sense.

Parts of the classification maps are shown in Figs. 17 and 18. Spatial correlation is destroyed due to the vectorization in SAE, so the classification map of SAE is not smooth with some "noise." Spatial features are extracted by CNN without vectorization in CNN + ML, and the classification map is much smoother than SAE. But the spectral features and spatial features are learned independently in CNN + ML, which limits the discriminative ability of the final spectral–spatial joint features and results in errors. Comparatively, the classification maps of Two-CNN and Two-CNN-transfer contain less errors.

The training of Two-CNN and Two-CNN-transfer is implemented on Caffe platform [46], with an NVIDIA GTX 960 graphic card. It takes about 30 min if we set the maximal training iterations to 3×10^5. Testing on the whole image costs less than one second, and it is fast because there is only

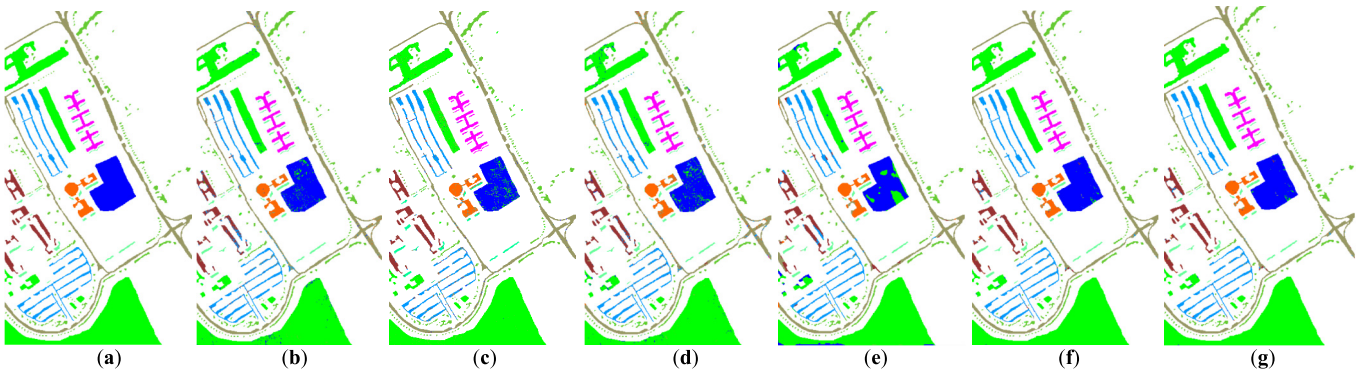


Fig. 18. Classification maps of different methods on *Pavia University* data, 20% samples per class are chosen for training. (a) Ground truth. (b) Result of SVM method. (c) Result of JSRC method. (d) Result of SAE method. (e) Result of CNN + ML method. (f) Result of Two-CNN method. (g) Result of Two-CNN-transfer method.

a feed-forward computation in the testing stage. The code of this work is available online https://github.com/polwork/Two-CNN-HSI.

V. Conclusion

In this paper, we propose to learn the deep joint spectral–spatial features for hyperspectral classification with a Two-CNN. It contains two branches of CNN; one branch learns features from the spectral domain and the other branch learn features from the spatial domain. The spectral features and the spatial features extracted by the two branches of CNN are then concatenated and fed to fully connected layers; higher-level spectral–spatial joint features are then obtained. In order to improve the performance in small sample cases, we also propose to train the model based on transfer learning. The bottom layers are generic to different data and can be transferred from networks that have been pretrained on other data; only the top few layers are specific to the target scene and needed to be trained from the target data. Experiment results demonstrate that the transfer learning can boost the performance of classification. However, the CNN architectures are empirically determined how the optimal architecture parameters still need further research. The data used for transfer learning now come from the same sensor, how to exploit data of other sensors and transfer the knowledge is the challenge of our future research.

Acknowledgment

The authors would like to thank the editors and the reviewers for their kind work and constructive suggestions. They would also like to thank Dr. W. Zhao from Peking University for helping with implementing CNN + ML method.

References

[1] L. Liang *et al.*, "Estimation of crop LAI using hyperspectral vegetation indices and a hybrid inversion method," *Remote Sens. Environ.*, vol. 165, pp. 123–134, Aug. 2015.

[2] G. V. Laurin *et al.*, "Biodiversity mapping in a tropical West African forest with airborne hyperspectral data," *PLoS ONE*, vol. 9, no. 6, p. e97910, 2014.

[3] N. Yokoya, J. C.-W. Chan, and K. Segl, "Potential of resolution-enhanced hyperspectral data for mineral mapping using simulated EnMAP and Sentinel-2 images," *Remote Sens.*, vol. 8, no. 3, p. 172, 2016.

[4] 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.

[5] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3804–3814, Nov. 2008.

[6] J. Li, P. R. Marpu, A. Plaza, J. M. Bioucas-Dias, and J. A. Benediktsson, "Generalized composite kernel framework for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 9, pp. 4816–4829, Sep. 2013.

[7] 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.

[8] 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.

[9] Y. Qian, M. Ye, and J. Zhou, "Hyperspectral image classification based on structured sparse logistic regression and three-dimensional wavelet texture features," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 4, pp. 2276–2291, Apr. 2013.

[10] S. Jia, L. Shen, and Q. Li, "Gabor feature-based collaborative representation for hyperspectral imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 2, pp. 1118–1129, Feb. 2015.

[11] 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.

[12] L. Zhang, L. Zhang, D. Tao, and X. Huang, "On combining multiple features for hyperspectral remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 3, pp. 879–893, Mar. 2012.

[13] J. Li *et al.*, "Multiple feature learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1592–1606, Mar. 2015.

[14] W. Liao, M. D. Mura, J. Chanussot, and A. Pižurica, "Fusion of spectral and spatial information for classification of hyperspectral remote-sensed imagery by local graph," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 583–594, Feb. 2016.

[15] L. Zhang, L. Zhang, D. Tao, and X. Huang, "Tensor discriminative locality alignment for hyperspectral image spectral–spatial feature extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 242–256, Jan. 2013.

[16] X. Guo, X. Huang, L. Zhang, L. Zhang, A. Plaza, and J. A. Benediktsson, "Support tensor machines for classification of hyperspectral remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3248–3264, Jun. 2016.

[17] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1349–1362, Mar. 2016.

[18] 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.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[20] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[21] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[22] 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.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," presented at the Int. Conf. Learn. Represent., 2015.

[24] 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.

[25] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva. (2015). "Land use classification in remote sensing images by convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1508.00092

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

[27] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectral–spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2438–2442, Dec. 2015.

[28] 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.

[29] 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.

[30] 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.

[31] W. Zhao and S. Du, "Learning multiscale and deep representations for classifying remotely sensed imagery," *ISPRS J. Photogramm. Remote Sens.*, vol. 113, pp. 155–165, Mar. 2016.

[32] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 253–256.

[33] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proc. Int. Conf. Artif. Neural Netw.*, 2010, pp. 92–101.

[34] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2018–2025.

[35] T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *Proc. IEEE Int. Conf. Image Process.*, 2014, pp. 5132–5136.

[36] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[37] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1019–1034, May 2014.

[38] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with Gaussian processes regression," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 188–203.

[39] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1717–1724.

[40] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing, "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 69–82.

[41] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep learning earth observation classification using ImageNet pretrained networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 1, pp. 105–109, Jan. 2016.

[42] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sens.*, vol. 7, no. 11, pp. 14680–14707, 2015.

[43] M. Xie, N. Jean, M. Burke, D. Lobell, and S. Ermon, "Transfer learning from deep features for remote sensing and poverty mapping," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 3929–3935.

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.

[45] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.

[46] Y.-Q. Zhao and J. Yang, "Hyperspectral image denoising via sparse representation and low-rank constraint," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 1, pp. 296–308, Jan. 2015.

[47] J. Yang, Y.-Q. Zhao, J. C.-W. Chan, and S. G. Kong, "Coupled sparse denoising and unmixing with low-rank constraint for hyperspectral image," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1818–1833, Mar. 2016.

[48] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 421–436.

[49] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

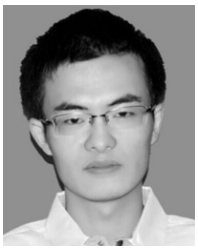
[50] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011.

[51] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[52] D. Tuia, C. Persello, and L. Bruzzone, "Domain adaptation for the classification of remote sensing data: An overview of recent advances," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 41–57, Jun. 2016.

[53] W. Liao, A. Pizurica, P. Scheunders, W. Philips, and Y. Pi, "Semisupervised local discriminant analysis for feature extraction in hyperspectral images," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 184–198, Jan. 2013.

[54] G. M. Foody, "Thematic map comparison," *Photogramm. Eng. Remote Sens.*, vol. 70, no. 5, pp. 627–633, 2004.



Jingxiang Yang (S'14) received the B.S. degree in automation from Anhui Polytechnic University, Wuhu, China, in 2011, and the M.S. degree in control theory and control engineering from Northwestern Polytechnical University, Xi'an, China, in 2014. He is currently pursuing the Ph.D. degree with the School of Automation, Northwestern Polytechnical University, Xi'an.

He is a joint Ph.D. Student with the Department of Electronics and Informatics, Vrije Universiteit Brussel, Brussels, Belgium. His research interests include hyperspectral image processing, sparse representation, and deep learning.

Yong-Qiang Zhao (M'05) received the B.S. degree in automation and the M.S. and Ph.D. degrees in control theory and control engineering from Northwestern Polytechnical University, Xi'an, China, in 1998, 2001, and 2004, respectively.

From 2007 to 2009, he was a Post-Doctoral Researcher at McMaster University, Hamilton, ON, Canada, and at Temple University, Philadelphia, PA, USA. He is currently a Professor with Northwestern Polytechnical University. His research interests include polarization imaging analysis, imaging spectrometry, compressive sensing, and pattern recognition.

Jonathan Cheung-Wai Chan received the Ph.D. degree from the University of Hong Kong, Hong Kong, in 1999.

From 1998 to 2001, he was with the Department of Geography, University of Maryland, College Park, MD, USA. From 2001 to 2005, he was with the Interuniversity Micro-Electronics Center, Leuven, Belgium. From 2005 to 2011, he was with the Department of Geography, Vrije Universiteit Brussel (VUB), Brussels, Belgium. From 2013 to 2014, he was a Marie Curie Fellow at Fondazione Edmund Mach, Trentino, Italy. He is currently a Guest Professor with the Department of Electronics and Informatics, VUB. He is also a Visiting Professor with the Warsaw University of Life Sciences, Warsaw, Poland, and with Northwestern Polytechnical University, Xi'an, China.

Dr. Chan was a Guest Editor of *Remote Sensing*, for the special issue on Spatial Enhancement of Hyperspectral Data and Applications.