Deep Few-Shot Learning for Hyperspectral Image Classification

Bing Liu, Xuchu Yu, Anzhu Yu, Pengqiang Zhang, Gang Wan, and Ruirui Wang

***Abstract*— Deep learning methods have recently been successfully explored for hyperspectral image (HSI) classification. However, training a deep-learning classifier notoriously requires hundreds or thousands of labeled samples. In this paper, a deep few-shot learning method is proposed to address the small sample size problem of HSI classification. There are three novel strategies in the proposed algorithm. First, spectral–spatial features are extracted to reduce the labeling uncertainty via a deep residual 3-D convolutional neural network. Second, the network is trained by episodes to learn a metric space where samples from the same class are close and those from different classes are far. Finally, the testing samples are classified by a nearest neighbor classifier in the learned metric space. The key idea is that the designed network learns a metric space from the training data set. Furthermore, such metric space could generalize to the classes of the testing data set. Note that the classes of the testing data set are not seen in the training data set. Four widely used HSI data sets were used to assess the performance of the proposed algorithm. The experimental results indicate that the proposed method can achieve better classification accuracy than the conventional semisupervised methods with only a few labeled samples.**

***Index Terms*— 3-D convolutional neural networks (3-D CNNs), few-shot learning, hyperspectral image (HSI) classification, residual learning.**

I. Introduction

HYPERSPECTRAL image (HSI) analysis can provide end users with abundant spectral and spatial information simultaneously [1]. Such abundant information has been widely used in applications related to earth observation such as environmental monitoring and accurate agriculture. HSI classification may prove to be of great significance to these fields. The purpose of HSI classification is to assign each pixel to one specific class based on the corresponding feature representation. According to the availability of labeled samples, classification could be generally divided into supervised classification, unsupervised classification, and semisupervised classification. In general, supervised classifiers would achieve higher classification accuracy than unsupervised classifiers. In this context, supervised spectral classifiers such as support vector machines (SVMs) [2], random forest [3], neural network [4], and extreme learning machines [5] have been heavily studied in the hyperspectral community. The most widely used supervised classifiers are available in [6]. However, the curse of dimensionality caused by high dimensions and the small samples is still a bottleneck for supervised classifiers.

Feature extraction is considered as a useful method to alleviate the problem of the curse of dimensionality. Simple feature extraction methods use principal component analysis [7], independent component analysis [8], or local linear feature extraction [9] for dimensionality reduction. Recent studies have also demonstrated that incorporating the spatial information into pixelwise classifiers is helpful to improve the classification performance [10]. For this reason, a large number of spectral–spatial feature extraction methods have been developed for HSI classification, such as extended morphological profiles [11], Markov random field [12], local binary pattern [13], Gabor wavelet filters [14], discriminant analysis [15], [16], and segmentation approaches [17]. In addition, band-selection methods [18], [19] are also applied in HSI to reduce feature dimensions. It is usually difficult for a supervised classifier to obtain a satisfactory classification result with limited labeled samples, although feature extraction and feature selection can alleviate the problem of lacking labeled samples.

Collecting labeled samples is generally time-consuming and laborious in HSIs. In contrast, unlabeled samples are easier to obtain. This fact fosters the idea of semisupervised learning that can jointly exploit labeled and unlabeled samples to deal with the problem of the curse of dimensionality. Semisupervised algorithms have gained more attention in the domain of HSI classification, and typically achieve better results than supervised approaches, especially with a small number of labeled samples. The traditional semisupervised algorithms include generative model [20], transductive support vector [21], self-training [22], cotraining [23], and graph-based methods [24]. Meanwhile, spatial information is also introduced into semisupervised classification to further improve the performance [25]–[28].

In recent years, deep learning has achieved great success in many challenging applications, especially in the fields of image recognition [29], object detection [30], and speech recognition [31]. In the wake of this success, the use of deep learning has also been introduced into remote sensing data processing [32]. The first attempt in HSI classification can be

Manuscript received June 19, 2018; revised July 17, 2018 and September 1, 2018; accepted September 21, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 41801388 and in part by the Scientific and Technological Project in Henan Province under Grant 152102210014. *(Corresponding author: Bing Liu.)*

The authors are with the Institute of Surveying and Mapping, Information Engineering University, Zhengzhou 450001, China (e-mail: liubing220524@126.com).

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.2018.2872830

found in [33], where a Stacked AutoEncoder is constructed to extract high-level features. An HSI classification method combining convolutional neural network (CNN) and randomized PCA (R-PCA) is then proposed in [34]. Subsequently, deep belief networks [35], recurrent neural networks [36], 1-D CNN [37], 2-D CNN [38]–[41], and 3-D CNN [42]–[44] have also been used for HSI classification and have achieved promising classification results.

Currently, the main problem in deep learning-based approaches is the lack of labeled samples. To deal with such problem, a pixel-pair method has been proposed to significantly increase the number of training samples [45]. A semisupervised CNN has been designed to further improve classification accuracy with limited labeled samples [46]. By constructing a five-layer CNN for classification (C-CNN), CNN's capability of feature learning is explored for HSIs in [47]. Based on C-CNN, a feature-learning-CNN is further constructed to learn spatial–spectral features for different image scenes acquired by a specific hyperspectral sensor. In this manner, C-CNN achieves higher classification accuracy than the compared CNN-based methods. A supervised deep feature extraction method has been proposed to solve the lack of training samples [48]. In addition, a self-taught learning method is used to extract features from unlabeled hyperspectral imagery [49]. Although the aforementioned methods have made great progress, using deep learning methods to classify HSI remains a challenging task, especially for a few labeled samples.

The goal of few-shot learning is to recognize new categories from very few labeled samples [50]. The availability of very few samples challenges the standard fine-tuning practice in deep learning. Recent approaches have made significant progress in few-shot learning. Matching networks [51] use an attention mechanism over a learned embedding of the labeled set of samples (the support set) to predict classes for the unlabeled samples (the query set). An insight of matching networks is that it utilizes sampled mini-batches called episodes during training. Each episode is designed to mimic the few-shot task by subsampling classes as well as samples, which make the training problem more faithful to the test environment. A metalearning approach to few-shot learning is designed in [52]. This model involves training a long short-term memory to produce the updates to a classifier, given an episode, such that it will generalize well to a testing data set. Prototypical networks [53] are also proposed for the problem of few-shot classification. Prototypical networks learn a metric space in which classification can be performed by computing distances to prototype representations of each class. Our method is inspired by the recent success of few-shot learning. Specifically, a CNN model is trained to learn a metric space. In other words, a CNN model is trained to learn how to extract generalizable features by training on sufficiently large quantities of labeled data that are distinct from the target data set. Once trained, the model can extract features from smaller labeled target data sets.

The goal of this paper is to solve the problem of using deep learning methods to classify HSIs with only a few labeled samples. More specifically, a deep few-shot learning (DFSL) method is proposed to perform HSI classification where image annotations are scarce (five labeled samples per class). The DFSL method consists of three parts. The first part is to learn a metric space from a training data set. S-CNN+SVM [48] has demonstrated that different classes can be effectively separated by Euclidean distance. Inspired by this, we adopt Euclidean distance to separate different classes in the metric space. The metric space is then parameterized with a deep 3-D CNN (D-3-D CNN) because deep CNN models have strong nonlinear expression ability and can be trained in end-to-end manner. The purpose of optimizing the loss function is to make the network learn a metric space where sample features have small intraclass spacing and large interclass spacing. Residual learning is also introduced to better train the D-3-D CNN. Therefore, the metric space is finally parameterized with a deep residual 3-D CNN (D-Res-3-D CNN). Once the network is learned, it can be considered as an embedding function. The second part is to extract features of all samples in the testing data set by means of a pretrained D-Res-3-D CNN. The third part is to classify the testing samples. Commonly used classifiers (e.g., SVM) need to use cross validation to determine the optimal parameters. Considering that only a few labeled samples are available, we adopt a simple nearest neighbor (NN) classifier to avoid the search of additional parameters.

The main contributions of this paper are as follows.

1) A DFSL method is proposed to train a network to learn a metric space that causes the samples of the same class to be close to one another. Importantly, such metric space will do the same for classes not seen during training. Consequently, classification on the testing data set can be completed by an NN classifier.
2) A deep 3-D CNN is used to parameterize the metric space. Moreover, residual learning is introduced to better train the network. Such a deep residual 3-D CNN can directly extract spectral–spatial features from data cubes without relying on any preprocessing.
3) Experiments are conducted on four well-known HSI data sets, which demonstrate that the proposed method can outperform conventional semisupervised methods with only a few labeled samples.

The remainder of this paper is structured as follows. In Section II, the proposed DFSL method is described in detail. In Section III, experimental results on four public available HSI data sets are presented. Finally, conclusions are provided in Section IV.

II. Proposed Method

Generally, training deep learning models with only a few labeled samples is challenging. The reason for this is the imbalance between the huge parameter space and a few labeled samples. A good balance between the parameter set and data set size is typically necessary to successfully train a deep learning classifier. Using standard optimization techniques in a few-shot scenario will have the undesirable tendency to extremely overfit the data. Humans, on the other hand, excel at the problem of few-shot classification, where only a few examples of each class are available. The reason is that

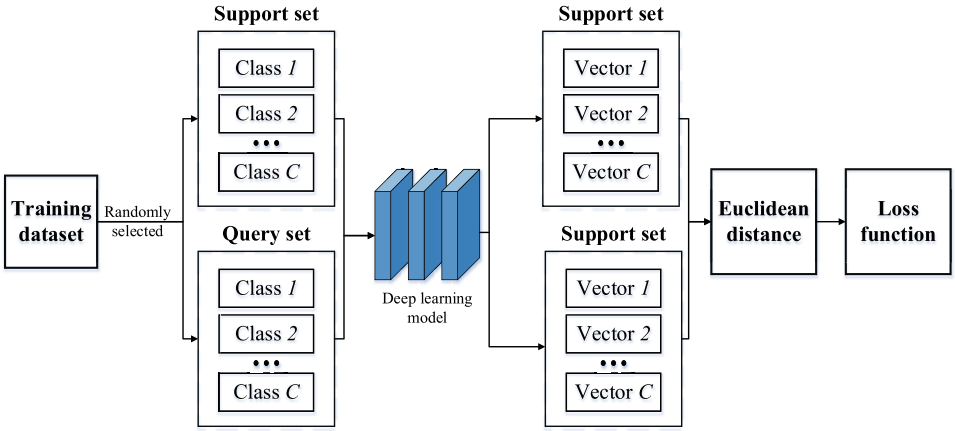


Fig. 1. Visual representation of DFSL method.

humans can exploit their learned knowledge to help solve new problems. Similar to humans, we take advantage of a few precollected training samples to learn a metric space. Subsequently, we demonstrate that such a learned metric space can generalize to new classes and improve the classification performance with only a few labeled samples in the testing data sets.

A. Deep Few-Shot Learning and Training Strategy

To learn a metric space suitable for few-shot classification, we follow Vinyals *et al.* [51] and Snell *et al.* [53] who utilized sampled minibatches called episodes to mimic the few-shot task. The metric space can be considered as an embedding function parameterized with a deep neural network. In other words, a deep neural network is used to compute a representation of each class through an embedding function $f_\phi\ \mathbb{R}^D \rightarrow \mathbb{R}^M$ with learnable parameters ϕ, where D refers to the dimension of the network inputs and M refers to the dimension of the network outputs.

A subset of classes from the training set are randomly selected to form an episode to compute gradients and update the network. As shown in Fig. 1, a subset of samples within each class is then chosen to act as the support set and a subset of the remainder is reserved as the query set. In this paper, only one sample per class is selected as the support set to mimic the situation of small sample classification in the testing data set. The samples of the support set and the query set are fed through the network to extract the embedding features. A distribution over classes for a query sample $\mathbf{x}$ is computed based on a softmax computation over distances to the samples of the support set in the embedding space

$$p_\phi(y=k|\mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k=1}^{N_C} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))} \tag{1}$$

where $\mathbf{c}_k$ denotes the embedding feature of the kth class in the support set, $\mathbf{x}$ denotes the embedding feature of the samples in the query set, y is the label of $\mathbf{x}$, and $d(\cdot)$ is a Euclidean distance function. The loss function is defined as the negative log-probability of the true class k

$$J(\phi) = -\log p_\phi(y=k|\mathbf{x}) = d(f_\phi(\mathbf{x}), \mathbf{c}_k) + \log \sum_{k=1}^{N_C} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k)). \tag{2}$$

Minimizing such a loss function via stochastic gradient-descent is to incentivize the network to learn, not necessarily a good prediction function, but a good metric space in which classes are conceptually similar to each other. More specifically, pseudocode for a training episode procedure is available as Algorithm 1.

B. Deep 3-D CNN

1) 3-D Convolution: CNN is initially designed for 2-D shape recognition and has achieved success in visual image classification. However, CNN has limitations in analyzing 3-D data such as video and HSIs. Therefore, CNN requires dimensionality reduction preprocessing before classifying the HSIs. However, dimensionality reduction will lose details that are generally helpful to distinguish different classes. 3-D CNN, capturing the motion information encoded in multiple contiguous frames, was first introduced to address the problems of video analysis [54]. Similarly, 3-D CNN can also capture the spectral–spatial correlation information encoded in the multiple contiguous bands of HSIs, which has been demonstrated in [42]–[44] and [55]. To this end, DFSL is parameterized with a deep 3-D CNN.

A 3-D CNN is constructed based on the 3-D convolutional operation. The 3-D convolution is similar to the 2-D convolution. An illustration of 3-D convolution is shown in Fig. 2. The value at position (x, y, z) is computed by

$$v_{ij}^{xyz} = f\left(b_{ij} + \sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right) \tag{3}$$

where i is the 3-D convolutional layer index, j is the feature map index, P_i, Q_i, R_i are the height, width, and spectral

Algorithm 1 Training Episode Procedure. N and K Denote the Sample Number and the Classes Number of the Training Data Set, Respectively, $N_C \leq K$ Denotes the Classes Number of Each Episode, N_Q Denotes the Sample Number Per Class in the Query Set, and RandomSample(S, M) Denotes Randomly Choosing M Elements From Set S, Without Replacement

Input: Training data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$, $(\mathbf{x}_i, y_i = k)$.
Output: The loss for a training episode.

$V \leftarrow \text{RandomSample}(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for an episode
for k in $\{1, \ldots, N_C\}$ **do**
  $S_k \leftarrow \text{RandomSample}(\mathcal{D}_{V_k}, 1)$ ▷ Select support samples
  $Q_k \leftarrow \text{RandomSample}(\mathcal{D}_{V_k} \backslash S_k, N_Q)$ ▷ Select query samples
  $\mathbf{c}_k \leftarrow f_\phi(\mathbf{x}_i)$ ▷ Compute the embedding features of support samples
end for
$J \leftarrow 0$ ▷ Initialize loss
for k in $\{1, \ldots, N_C\}$ **do**
  for $(\mathbf{x}, y)$ in Q_k **do**
    $J \leftarrow J + \left[d(f_\phi(\mathbf{x}), \mathbf{c}_k) + \log \sum_{k=1}^{N_C} \exp(-d(f_\phi(\mathbf{x}, \mathbf{c}_k)) \right]$ ▷ Update loss
  end for
end for

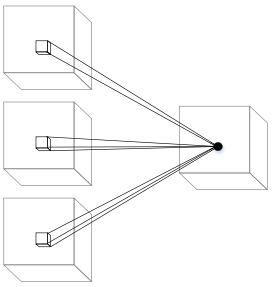


Fig. 2. Illustration of 3-D convolution.

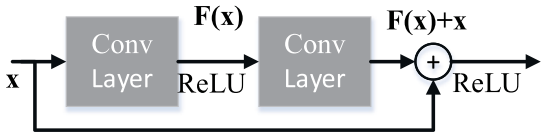


Fig. 3. Illustration of residual learning.

dimensions of a 3-D kernel, $f(\cdot)$ is an activation function, and b_{ij} is the bias. m indicates the indexes over the set of feature maps, and w_{ijm}^{pqr} indicates the value of position (p, q, r) associated with the mth feature map.

2) Residual Learning: Compared with traditional machine learning methods, the deep network with more hidden layers has strong feature learning capability and feature representation. Recent studies [56] have revealed the crucial importance of network depth. However, deep neural networks often suffer degradation with increasing network depth, given limited training samples. A deep residual learning method has been proposed to solve the degradation problem of deep networks [57]. An illustration of residual learning is shown in Fig. 3, where $\mathbf{x}$ is the input of a neural network and $\mathbf{F}(\mathbf{x})$ is an underlying mapping to fit so that a residual mapping $\mathbf{H}(\mathbf{x}) := \mathbf{F}(\mathbf{x}) - \mathbf{x}$ can be defined. Then, the original function $\mathbf{F}(\mathbf{x})$ can be represented as $\mathbf{H}(\mathbf{x}) + \mathbf{x}$. In the feedforward neural network, $\mathbf{H}(\mathbf{x}) + \mathbf{x}$ can be considered as the addition of the main path $\mathbf{H}(\mathbf{x})$ and a shortcut $\mathbf{x}$. Therefore, no additional parameters are introduced, which does not affect the complexity of the network, and the backpropagation algorithm can still be used to train the network. The idea of residual learning is to introduce a shortcut based on traditional network architecture. Such a shortcut can skip around the connections of some layers and add an alternate main path. Therefore, the underlying error in the training process can be transmitted to the upper level through the shortcut, which alleviates the notorious problem of vanishing/exploding gradients caused by too many layers and simplifies the training of deep networks.

3) Network Architecture: As shown in Fig. 4, a deep 3-D CNN, with two residual blocks, two pooling layers, and a convolutional layer is designed as the embedding function. *Conv* represents a 3-D convolutional layer with a $3 \times 3 \times 3$ kernel. *Pooling* represents a 3-D max-pooling function with a stride of $2 \times 2 \times 4$. The rectified linear units (ReLU) [58] activation function has faster convergence speed and can be implemented by simply thresholding a matrix of activations at zero, compared with the traditional sigmoid and tanh activation functions that involve expensive operations (exponentials, etc.). Therefore, ReLU is adopted as the activation function. The ReLU activation function is computed by

$$f(x) = \max(0, x).$$

In Fig. 4, the dashed box is a residual block. A pooling layer will cause different dimensions of shortcuts and main paths. Therefore, no pooling layer is used in the residual block. In contrast, each residual block is connected with a 3-D max-pooling layer to reduce computation and aggregate features. Owing to the high spectral dimension of the input cube, the stride along the spectral dimension is set as 4 and the stride along the spatial dimension is set as 2. Finally, the feature maps are flattened into 1-D vectors. The designed deep 3-D CNN is trained using the strategy shown in Algorithm 1. In this manner, the designed network can learn a metric

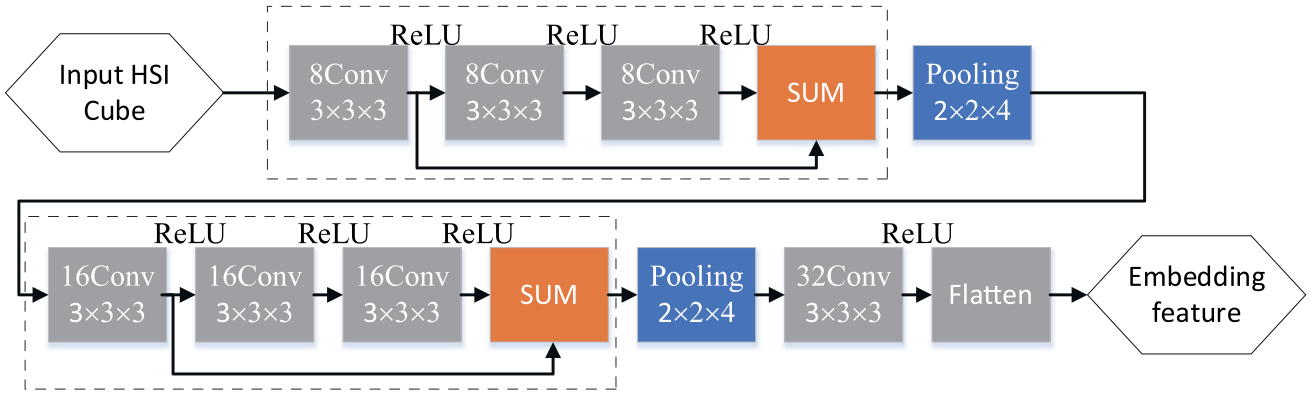


Fig. 4. Architecture of deep 3-D CNN.

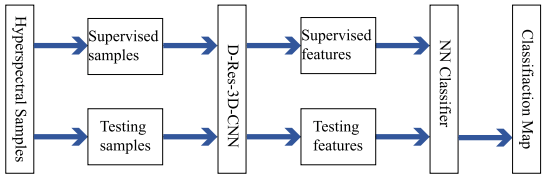


Fig. 5. Flowchart of classification on testing data set.

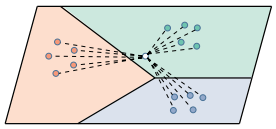


Fig. 6. Classification in the testing data set (five labeled samples per class).

space where samples with the same class are close to each other.

C. Classification With the Nearest Neighbor

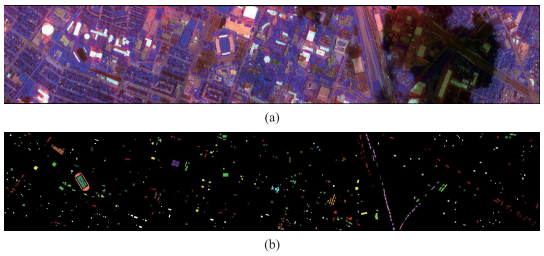
As shown in Fig. 5, the classification of the testing data sets mainly includes three steps:

1) extracting embedded features via a pretrained deep residual 3-D CNN;
2) computing the Euclidean distance between the labeled samples and the samples to be classified;
3) determining the final labels by means of an NN classifier.

In fact, the designed deep residual 3-D CNN can be considered as an embedding function after training. In the classification process of testing data sets, all samples are fed through the pretrained deep residual 3-D CNN to extract the features. Then, a few labeled samples are randomly selected as the supervised samples. The trained network enables similar samples to be close to each other in the embedding space. Therefore, the testing samples can be classified by simple NN analysis, as shown in Fig. 6. It is important to note that the training data sets and the testing data sets are independent of each other. Finally, the classification maps, produced by the labels of the testing samples, are matched with the ground-truth maps to assess the different classification methods.

TABLE I
DETAILS OF FOUR TRAINING DATA SETS. KENNEDY SPACE CENTER (KSC), GSD (IN METERS), SPATIAL SIZE (PIXEL), SPECTRAL RANGE (IN NANOMETERS), AND AIRBORNE VISIBLE INFRARED IMAGING SPECTROMETER (AVIRIS)

	Houston	**Botswana**	**KSC**	**Chikusei**
Spatial size	349 × 1905	1476 × 256	512 × 614	2517 × 2335
Spectral range	380-1050	400-2500	400-2500	363-1018
No. of bands	144	145	176	128
GSD	2.5	30	18	2.5
Sensor type	ITRES-CASI 1500	EO-1	AVIRIS	Hyperspec -VNIR-C
Areas	Houston	Botswana	Florida	Chikusei
No. of classes	31	14	13	19



(a)

(b)

Fig. 7. Houston data set. (a) Pseudocolor image. (b) Ground-truth map.

III. EXPERIMENTS AND DISCUSSION

All experiments were carried out on a laptop with an Intel Core i7-5700HQ, 2.7 GHz and an Nvidia GeForce GTX 970M. The laptop's memory is 32 GB.

A. Experimental Data Sets

1) Training Data Sets: To train the deep 3-D CNN, four publicly available HSI data sets were collected. The details of the four data sets are listed in Table I. The Chikusei data set is provided in [59]. The number of bands listed in Table I denotes the number of valid bands for classification. Samples from four data sets can be seen in Figs. 7–10. The data sets cover different areas and different ground sample

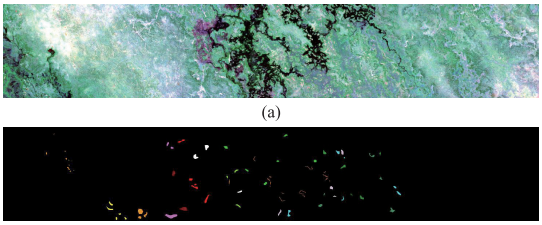


Fig. 8. Botswana data set. (a) Pseudocolor image. (b) Ground-truth map.

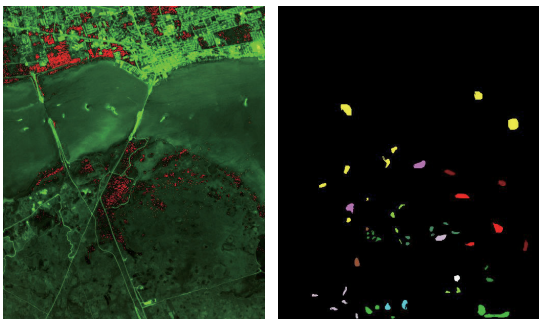


Fig. 9. KSC data set. (a) Pseudocolor image. (b) Ground-truth map.

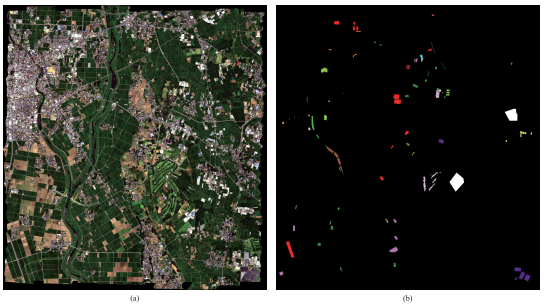


Fig. 10. Chikusei data set. (a) Pseudocolor image. (b) Ground-truth map.

distances (GSDs) that ensure the diversity of the samples. There are a total of 77 classes in the four testing data sets. However, the classes with fewer than 200 samples were discarded to ensure the same number for each training class. Therefore, 44 classes were available for training. Finally, we randomly selected 200 samples of each class (44 classes). We also test 75 classes (100 labeled samples per class) and 35 classes (400 labeled samples per class). The classification accuracy will decrease a little [University of Pavia (UP) data set:78.39 $\pm$ 1.64, 77.71 $\pm$ 2.49]. Note that 100 bands were selected via the graph representation-based band selection [18] for training to ensure the same input dimensions of the D-Res-3-D CNN. This ensures the reasonableness of the selected band. We followed [46], [55], and [48] to select 9×9 neighborhoods as the input of the network. In other words, each sample is represented by a $9 \times 9 \times 100$ data cube.

2) Testing Data Sets: To demonstrate the classification power of the proposed method, experiments are conducted on four well-known HSI data sets including the UP, the Pavia Center (PC), the Indian Pines (IP), and the Salinas. The details of the four testing data sets are listed in Table II. The first 100 bands of the UP and the PC data sets are used to ensure the same input dimensionality. Similarly, the first 200 bands of the IP and the Salinas data sets are used for classification. In the testing of the IP and Salinas data sets, the first 100 bands and the last 100 bands pass through the pretrained D-Res-3-D CNN to extract features. The features extracted from the first 100 bands and the last 100 bands are then connected to form a new feature vector that is used as the final feature for classification.

TABLE II
DETAILS OF FOUR TESTING DATA SETS. UP, PC, IP GSD (IN METERS), SPATIAL SIZE (IN PIXELS), SPECTRAL RANGE (IN NANOMETERS), REFLECTIVE OPTICS SYSTEM IMAGING SPECTROMETER (ROSIS), AVIRIS

	UP	**PC**	**IP**	**Salinas**
Spatial size	610×340	1096×715	145×145	512×217
Spectral range	430-860	430-860	400-2500	400-2500
No. of bands	103	102	200	204
GSD	1.3	1.3	20	3.7
Sensor type	ROSIS	ROSIS	AVIRIS	AVIRIS
Areas	Pavia	Pavia	Indiana	California
No. of classes	9	9	16	16

B. Experimental Setting

The network for DFSL can be an arbitrary neural network. In this paper, a deep residual 3-D CNN (D-Res-3-D CNN) is designed. We also tested other neural network architectures, such as a 2-D CNN, a deep 2-D CNN (D-2-D CNN), a deep residual 2-D CNN (D-Res-2-D CNN), a 3-D CNN, and a deep 3-D CNN (D-3-D CNN). Note that the 2-D CNN, D-2-D CNN, and D-Res-2-D CNN require dimensionality reduction processing. According to the findings in 2-D CNN [56], small receptive fields of 3×3 convolution kernels with deeper architectures generally yield better results. Li *et al.* [43] have also demonstrated that small $3 \times 3 \times 3$ kernels are an effective choice for 3-D CNN in HSI classification. Inspired by this, we fix the size of the 2-D kernels and 3-D convolution kernels to 3×3 and $3 \times 3 \times 3$, respectively. More detailed parameters for different networks are summarized in Tables III–V.

Five labeled samples per class are randomly selected as the supervised samples. Overall accuracy with different neural network architectures on the four testing data sets is shown in Table VI. In all cases, the network architectural complexity increases as the number of parameters increases. From Table VI, the classification accuracy increases with increasing network complexity, which demonstrates the importance of the depth of the network. Furthermore, the introduction of residual learning can also improve the classification performance.

In general, the number of kernels will influence the classification performance of D-Res-3-D CNN. The number of kernels for the first residual block is set to 2, 4, 8, 16, and 32, respectively. The number of kernels for the second residual block is set to double that for the first residual block. Overall accuracy with different number of kernels is listed in Table VII. It is observed that a small number of kernels

TABLE III
PARAMETERS OF 2-D CNN AND 3-D CNN

2D-CNN				3D-CNN			
Layer Name	**Output Shape**	**Filter Size**	**Padding**	**Layer Name**	**Output Shape**	**Filter Size**	**Padding**
Input	$9\times9\times3$	N/A	N	Input	$9\times9\times100$	N/A	N
Convolution 1	$9\times9\times8$	$3\times3\times8$	Y	Convolution 1	$9\times9\times100\times8$	$3\times3\times3\times8$	Y
Max Pooling 1	$5\times5\times8$	2×2	N	Max Pooling 1	$5\times5\times25\times8$	$2\times2\times4$	N
Convolution 2	$5\times5\times16$	$3\times3\times16$	Y	Convolution 2	$5\times5\times25\times16$	$3\times3\times3\times16$	Y
Max Pooling 2	$3\times3\times16$	2×2	N	Max Pooling 2	$3\times3\times7\times16$	$2\times2\times4$	N
Flatten	144	N/A	N	Convolution 3	$1\times1\times5\times32$	$3\times3\times3\times32$	N
Output	144	N/A	N	Flatten	160	N/A	N
				Output	160	N/A	N

TABLE IV
PARAMETERS OF D-2-D CNN AND D-3-D CNN

D-2D-CNN				D-3D-CNN			
Layer Name	**Output Shape**	**Filter Size**	**Padding**	**Layer Name**	**Output Shape**	**Filter Size**	**Padding**
Input	$9\times9\times3$	N/A	N	Input	$9\times9\times100$	N/A	N
Convolution 1	$9\times9\times8$	$3\times3\times8$	Y	Convolution 1	$9\times9\times100\times8$	$3\times3\times3\times8$	Y
Convolution 2	$9\times9\times8$	$3\times3\times8$	Y	Convolution 2	$9\times9\times100\times8$	$3\times3\times3\times8$	Y
Convolution 3	$9\times9\times8$	$3\times3\times8$	Y	Convolution 3	$9\times9\times100\times8$	$3\times3\times3\times8$	Y
Max Pooling 1	$5\times5\times8$	2×2	N	Max Pooling 1	$5\times5\times25\times8$	$2\times2\times4$	N
Convolution 4	$5\times5\times16$	$3\times3\times16$	Y	Convolution 4	$5\times5\times25\times16$	$3\times3\times3\times16$	Y
Convolution 5	$5\times5\times16$	$3\times3\times16$	Y	Convolution 5	$5\times5\times25\times16$	$3\times3\times3\times16$	Y
Convolution 6	$5\times5\times16$	$3\times3\times16$	Y	Convolution 6	$5\times5\times25\times16$	$3\times3\times3\times16$	Y
Max Pooling 2	$3\times3\times16$	2×2	N	Max Pooling 2	$3\times3\times7\times16$	$2\times2\times4$	N
Flatten	144	N/A	N	Convolution 7	$1\times1\times5\times32$	$3\times3\times3\times32$	N
Output	144	N/A	N	Flatten	160	N/A	N
				Output	160	N/A	N

TABLE V
PARAMETERS OF D-RES-2-D CNN AND D-RES-3-D CNN

D-Res-2D-CNN				D-Res-3D-CNN			
Layer Name	**Output Shape**	**Filter Size**	**Padding**	**Layer Name**	**Output Shape**	**Filter Size**	**Padding**
Input	$9\times9\times3$	N/A	N	Input	$9\times9\times100$	N/A	N
Convolution 1	$9\times9\times8$	$3\times3\times8$	Y	Convolution 1	$9\times9\times100\times8$	$3\times3\times3\times8$	Y
Convolution 2	$9\times9\times8$	$3\times3\times8$	Y	Convolution 2	$9\times9\times100\times8$	$3\times3\times3\times8$	Y
Convolution 3	$9\times9\times8$	$3\times3\times8$	Y	Convolution 3	$9\times9\times100\times8$	$3\times3\times3\times8$	Y
Shortcut	Convolution 1 + Convolution 3			Shortcut	Convolution 1 + Convolution 3		
Max Pooling 1	$5\times5\times8$	2×2	N	Max Pooling 1	$5\times5\times25\times8$	$2\times2\times4$	N
Convolution 4	$5\times5\times16$	$3\times3\times16$	Y	Convolution 4	$5\times5\times25\times16$	$3\times3\times3\times16$	Y
Convolution 5	$5\times5\times16$	$3\times3\times16$	Y	Convolution 5	$5\times5\times25\times16$	$3\times3\times3\times16$	Y
Convolution 6	$5\times5\times16$	$3\times3\times16$	Y	Convolution 6	$5\times5\times25\times16$	$3\times3\times3\times16$	Y
Shortcut	Convolution 4 + Convolution 6			Shortcut	Convolution 4 + Convolution 6		
Max Pooling 2	$3\times3\times16$	2×2	N	Max Pooling 2	$3\times3\times7\times16$	$2\times2\times4$	N
Flatten	144	N/A	N	Convolution 7	$1\times1\times5\times32$	$3\times3\times3\times32$	N
Output	144	N/A	N	Flatten	160	N/A	N
				Output	160	N/A	N

(e.g., 2, 4) greatly decrease the classification accuracy and a large number of kernels (e.g., 32) will also decrease the classification accuracy. It is appropriate to set the number of kernels to 8 or 16. In addition, a large number of kernels (e.g., 16) will increase the training time greatly. Therefore, the number of kernels for the first residual block is set to 8.

The proposed DFSL network is sensitive to the number of episodes, the learning rate, the number of classes for each episode N_C, and the number of query samples for each class N_Q. The number of episodes is empirically set to 10 000, which is sufficient to fully train the network. Learning rate is an important parameter that can greatly affect the

TABLE VI
OVERALL ACCURACY (%) WITH DIFFERENT NEURAL NETWORK ARCHITECTURES (AVERAGE OF 10 RUNS ± STANDARD DEVIATION)

	2D-CNN	D-2D-CNN	D-Res-2D-CNN	3D-CNN	D-3D-CNN	D-Res-3D-CNN
UP	66.76 ± 3.70	72.68 ± 4.03	76.51 ± 2.18	73.47 ± 4.87	76.58 ± 3.82	80.81 ± 3.12
PC	95.51 ± 0.60	96.01 ± 0.57	96.11 ± 0.39	95.92 ± 0.81	96.09 ± 0.73	96.39 ± 0.59
IP	58.34 ± 2.19	58.97 ± 1.31	63.87 ± 0.89	62.46 ± 0.58	65.89 ± 0.96	67.84 ± 1.29
Salinas	83.32 ± 2.34	85.81 ± 1.93	88.19 ± 1.31	83.95 ± 1.94	86.50 ± 1.33	88.40 ± 1.54

TABLE VII
OVERALL ACCURACY (%) WITH DIFFERENT NUMBER OF KERNELS (AVERAGE OF 10 RUNS ± STANDARD DEVIATION)

	2	4	8	16	32
UP	61.27 ± 12.47	72.61 ± 5.48	80.81 ± 3.12	81.08 ± 2.20	80.55 ± 2.56
PC	93.82 ± 0.75	95.46 ± 0.82	96.39 ± 0.59	96.17 ± 0.48	96.01 ± 0.62
IP	49.74 ± 13.41	63.50 ± 1.63	67.84 ± 1.29	68.14 ± 0.91	67.53 ± 1.56
Salinas	81.32 ± 5.22	84.68 ± 2.27	88.40 ± 1.54	88.41 ± 1.28	85.99 ± 1.14

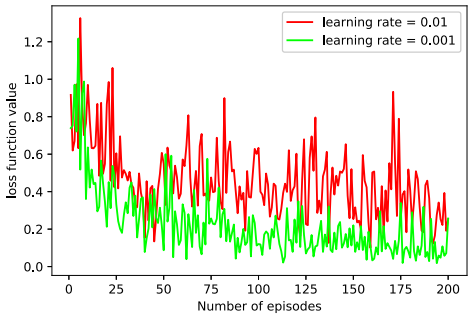


Fig. 11. Curves of loss function for different learning rates.

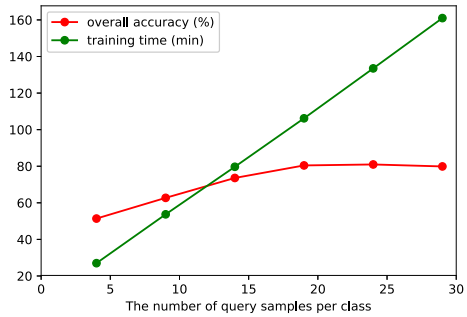


Fig. 12. Overall accuracy (%) and training time (in minutes) with different numbers of query samples per class for the UP data set (Average of 10 runs).

final classification performance and determine the convergence speed of neural networks. Learning rate is set to 0.01 and 0.001 for analysis. The curve of the loss function is shown in Fig. 11. We observed that a large learning rate (e.g., 0.01) causes the loss function value to fluctuate significantly, which may result in divagation of the network. In contrast, a small learning rate (e.g., 0.001) can fully reduce the loss function value. Therefore, the learning rate is set to 0.001.

The maximum number of classes for the four testing data sets is 16. In general, the number of classes per episode N_C should be set to be larger than 16. Considering that only 44 classes are available for training, N_C was finally set to 20. N_C is also set to be 5, 10, 15, 20, and 25 for analysis. We find that a small N_C (e.g., 5, 10) will greatly reduce the classification performance. Theoretically, the larger value of N_Q, the better it can mimic the small sample of classes in the testing data sets. On the one hand, the computing power of the laptop is limited. On the other hand, the available training samples are also limited. Taking the UP data set as an example, overall accuracy and training times with different numbers of query samples per class are shown in Fig. 12. N_Q is set to 4, 9, 14, 19, 24, and 29, respectively. Note that training time increases linearly with N_Q. However, accuracy will soon saturate as N_Q increases. Consequently, N_Q is set to 19, and promising results are obtained.

C. Compared With Semisupervised Methods

In this section, the proposed method is compared with standard SVM and several semisupervised methods provided in [25]. The semisupervised methods include the Laplacian SVM (LapSVM) [60], the Transductive SVM (TSVM) [21], the spatial–contextual semisupervised SVM (SCS^{3}VM) [61], and the spatial–spectral label propagation based on the SVM [25]. For the UP and IP data sets, the results are also compared with KNN+SNI [27] and MLR+RS [62]. In addition, a related work SVM+S-CNN (Siamese CNN) presented in [48] is also analyzed.

Because the main target of this paper is to address the small-scale classification problem, the classification methods were evaluated in each testing data set with varying numbers of supervised samples per class (5–25 in increments of 5). We also tested the proposed method with an SVM classifier to further demonstrate the advantage of NN classifier. Note that the optimal hyperplane parameters of SVM, including the penalty value and the spread of the RBF kernel, are determined by fivefold cross validation.

The summary of overall accuracy scores is given in Tables VIII–XI. Four observations can be made from Tables VIII–XI as follows.

TABLE VIII

OVERALL ACCURACY (%) OF THE DIFFERENT METHODS FOR THE UP DATA SET (AVERAGE OF 10 RUNS ± STANDARD DEVIATION; L IS THE NUMBER OF SUPERVISED SAMPLES FOR EACH CLASS; BOLD VALUES REPRESENT THE BEST ACCURACY AMONG THE METHODS IN EACH CASE)

Method	L=5	L=10	L=15	L=20	L=25
SVM [25]	53.73 ± 1.30	61.53 ± 1.14	60.43 ± 0.94	64.89 ± 1.14	68.01 ± 2.62
LapSVM [25]	65.72 ± 0.34	68.26 ± 2.20	68.34 ± 0.29	65.91 ± 0.45	68.88 ± 1.34
TSVM [25]	63.43 ± 1.22	63.73 ± 0.45	68.45 ± 1.07	73.72 ± 0.27	69.96 ± 1.39
SCS^{3}VM [25]	56.76 ± 2.28	64.25 ± 0.40	66.87 ± 0.37	68.24 ± 1.18	69.45 ± 2.19
SS-LPSVM [25]	69.60 ± 2.30	75.88 ± 0.22	80.67 ± 1.21	78.41 ± 0.26	85.56 ± 0.09
KNN+SNI [27]	70.21 ± 1.29	78.97 ± 2.33	82.56 ± 0.51	85.18 ± 0.65	86.26 ± 0.37
MLR+RS [62]	69.73 ± 3.15	80.30 ± 2.54	84.10 ± 1.94	83.52 ± 2.13	87.97 ± 1.69
SVM+S-CNN	23.68 ± 6.34	66.64 ± 2.37	68.35 ± 4.70	78.43 ± 1.93	72.87 ± 7.36
DFSL+NN	**80.81 ± 3.12**	**84.79 ± 2.27**	86.68 ± 2.61	89.59 ± 1.05	91.11 ± 0.83
DFSL+SVM	72.57 ± 3.93	84.56 ± 1.83	**87.23 ± 1.38**	**90.69 ± 1.29**	**93.08 ± 0.92**

TABLE IX

OVERALL ACCURACY (%) OF THE DIFFERENT METHODS FOR THE PC DATA SET (AVERAGE OF 10 RUNS ± STANDARD DEVIATION; L IS THE NUMBER OF SUPERVISED SAMPLES FOR EACH CLASS; BOLD VALUES REPRESENT THE BEST ACCURACY AMONG THESE METHODS IN EACH CASE)

Method	L=5	L=10	L=15	L=20	L=25
SVM [25]	79.51 ± 0.64	89.33 ± 0.28	90.88 ± 1.72	92.79 ± 0.42	91.23 ± 0.83
LapSVM [25]	88.03 ± 0.80	89.91 ± 0.97	91.66 ± 0.40	92.09 ± 1.26	92.44 ± 0.63
TSVM [25]	68.00 ± 0.56	67.89 ± 0.69	72.94 ± 0.63	76.78 ± 1.40	77.85 ± 2.55
SCS^{3}VM [25]	91.24 ± 0.70	91.72 ± 0.38	93.68 ± 1.00	93.79 ± 1.00	93.70 ± 0.02
SS-LPSVM [25]	91.68 ± 1.17	93.28 ± 0.70	95.76 ± 0.36	95.76 ± 0.56	95.75 ± 0.64
KNN+SNI [27]	90.38 ± 1.59	92.97 ± 1.13	92.73 ± 0.81	95.24 ± 0.95	96.07 ± 0.86
MLR+RS [62]	89.63 ± 0.86	93.41 ± 0.79	94.22 ± 0.81	94.51 ± 0.73	95.66 ± 0.58
SVM+S-CNN	57.44 ± 5.01	81.57 ± 3.30	91.47 ± 0.61	90.93 ± 3.75	96.04 ± 1.47
DFSL+NN	**96.39 ±0.59**	**97.79 ± 0.51**	**98.13 ± 0.49**	**98.24 ± 0.36**	98.15 ± 0.23
DFSL+SVM	94.97± 0.94	96.21 ± 0.68	97.10 ± 0.89	97.82 ± 0.54	**98.33 ± 0.45**

TABLE X

OVERALL ACCURACY (%) OF THE DIFFERENT METHODS FOR THE IP DATA SET (AVERAGE OF 10 RUNS ± STANDARD DEVIATION; L IS THE NUMBER OF SUPERVISED SAMPLES FOR EACH CLASS; BOLD VALUES REPRESENT THE BEST ACCURACY AMONG THESE METHODS IN EACH CASE)

Method	L=5	L=10	L=15	L=20	L=25
SVM [25]	50.23 ± 1.74	55.56 ± 2.04	58.58 ± 0.80	62.93 ± 0.64	65.12 ± 0.63
LapSVM [25]	52.31 ± 0.67	56.36 ± 0.71	59.99 ± 0.65	64.13 ± 1.19	65.36 ± 0.62
TSVM [25]	62.57 ± 0.23	63.45 ± 0.17	65.42 ± 0.02	64.43 ± 0.20	67.68 ± 1.67
SCS^{3}VM [25]	55.42 ± 0.35	60.86 ± 5.08	67.24 ± 0.47	68.34 ± 1.57	72.42 ± 1.21
SS-LPSVM [25]	56.95 ± 0.95	64.74 ± 0.39	78.76 ± 0.04	80.29 ± 0.80	84.11±0.08
KNN+SNI [27]	56.39 ± 1.03	74.88 ± 0.54	78.92 ± 0.61	80.08 ± 0.59	82.60 ± 0.73
MLR+RS [62]	55.38 ± 3.98	69.28 ± 2.63	75.15 ± 1.43	77.68 ± 1.57	79.32 ± 0.88
SVM+S-CNN	10.02 ± 1.48	17.71 ± 4.90	44.00 ± 5.73	45.08 ± 10.90	51.30 ± 10.41
DFSL+NN	**67.84 ±1.29**	**76.49 ± 1.44**	78.62 ± 1.59	81.74 ± 0.95	84.74 ±1.44
DFSL+SVM	64.58 ± 2.78	75.53 ± 1.89	**79.98 ± 2.23**	**83.01 ± 1.67**	**85.47 ± 1.21**

1) The overall accuracy of the classification methods generally rises as L increases.
2) The semisupervised methods (LapSVM, TSVM, SCS^{3}VM, and SS-LPSVM) generally outperform the standard SVM.
3) SVM+S-CNN shows very poor performance with few supervised samples.
4) The proposed DFSL method with NN classifier (DFSL+NN) and SVM classifier (DFSL+SVM) both outperforms other methods. In particular, for a small sample of 5 labeled samples per class, DFSL+NN achieves excellent classification performance.

It is easy to understand the first two observations, as increasing the number of labeled samples and using unlabeled sam-

TABLE XI

OVERALL ACCURACY (%) OF THE DIFFERENT METHODS FOR THE SALINAS DATA SET (AVERAGE OF 10 RUNS ± STANDARD DEVIATION; L IS THE NUMBER OF SUPERVISED SAMPLES FOR EACH CLASS; BOLD VALUES REPRESENT THE BEST ACCURACY AMONG THESE METHODS IN EACH CASE)

Method	L=5	L=10	L=15	L=20	L=25
SVM [25]	73.90 ± 1.91	75.62 ± 1.73	79.08 ± 1.45	77.89 ± 1.20	78.05 ± 1.49
LapSVM [25]	75.31 ± 2.31	76.34 ± 1.77	77.93 ± 2.42	79.40 ± 0.73	80.56 ± 1.33
TSVM [25]	60.43 ± 1.40	67.47 ± 1.05	69.12 ± 1.32	71.03 ± 1.78	71.83 ± 1.16
SCS^{3}VM [25]	74.12 ± 2.44	78.49 ± 2.02	81.83 ± 0.93	81.22 ± 1.27	77.08 ± 0.80
SS-LPSVM [25]	86.79 ± 1.75	90.36 ± 1.35	90.86 ± 1.36	91.77 ± 0.96	92.11 ± 1.07
KNN+SNI [27]	80.39 ± 1.58	84.64 ± 1.54	86.94 ± 1.52	88.28 ± 1.49	87.64 ± 1.72
MLR+RS [62]	78.29 ± 2.16	85.03 ± 1.43	87.20 ± 1.74	88.76 ± 1.87	89.42 ± 0.85
SVM+S-CNN	12.66 ± 2.75	50.04 ± 14.34	60.72 ± 4.85	70.30 ± 2.61	71.62 ± 12.05
DFSL+NN	**88.40 ± 1.54**	**89.86 ± 1.69**	**92.15 ± 1.24**	92.69 ± 0.98	93.61 ± 0.83
DFSL+SVM	85.58 ± 1.87	89.73 ± 1.24	91.21 ± 1.64	**93.42 ± 1.25**	**94.28 ± 0.80**

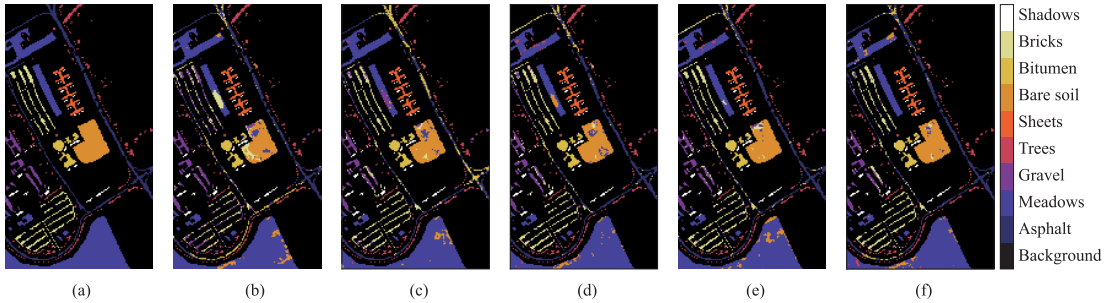


Fig. 13. Classification maps resulting from DFSL+NN for the UP data set. (a) Ground-truth map. (b)–(f) Examples of the classification maps with 5, 10, 15, 20, and 25 labeled samples, respectively.

ples improve the classification performance. The fundamental operation of SVM+S-CNN and DFSL+NN is similar. Their purpose is to learn a metric space where classification can be conducted by a simple classifier (e.g., linear SVM or NN). The difference is that SVM+S-CNN uses a Siamese CNN to learn a metric space on a specific data set. Therefore, SVM+S-CNN requires sufficient labeled samples to enable its generalization ability (200 labeled samples per class). Using few samples causes overfitting to the training samples, which leads to poor classification of unknown classes. In contrast, the proposed method trains a network to learn a metric space from a collected data set. The training data set with 44 classes enables the generalization of new, unknown classes in the testing data sets. Furthermore, training by episodes can mimic the few-shot sample classification. Consequently, the proposed method can achieve a better classification performance compared with other methods including SVM, LapSVM, SCS^{3}VM, SS-LPSVM, KNN+SNI, MLR+RS, and SVM+S-CNN.

As for a small-scale classification problem (e.g., 5, 10, and 15 labeled samples per class), the SVM classifier easily overfits the supervised samples. Therefore, the classification accuracy of the SVM classifier is lower than an NN classifier. Having more labeled samples can increase the overall accuracy of both SVM and NN. The classification performance of SVM and NN is relatively close when $L = 25$. Per the above-mentioned observations, using an NN classifier can avoid the search of additional parameters when only a few labeled samples are available. Moreover, it can reduce the risk of overfitting. Thus, an NN classifier can outperform an SVM classifier in the case of very few labeled samples (e.g. 5 or 10 labeled samples per class).

In order to better observe the classification results of DFSL+NN, graphical visualizations are presented in Figs. 13–16. Each presents a ground-truth map and classification maps resulting from DFSL+NN with 5, 10, 15, 20, and 25 labeled samples, respectively. The classification noise decreases with the increase of L, which indicates the effectiveness of DFSL+NN. In fact, these maps are consistent with the results listed in Tables VIII–XI. To further validate the classification performance of DFSL+NN, we carried out paired t-test calculations on overall accuracy. According to the results from Tables VIII–XI, DFSL+NN outperforms the compared methods at a confidence level of 97.5%, which also demonstrates that the increase in overall accuracy is statistically significant.

D. Compared With CNN-Based Methods

In this section, experimental results over the four testing data sets are compared with several CNN-based methods to further demonstrate the performance of the proposed method. The compared methods include CNN [37], R-PCA-CNN [34], CNN-PPF [45], and C-CNN [47]. In addition, D-Res-3-D CNN is also trained as a supervised classifier to demonstrate its classification performance. For a fair comparison, 200 labeled

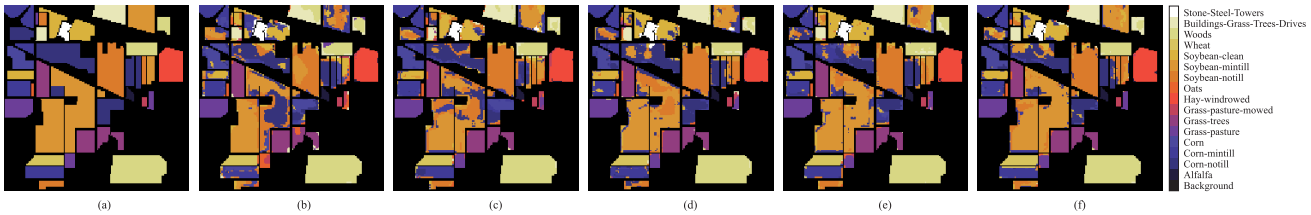


Fig. 14. Classification maps resulting from DFSL+NN for the IP data set. (a) Ground-truth map. (b)–(f) Examples of the classification maps with 5, 10, 15, 20, and 25 labeled samples, respectively.

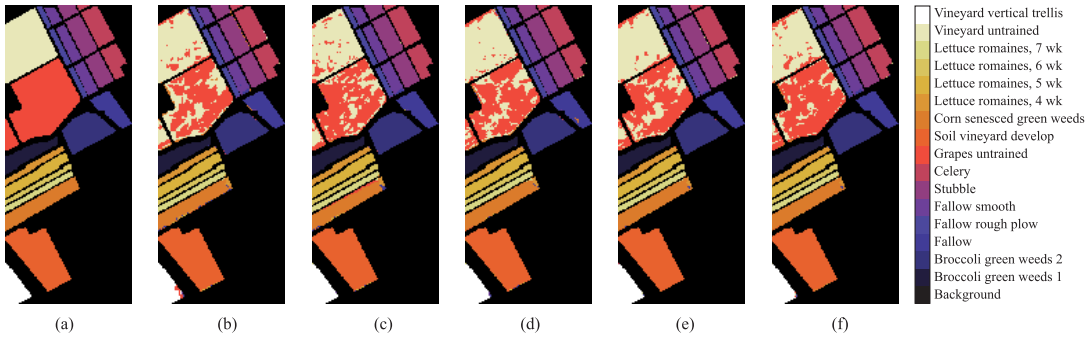


Fig. 15. Classification maps resulting from DFSL+NN for the Salinas data set. (a) Ground-truth map. (b)–(f) Examples of the classification maps with 5, 10, 15, 20, and 25 labeled samples, respectively.

samples per class are randomly selected as the supervised samples for all methods. Note that there are 16 different land-cover classes in the original ground truth of the IP data set. However, only nine classes are used in CNN [37], CNN-PPF [45], and C-CNN [47] so as to avoid a few classes that have very few training samples. Therefore, only nine classes of the IP data set are used for experiments so as to ensure the same experimental conditions.

Class-specific accuracy, OA, average accuracy (AA), and κ of different techniques for the four testing data sets are listed in Tables XII–XV. First, D-Res-3-D CNN could outperform CNN [37] and R-PCA-CNN [34], which demonstrate the effectiveness of D-Res-3-D CNN. Furthermore, we can see that DFSL+NN and DFSL+SVM could provide competitive results compared with the state-of-the-art CNN-based methods. This demonstrates that the proposed method can also achieve promising results when the labeled samples are relatively adequate. In particular, DFSL+SVM has the best performance on the UP, IP, and Salinas data sets. This indicates that DFSL+SVM can achieve better results than DFSL+NN, in the case of sufficient labeled samples.

E. Execution Time Analysis

The classification framework comprises three parts: training the D-Res-3-D CNN, extracting the embedded features of all testing samples, and classifying the testing data set via an NN classifier or an SVM classifier. Training a deep learning model is time-consuming. However, we are not retraining the D-Res-3-D CNN for each testing data set. A great advantage is that the D-Res-3-D CNN must only be trained once, and then it can be used for all testing data sets. In this paper, 106.2 min were required to train a D-Res-3-D CNN. Once the network is trained, the classifier simply calculates the Euclidean distance of the embedded features and typically executes quickly. In practice, a deployed classification system would be pretrained, so the model would only need to extract features and make a prediction. As listed in Table XVI, we show the execution times of SS-LPSVM, C-CNN, DFSL+NN, and DFSL+SVM on the IP data set because they have presented the closest performance. The execution times of SS-LPSVM comprise of five parts and the total time is shown. The execution times of C-CNN comprise of training and testing. The execution times of DFSL+NN and DFSL+SVM comprise of features extraction and testing. We can see from Table XVI that DFSL+NN and DFSL+SVM need less time compared to SS-LPSVM and C-CNN. Note that the time of features extraction for DFSL+NN and DFSL+SVM is independent of the number of labeled samples. SVM is required to search the optimal parameters. Therefore, the execution of DFSL+SVM needs more time than DFSL+NN.

TABLE XVI

EXECUTION TIMES OF TRAINING AND TESTING PROCEDURES IN THE IP DATA SET

L=5		
SS-LPSVM	DFSL+NN	DFSL+SVM
193.80(s)	11.14(s)+0.36(s)	11.14(s)+2.21(s)
L=200		
C-CNN	DFSL+NN	DFSL+SVM
27(min)+0.74(s)	11.14(s)+2.00(s)	11.14(s)+168.13(s)

F. Discussion

It is usually difficult to train a supervised deep learning model (e.g., D-Res-3-D CNN) with very few labeled samples. However, DFSL+NN and DFSL+SVM outperform

TABLE XII
CLASS-SPECIFIC ACCURACY, OA, AND AA OF DIFFERENT TECHNIQUES FOR THE UP DATA SET

Class No.	Training	Testing	CNN [37]	R-PCA-CNN [34]	CNN-PPF [45]	C-CNN [47]	D-Res-3D-CNN	DFSL+NN	DFSL+SVM
1	200	6431	85.62	92.43	97.42	97.40	95.23	95.49	97.18
2	200	18449	88.95	94.84	95.76	99.40	96.88	98.92	99.40
3	200	1899	80.15	90.89	94.05	94.84	94.85	97.47	97.90
4	200	2864	96.93	93.99	97.52	99.16	98.43	98.30	98.40
5	200	1145	99.30	100.0	100.0	100.0	99.85	100.0	100.0
6	200	4829	84.30	92.86	99.13	98.70	93.28	99.42	99.56
7	200	1130	92.39	93.89	96.19	100.0	97.67	99.40	99.25
8	200	3482	80.73	91.18	93.62	94.57	95.71	92.53	95.52
9	200	747	99.20	99.33	99.60	99.87	100.0	99.68	99.68
OA (%)			87.90	94.38	96.48	98.41	96.30	97.85	**98.62**
AA (%)			89.73	93.87	97.03	98.22	96.88	97.91	**98.54**
κ (%)			83.81	92.32	95.09	97.89	95.11	97.15	**98.17**

TABLE XIII
CLASS-SPECIFIC ACCURACY, OA, AND AA OF DIFFERENT TECHNIQUES FOR THE PC DATA SET

Class No.	Training	Testing	CNN [37]	R-PCA-CNN [34]	CNN-PPF [45]	C-CNN [47]	D-Res-3D-CNN	DFSL+NN	DFSL+SVM
1	200	65771	99.94	99.91	98.94	100.0	99.38	100.0	100.0
2	200	7398	93.34	94.58	98.04	99.18	96.79	96.76	96.91
3	200	2890	93.01	95.36	97.44	99.45	94.95	98.90	98.67
4	200	2485	90.34	96.02	99.11	99.64	99.33	99.81	99.40
5	200	6384	93.00	94.94	98.75	99.81	96.22	99.13	99.64
6	200	9048	95.09	97.13	98.82	99.27	98.03	97.40	99.29
7	200	7087	92.30	93.64	93.69	99.28	92.95	98.83	99.31
8	200	42626	99.33	99.03	99.72	99.92	99.40	99.52	99.67
9	200	2663	99.55	99.47	100.0	100.0	99.97	99.72	99.86
OA (%)			98.15	98.53	98.85	**99.83**	98.63	99.40	99.61
AA (%)			95.10	96.68	98.28	**99.62**	97.44	98.90	99.19
κ (%)			97.65	97.72	98.83	**99.49**	98.06	99.16	99.45

TABLE XIV
CLASS-SPECIFIC ACCURACY, OA, AND AA OF DIFFERENT TECHNIQUES FOR THE IP DATA SET

Class No.	Training	Testing	CNN [37]	R-PCA-CNN [34]	CNN-PPF [45]	C-CNN [47]	D-Res-3D-CNN	DFSL+NN	DFSL+SVM
1	200	1288	79.63	82.39	92.99	96.28	86.55	95.59	98.32
2	200	630	69.48	85.41	96.66	92.26	94.58	99.16	99.76
3	200	283	85.80	95.24	98.58	99.30	98.96	100.0	100.0
4	200	530	94.95	99.25	100.0	99.25	99.86	99.86	100.0
5	200	278	98.57	100.0	100.0	100.0	100.0	100.0	100.0
6	200	772	71.41	82.76	96.24	92.84	94.75	95.88	97.84
7	200	2255	90.44	96.20	87.80	98.21	86.97	94.91	95.93
8	200	393	70.26	82.14	98.98	92.45	96.46	99.16	99.66
9	200	1065	99.90	99.81	99.81	98.98	99.84	99.92	99.76
OA (%)			84.44	91.09	94.34	96.76	93.10	97.38	**98.35**
AA (%)			84.50	91.47	96.78	96.62	95.33	98.28	**99.03**
κ (%)			82.51	89.13	94.73	95.94	91.94	96.93	**98.07**

the conventional semisupervised classifiers when very few labeled samples are available. The main reason for such a performance improvement is that the proposed DFSL method can train the designed D-Res-3-D CNN to learn a metric space. In the learned metric space, samples from the same class are close and those from different classes are far. We also use the proposed method to train other networks (e.g., 2-D CNN, D-2-D CNN, D-Res-2-D CNN, 3-D CNN, D-3-D CNN). The corresponding classification accuracy is lower than that of using D-Res-3-D CNN, which demonstrates the effectiveness of D-Res-3-D CNN.

To further validate the effectiveness of the proposed methods, D-Res-3-D CNN, DFSL+NN, and DFSL+SVM are compared with several CNN-based methods when a reasonable number of labeled samples are available. Per the experimental results, D-Res-3-D CNN outperforms CNN [37] and R-PCA-CNN [34], which also demonstrates that D-Res-3-D CNN provides a way to effectively utilize the spatial spectral features. Moreover, DFSL+NN and DFSL+SVM can also achieve promising classification results, which further demonstrates the effectiveness of the proposed DFSL method.

In general, we find that DFSL+NN and DFSL+SVM can not only achieve promising classification results in small samples but also achieve competitive results compared with the state-of-the-art CNN-based methods when a reasonable number of labeled samples are available. Note that an

TABLE XV
CLASS-SPECIFIC ACCURACY, OA, AND AA OF DIFFERENT TECHNIQUES FOR THE SALINAS DATA SET

Class No.	Training	Testing	CNN [37]	R-PCA-CNN [34]	CNN-PPF [45]	C-CNN [47]	D-Res-3D-CNN	DFSL+NN	DFSL+SVM
1	200	1809	98.73	98.84	100.0	100.0	99.70	100.0	100.0
2	200	3526	99.12	99.61	99.88	99.89	100.0	100.0	99.97
3	200	1776	96.08	99.75	99.60	99.89	98.73	100.0	100.0
4	200	1194	99.71	98.79	99.49	99.25	99.78	100.0	99.86
5	200	2478	97.04	99.84	98.34	99.39	99.22	99.89	100.0
6	200	3759	99.59	99.70	99.97	100.0	100.0	100.0	100.0
7	200	3379	99.33	79.05	100.0	99.82	99.89	99.97	100.0
8	200	11071	78.64	99.17	88.68	91.45	89.51	91.70	91.67
9	200	6003	98.04	96.88	98.33	99.95	99.81	99.98	99.69
10	200	3078	92.38	99.31	98.60	98.51	97.04	99.21	99.79
11	200	868	99.14	100.0	99.54	99.31	98.03	100.0	100.0
12	200	1727	99.88	100.0	100.0	100.0	100.0	100.0	100.0
13	200	716	97.84	98.97	99.44	99.72	99.78	100.0	100.0
14	200	870	96.17	82.24	98.96	100.0	99.91	100.0	100.0
15	200	7068	72.69	97.57	83.53	96.24	85.13	96.81	97.01
16	200	1607	98.59	99.61	99.31	99.63	99.94	99.83	99.94
OA (%)			90.25	92.39	94.80	97.42	95.46	97.78	**97.81**
AA (%)			95.19	96.83	97.73	98.94	97.90	99.21	**99.25**
κ (%)			89.51	91.84	93.97	97.11	95.46	97.53	**97.56**

NN classifier is essential when there are very few samples (e.g., 5 or 10 labeled samples per class), because an NN classifier can avoid the search of additional parameters when only a few labeled samples are available. Thus, it can reduce the risk of overfitting. In contrast, we recommend using an SVM classifier for the proposed method when the labeled samples are relatively sufficient (e.g., 200 labeled samples per class). In addition, DFSL+NN and DFSL+SVM are faster than the algorithms with similar performance. The main reason is that the D-Res-3-D CNN must only be trained once. In practice, a deployed classification system would be pretrained, so the model would only need to extract features and make a prediction. Finally, we find that the proposed method is sensitive to some parameters. However, when reasonable parameters are set, the performance is considerably better than that of the conventional semisupervised methods and CNN-based methods.

IV. CONCLUSION

Although deep learning methods have demonstrated excellent performance in HSI classification, sparse-labeled samples remain a major obstacle to their application. In this paper, a DFSL method is proposed which provide a novel means to address small-scale HSI classification. The key insight is that the DFSL method trains a network (a deep residual 3-D CNN in this paper) to learn a metric space, where samples from the same class are close and those from different classes are separated by distance. In such space, classification can be conducted by a simple classifier (e.g., NN). Experiments are conducted on four widely used HSI data sets, and the results demonstrate the generalization ability of the trained model. The proposed method outperforms conventional semisupervised methods when few labeled samples are available.

Despite obtaining promising results, more studies are required to further improve the performance of the DFSL method. In the future work, we will attempt to collect large-scale hyperspectral data sets to improve the generalization ability of the network.

ACKNOWLEDGMENT

The codes of graph representation-based band selection were provided by the https://github.com/whuhenry/BandSelect.

REFERENCES

[1] P. Ghamisi *et al.*, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 37–78, Dec. 2017.

[2] 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.

[3] P. O. Gislason, J. A. Benediktsson, and J. R. Sveinsson, "Random forests for land cover classification," *Pattern Recognit. Lett.*, vol. 27, no. 4, pp. 294–300, 2006.

[4] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, "Neural network approaches versus statistical methods in classification of multisource remote sensing data," in *Proc. 12th Can. Symp. Remote Sens. Geosci. Remote Sens. Symp.*, vol. 2, Jul. 1989, pp. 489–492.

[5] M. Pal, "Extreme-learning-machine-based land cover classification," *Int. J. Remote Sens.*, vol. 30, no. 14, pp. 3835–3841, 2009.

[6] 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.

[7] C. Rodarmel and J. Shan, "Principal component analysis for hyperspectral image classification," *Surv. Land Inf. Sci.*, vol. 62, no. 2, p. 115, 2002.

[8] A. Villa, J. A. Benediktsson, J. Chanussot, and C. Jutten, "Hyperspectral image classification with independent component discriminant analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 12, pp. 4865–4876, Dec. 2011.

[9] 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.

[10] 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.

[11] 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.

[12] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson, "SVM- and MRF-based method for accurate classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 4, pp. 736–740, Oct. 2010.

[13] W. Li, C. Chen, H. Su, and Q. Du, "Local binary patterns and extreme learning machine for hyperspectral imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 7, pp. 3681–3693, Jul. 2015.

[14] S. Jia, J. Hu, Y. Xie, L. Shen, X. Jia, and Q. Li, "Gabor cube selection based multitask joint sparse representation for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3174–3187, Jun. 2016.

[15] Q. Wang, Z. Meng, and X. Li, "Locality adaptive discriminant analysis for spectral–spatial classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 11, pp. 2077–2081, Nov. 2017.

[16] M. Chen, Q. Wang, and X. Li, "Discriminant analysis with graph learning for hyperspectral image classification," *Remote Sens.*, vol. 10, no. 6, p. 836, 2018.

[17] Y. Tarabalka, J. A. Benediktsson, and J. Chanussot, "Spectral–spatial classification of hyperspectral imagery based on partitional clustering techniques," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 8, pp. 2973–2987, Aug. 2009.

[18] K. Sun *et al.*, "A robust and efficient band selection method using graph representation for hyperspectral imagery," *Int. J. Remote Sens.*, vol. 37, no. 20, pp. 4874–4889, 2016.

[19] Q. Wang, J. Lin, and Y. Yuan, "Salient band selection for hyperspectral image classification via manifold ranking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1279–1289, Jun. 2016.

[20] B. Krishnapuram, L. Carin, M. A. T. Figueiredo, and A. J. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 957–968, Jun. 2005.

[21] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th Int. Conf. Mach. Learn.*, vol. 99, 1999, pp. 200–209.

[22] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proc. IEEE Workshops Appl. Comput. Vis.*, Jan. 2005, pp. 29–36.

[23] R. K. Ando and T. Zhang, "Two-view feature generation model for semi-supervised learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 25–32.

[24] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 19–26.

[25] L. Wang, S. Hao, Q. Wang, and Y. Wang, "Semi-supervised classification for hyperspectral imagery based on spatial-spectral label propagation," *ISPRS J. Photogram. Remote Sens.*, vol. 97, pp. 123–137, Nov. 2014.

[26] K. Tan, E. Li, Q. Du, and P. Du, "An efficient semi-supervised classification approach for hyperspectral imagery," *ISPRS J. Photogramm. Remote Sens.*, vol. 97, pp. 36–45, Nov. 2014.

[27] K. Tan, J. Hu, J. Li, and P. Du, "A novel semi-supervised hyperspectral image classification approach based on spatial neighborhood information and classifier combination," *ISPRS J. Photogramm. Remote Sens.*, vol. 105, pp. 19–29, Jul. 2015.

[28] J. Cao and B. Wang, "Embedding learning on spectral–spatial graph for semisupervised hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 10, pp. 1805–1809, Oct. 2017.

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

[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

[31] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[32] X. X. Zhu *et al.*, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017.

[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] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2015, pp. 4959–4962.

[35] 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.

[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] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. Sensors*, vol. 2015, 2015, Art. no. 258619, doi: 10.1155/2015/258619.

[38] 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, 2015.

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

[40] 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.

[41] B. Liu, X. Yu, A. Yu, and G. Wan, "Deep convolutional recurrent neural network with transfer learning for hyperspectral image classification," *J. Appl. Remote Sens.*, vol. 12, no. 2, p. 12–17, 2018.

[42] 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.

[43] 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.

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

[45] 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.

[46] B. Liu, X. Yu, P. Zhang, X. Tan, A. Yu, and Z. Xue, "A semi-supervised convolutional neural network for hyperspectral image classification," *Remote Sens. Lett.*, vol. 8, no. 9, pp. 839–848, Sep. 2017.

[47] S. Mei, J. Ji, J. Hou, X. Li, and Q. Du, "Learning sensor-specific spatial-spectral features of hyperspectral images via convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4520–4533, Aug. 2017.

[48] B. Liu, X. Yu, P. Zhang, A. Yu, Q. Fu, and X. Wei, "Supervised deep feature extraction for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 1909–1921, Apr. 2018.

[49] R. Kemker and C. Kanan, "Self-taught feature learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 5, pp. 2693–2705, May 2017.

[50] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018.

[51] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst.* Cambridge, MA, USA: MIT Press, 2016, pp. 3630–3638.

[52] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. Int. Conf. Learn. Represent.*, Apr. 2017. [Online]. Available: ttps://openreview.net/pdf?id=rJY0-Kcll

[53] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.* Cambridge, MA, USA: MIT Press, 2017, pp. 4080–4090.

[54] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[55] B. Liu, X. Yu, P. Zhang, X. Tan, R. Wang, and L. Zhi, "Spectral–spatial classification of hyperspectral image using three-dimensional convolution network," *J. Appl. Remote Sens.*, vol. 12, no. 1, p. 016005, 2018.

[56] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[57] 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.

[58] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[59] N. Yokoya and A. Iwasaki, "Airborne hyperspectral data over Chikusei," Space Appl. Lab., Univ. Tokyo, Tokyo, Japan, Tech. Rep. SAL-2016-05-27, May 2016.
[60] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Nov. 2006.
[61] B.-C. Kuo, C.-S. Huang, C.-C. Hung, Y.-L. Liu, and I.-L. Chen, "Spatial information based support vector machine for hyperspectral image classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2010, pp. 832–835.
[62] I. Dópido, J. Li, P. R. Marpu, A. Plaza, J. M. B. Dias, and J. A. Benediktsson, "Semisupervised self-learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 7, pp. 4032–4044, Jul. 2013.

Bing Liu received the B.S. and M.S. degrees from the Institute of Surveying and Mapping, Information Engineering University, Zhengzhou, China, in 2013, and 2016, respectively, where he is currently pursuing the Ph.D. degree.

His research interests include machine learning, pattern recognition, and hyperspectral image classification.

Xuchu Yu received the Ph.D. degree from the Institute of Surveying and Mapping, Information Engineering University, Zhengzhou, China, in 1997.

He is currently a Professor and a Doctoral Supervisor with the Information Engineering University. His research interests include photogrammetry, remote sensing, and pattern recognition.

Anzhu Yu received the Ph.D. degree from the Institute of Surveying and Mapping, Information Engineering University, Zhengzhou, China, in 2018.

He is currently a Lecturer with the Institute of Surveying and Mapping, Information Engineering University. His research interests include signal processing in earth observation.

Pengqiang Zhang received the Ph.D. degree in photogrammetry and remote sensing from the Institute of Surveying and Mapping, Information Engineering University, Zhengzhou, China, in 2009.

He is currently an Associate Professor with the Information Engineering University. His research interests include remote sensing image processing and pattern recognition.

Gang Wan received the Ph.D. degree from the Institute of Surveying and Mapping, Information Engineering University, Zhengzhou, China, in 2006.

He is currently a Professor and Doctoral Supervisor with the Institute of Surveying and Mapping, Information Engineering University. His research interests include geospatial information grid, virtual geographic environment, and big data visual analysis.

Ruirui Wang received the B.S. degree in surveying and mapping engineering from Xuchang University, Xuchang, China, in 2014. She is currently pursuing the master's degree in photogrammetry and remote sensing with the Institute of Surveying and Mapping, Information Engineering University, Zhengzhou, China.

Her research interests include machine learning and feature extraction.