Caps-TripleGAN: GAN-Assisted CapsNet for Hyperspectral Image Classification

Xue Wang, Kun Tan, *Senior Member, IEEE*, Qian Du, *Fellow, IEEE*, Yu Chen, and Peijun Du, *Senior Member, IEEE*

***Abstract*— The increase in the spectral and spatial information of hyperspectral imagery poses challenges in classification due to the fact that spectral bands are highly correlated, training samples may be limited, and high resolution may increase intra-class difference and interclass similarity. In this paper, in order to better handle these problems, a Caps-TripleGAN framework is proposed by exploring the 1-D structure triple generative adversarial network (TripleGAN) for sample generation and integrating CapsNet for hyperspectral image classification. Moreover, spatial information is utilized to verify the learning capacity and discriminative ability of the Caps-TripleGAN framework. The experimental results obtained with three real hyperspectral data sets confirm that the proposed method outperforms most of the state-of-the-art methods.**

***Index Terms*— CapsNet, hyperspectral image classification, triple generative adversarial network (TripleGAN).**

I. Introduction

DUE to advances in hyperspectral remote sensing technology, the use of hyperspectral imagery has resulted in great breakthroughs in the field of earth observations. A hyperspectral image acquired by an imaging spectrometer is a 3-D data cube, where each pixel represents contiguous spectral information. However, the increase in spectral and spatial information poses some challenges in classification.

1) Hyperspectral images are comprised of hundreds of bands with high spectral resolutions, and the resulting spectral vector is highly correlated.
2) Because of the wide spatial coverage and the costly field surveying [1] and labeling, classification is confronted with imbalance between the small number of labeled samples and the high feature dimensions, leading to the Hughes phenomenon which describes that the performance of a classifier is degraded by high-dimensional features under limited samples; although class separability may be enhanced by increasing data dimensionality, accurate estimation of class-conditional probability density function needs more samples due to the additional dimension.
3) The high resolution may increase intraclass variation and decrease interclass difference, which gives rise to poor class separability.

For the first challenge, efficient approaches should be explored to decrease the spectral redundancy, and the specific features suited to a given classification task should be used, instead of using all the features. To address the second problem, unlabeled samples may be considered to improve the performance of classification because labeled and unlabeled samples belong to the same source domain, ignoring the transfer between algorithms and data. In transfer learning, the training data set and test data set usually belong to different data sources, which is out of the scope of our discussion. With regard to the phenomenon of intraclass difference, spatial information should be efficiently utilized to produce higher accuracies. Above all, removing the redundancy effectively, combining the small number of samples for classification, and mining the spatial information to improve the accuracy have become hot topics in the classification of hyperspectral imagery.

In the early development of hyperspectral image classification, the algorithms were mainly based on the pattern recognition approaches. To address the issue of the Hughes phenomenon, hyperspectral classification has primarily focused on dimensionality reduction, using either feature selection or feature extraction as a way to feed the above classifiers [2]–[4]. In addition, hand-crafted feature extraction methods, such as Haralick feature extraction [5], scale-invariant feature transform [6], and local binary patterns [7], have been introduced into remote sensing classification. Most recently, deep learning has led to significant advances in various fields, such as imaging technology [8] and natural language processing [9], and has brought new opportunities for hyperspectral techniques. The deep learning approaches have demonstrated great potential in the field of remote sensing for extracting features which are invariant to most local changes. Some researchers have adopted deep learning networks to simultaneously optimize the feature extraction

Manuscript received March 11, 2019; revised April 13, 2019; accepted April 16, 2019. Date of publication May 9, 2019; date of current version August 27, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 41871337 and Grant 41471356, in part by the Xuzhou Scientific Funds under Grant KC16SS092, and in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions. (Xue Wang and Kun Tan contributed equally to this work.) (Corresponding author: Kun Tan.)

X. Wang and Y. Chen are with the Key Laboratory for Land Environment and Disaster Monitoring of NASG, China University of Mining and Technology, Xuzhou 221116, China.

K. Tan is with the Key Laboratory for Land Environment and Disaster Monitoring of NASG, China University of Mining and Technology, Xuzhou 221116, China, and also with the Key Laboratory of Geographic Information Science, Ministry of Education, East China Normal University, Shanghai 200241, China (e-mail: tankuncu@gmail.com).

Q. Du is with the Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS 39762 USA.

P. Du is with the Key Laboratory for Satellite Mapping Technology and Applications of NASG, Nanjing University, Nanjing 210023, China (e-mail: dupjrs@gmail.com).

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

Digital Object Identifier 10.1109/TGRS.2019.2912468

and classification stages for hyperspectral imagery, which resulted in better performance than traditional algorithms in machine learning [10], [11]. The deep networks can be divided into discriminative models, such as convolutional neural networks (CNNs) [8], recurrent neural networks (RNNs) [9], and the stacked denoising autoencoder (SDA) [12], and generative models, such as deep belief networks (DBNs) [13], deep restricted Boltzmann machines (DRBMs) [14], and generative adversarial networks (GANs) [15]. He *et al.* [16] used a 3-D CNN for hyperspectral image classification at multiple scales; Song *et al.* [17] proposed a deep feature fusion network to optimize a general CNN in hyperspectral classification; Zhang and Li [18] adopted 1-D kernels to fit the hyperspectral context; and Mou *et al.* [19] used an RNN model to analyze hyperspectral pixels as sequential data for the classification task. While CNN is one of the most popular and effective deep learning models, the output of CNN just represents the specific feature variants, ignoring detecting the variations of features. Moreover, the CNN adopts max-pooling to improve the discriminative ability, but loses many local features. When the spatial–spectral convolution is operated, convolution in the spatial dimension results in the local pixels introducing classification error and the massive similar results are produced.

With this consideration, we have explored CapsNet [20] and modified the original architecture to fit the hyperspectral image classification task. The most important difference with the conventional CNN is that CapsNet has capsules, which consist of a group of neurons. Moreover, the pooling operation is disabled. Recently, CapsNet has been used in hyperspectral image classification [21], [22].

To confront the insufficient training sample issue, semi-supervised learning (SSL) can be utilized in hyperspectral image classification. The simplest algorithm is based on a self-training scheme [23], where the classifier is bootstrapped with additional labeled data obtained from its own highly confident predictions. Transductive SVM (TSVM) extends SVM with the aim of max-margin classification, while ensuring that there are as few unlabeled observations near the margins as possible [24]. Moreover, other SSL methods, such as graph-based methods [25], cotraining [26], and manifold methods [27], have been introduced in hyperspectral image classification. Recently, a generative model which recognizes SSL problem as a specialized missing data imputation task [28] has been exploited for classification. The existing methods, such as Gaussian mixture models or hidden Markov models, do not perform well in remote sensing classification because of the need for a large number of mixed components. More recently, variational approximations have also been exploited for pan-sharpening [29] and classification [30]. GANs, which were first proposed in 2014 [15], have made significant breakthroughs in the field of image generation. GANs have provided a new pathway for sample generation by the use of an adversarial process to address the issue of insufficient training samples in remote sensing classification. However, in general, with GANs, the discriminator only takes the data into account, ignoring the label information when discriminating if the data are from the real data or model distribution, and the latent variable cannot learn the label knowledge in the generator during the adversarial training process. Some GAN-based models considering label information have been modified, such as conditional GAN [31], InfoGAN [32], auxiliary classifier GAN (ACGAN) [33], deep convolutional GAN (DCGAN) [34], and categorical GAN (CatGAN) [35]. In the remote sensing area, GANs have been exploited in various applications. Conditional GAN models have been explored in thin cloud removal in multispectral imagery [36], remote sensing image synthesis [37], heterogeneous image matching between synthetic aperture radar (SAR) and optical images [38], height simulation using a digital surface model (DSM) and optical images [39], and lung histology with hyperspectral images [40]; DCGANs have been used with high-resolution remote sensing imagery to boost the scene classification results [41] and in unsupervised representation by multiple-layer feature matching [42]. GANs have also been used in hyperspectral image classification [43]–[46], change detection [47], and retrieval tasks [48]. Although some approaches based on GANs have become feasible for semi-supervised classification, two-player GAN games are weak at optimizing the discriminative model and categorical model at the same time [49]. To solve this problem, a triple generative adversarial network (TripleGAN) is exploited in this paper and a composite pattern with CapsNet is designed.

In this paper, the Caps-TripleGAN framework is designed by exploring the 1-D structure TripleGAN for sample generation and integrating CapsNet for hyperspectral image classification. Moreover, spatial information is utilized to enhance the discriminative ability of the Caps-TripleGAN framework.

The rest of this paper is organized as follows. Section II gives the background to this paper. Section III details the Caps-TripleGAN framework. Section IV describes the three real hyperspectral images used in the experiments, the experimental results, and the comparisons with other methods. Finally, the conclusions of this paper are drawn in Section V.

II. Previous Work

In this paper, the main work is based on CapsNet and GANs. Therefore, in this section, we briefly review the CNNs, CapsNet, and GANs.

A. CNNs

CNNs have recently made great breakthroughs in remote sensing analysis [50]. As shown in Fig. 1, a typical CNN is a multilayer neural network which is composed of a convolution layer, a pooling layer, and a fully connected layer. In the convolution layer, each filter f in Fig. 1 is represented by a fixed-size matrix. The convolution operation involves multiplying the corresponding position and summing, which can be regarded as obtaining the correlation between the data and the filter. Each filter maintains different interests in various features, such as the color intensity, the border, and the specific contour. The feature map is generated by transforming the convolution output with an activation function. Each layer has various filters to detect the corresponding feature and, as a result, the features often contain redundancies. The pooling layer is appended to downsample the features into a small size, which can be regarded as reducing the feature redundancy.

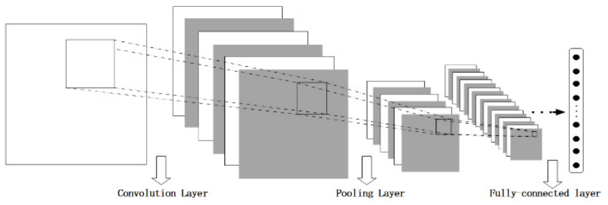

Fig. 1. Typical architecture of a CNN.

A CNN can learn various features, but the relationships between these features are rarely explored. For example, if the CNN detects the features of nose, eyes, and mouth, it will then label this image as a human face. However, if the position, relationship, or size of these features is wrong, the CNN will still give the "human face" label. In this case, CapsNet could perform better than the standard CNN.

B. CapsNet

The key concept of CapsNet is that the output of neurons is not a scalar on behalf of the activation, but a vector to recognize the presence of feature entities and to encode the entities' properties into the vector outputs. That is, CapsNet is dedicated to detecting the features and also the variations of features, rather than just the specific feature variants as in the conventional CNN. Moreover, CapsNet can recognize the objects which can transform into each other; for instance, a capsule can detect if an object is rotated in the characteristic space, instead of realizing a rotated object. On this account, CapsNet is forced to learn the features and their variance so that more variants can be inferred effectively with fewer training data.

In conventional CNN hierarchy, the pooling layer is used to reduce the computational burden and handle the feature invariance. Intuitively, the category label can be kept the same when the features have only slightly changed. This operation maintains important features and discards others to ensure robust results in regular image classification; however, in advanced tasks, such as target detection or hyperspectral image classification, which should consider more detailed spectral information to handle the problem of the similar spectra from different materials and the same material with different spectra, the intrinsic activation value should be taken into account to keep the task manageable.

C. GANs

A GAN is essentially a kind of training model, which is based on a deep generative model, and is not a specific network [15]. Its structure is composed of more than one network, and each network plays a different role within its architecture. A typical GAN consists of two networks named a generator and a discriminator. The generator learns the real data distribution and makes the generated data more realistic. The discriminator learns whether a sample is from the model distribution or the real data distribution.

During the training processing step, the generator can be regarded as a "maturing counterfeiter," trying to produce the data and fool the discriminator, while the discriminator can be seen as analogous to an inspector, intending to detect the fake data. The generative models can be estimated via an adversarial game process. As the processing takes place, both the generator and the discriminator converge to a dynamic equilibrium, in which the generated data are close to the real data distribution, so that the discriminator cannot discern which is fake or real and gives the probability of a real prediction as being close to 0.5 (equivalent to a random guess). The discriminator is essentially a binary classifier, which can be undertaken by a conventional CNN, and the generator uses a deconvolutional neural network to transform the latent variable into fake data which is consistent with the model distribution. In the typical adversarial model, these two roles are both deep neural networks, and a uniform distribution variable $p_z(z)$ is used to learn the generator's target distribution p_g and produce a sample $G(z)$ in the data space, which is represented by a deep neural network with parameters θ_g, so that the generator is $G(z;\theta_g)$. Similarly, the discriminator is $D(x;\theta_d)$, where $p(x)$ is the real data distribution and $D(x)$ represents the probability of x coming from the real data rather than the learned distribution p_g. The training procedure is now to solve minimax problem by two-player game

$$\min_G \max_D V(G,D) = E_{x\sim p(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (1)$$

where $V(\cdot)$ denotes the observed value. The training goals are maximizing the probability of assigning the label to the real data and fake data from the generator and minimizing $\log(1 - D(G(z)))$. The optimal model $D(x)$ is $p(x)/(p(x)+p_g(x))$, and the global equilibrium of this two-player game is obtained when $p(x) = p_g(x)$ [15].

III. Proposed Method

A. CapsNet for Hyperspectral Image Classification

One of the aims in this paper is to overcome the deficiencies of the existing CNNs in hyperspectral image classification. In this regard, we explore CapsNet and modify the original architecture to fit this task.

Suppose that a hyperspectral data set with b spectral bands contains N labeled samples for L classes, and each is represented by $\{x_1, x_2, \ldots, x_N\} \in R^{1\times b}$, and the corresponding label vector is $Y = \{y_1, y_2, \ldots, y_L\} \in R^{1\times L}$. As shown in Fig. 2, the spectral features are used as the input of CapsNet, which is first convoluted by the 1-D filter. Let the number of channels be denoted as CH_1, the size of the kernel be $1 \times K$, and the stride be equal to one. The first rectified linear unit (ReLU) Conv1 layer can be obtained by conventional channel convolution. A data block of size $((N-K+1)) \times \mathrm{CH}_1$ is generated. The primary capsules are established in the next layer, where the number of channels is CH_2, the size of the kernel is $1 \times K$, and the stride is equal to two. The number of capsules in this layer equals $\mathrm{CH}_2 \times N_2$ (where each output is a vector) and each capsule in $N_2 \times 1$ shares their weights with the others, where d_1 is the vector dimension and N_2 equals $(N/2 - K + 1)$. Unlike the feed-forward and backward propagation used in a CNN, dynamic routing is utilized to concatenate different capsule layers, which groups the capsules

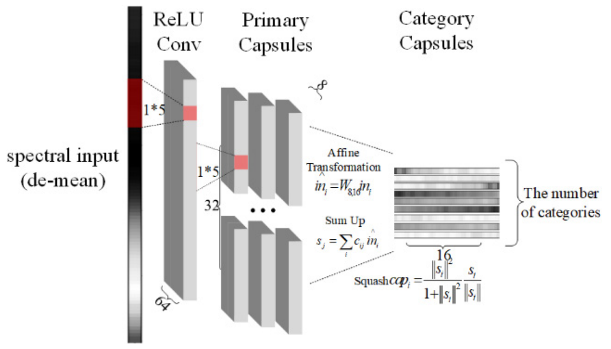

Fig. 2. Structure of the 1-D CapsNet.

to form a parent capsule and also calculates the capsules' output.

Intuitively, for one of the capsules, the input in_i and the output cap_i are vectors. First, the transformation matrix $W_{i,j}$ is applied to ensure that the previous capsule vector in_i is aligned with the upper-layer capsule cap_i. This process can be seen as an affine transformation to make the two vectors have the same dimension, which can be expressed as

$$\widehat{\text{in}_i} = W_{i,j}\,\text{in}_i \tag{2}$$

where $\widehat{\text{in}_i}$ has the same dimension as cap_i. The aligned vector $\widehat{\text{in}_i}$ is treated as the vote from capsule i on the output of capsule j. The total input s_i of capsule i is obtained by summing each $\widehat{\text{in}_i}$ as

$$s_i = \sum_i c_{i,j}\widehat{\text{in}_i} \tag{3}$$

where $c_{i,j}$ denotes the weight between the capsules of two layers, which is determined by the dynamic routing process, $i \in \{1, 2, \ldots, (\text{CH}_2 \times N_2)\}$ and $j \in \{1, 2, \ldots, L\}$.

The output of a capsule is a vector, whose length represents the probability of the current input. A nonlinear "squashing" function is used to scale the vector between zero and one, i.e., the small vectors are scaled to almost zero and the large vectors to a length close to one, which can be expressed as

$$\begin{aligned}
\text{cap}_i &= \frac{\|s_i\|^2}{1+\|s_i\|^2}\frac{s_i}{\|s_i\|}\\
\text{cap}_i &\approx \|s_i\| s_i \text{ when } s_i \text{ is small}\\
\text{cap}_i &\approx \frac{\|s_i\|}{s_i} \text{ when } s_i \text{ is large}
\end{aligned} \tag{4}$$

where s_i denotes the input of capsule i achieved by the weighted sum of all the capsules' output in the previous layer, and cap_i represents the output of capsule i after the squashing activation function. The first part of (4) is to shrink the length to between zero and one, and the second part is to keep the direction of the capsule. Here, cap_i and s_i have the same direction. In (3), $c_{i,j}$ among the different capsules in the previous layer are different; namely, the contributions of the previous capsules to the up-layer capsules are different and are determined by a "routing softmax," that is

$$c_{i,j} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}. \tag{5}$$

Here, the normalization of the coupling coefficients is computed as the softmax of b_{ij}, where b_{ij} is used to measure the agreement between the weighted input vector $\widehat{\text{in}_i}$ and the output vector cap_i. If cap_i is close in value to $\widehat{\text{in}_i}$, it can be concluded that both capsules are highly related. This similarity is measured using the scalar product of $\widehat{\text{in}_i}$ and cap_i as

$$b_{ij} = \widehat{\text{in}_i}\cdot\text{cap}_i. \tag{6}$$

From this equation, the measure of the similarity b_{ij} considers both the probability and the features' properties, instead of just the probability in the neurons, as in the conventional CNN. b_{ij} is treated as if it was a log likelihood. To make b_{ij} more accurate, it is updated iteratively as

$$b_{ij} \leftarrow b_{ij} + \widehat{\text{in}_i}\cdot\text{cap}_i. \tag{7}$$

As the iteration progresses, the similarity is evaluated between a specific capsule and all the capsules in the next layer. If the score is low, the weight c is small, and vice versa. When the iteration is completed, the capsule only links to the next-layer capsules which it identified.

For the hyperspectral image classification task, the norm of the vector is used to indicate the probability of a local spectral feature existence. With regard to a specific land category, the capsule of the category layer should have a large output vector. Therefore, a separate margin loss L_C for each category is defined as

$$\begin{aligned}
L_C = T_C \max\left(0, m^+ - \|\vec{v_c}\|\right)^2&\\
+\lambda(1-T_C)\max\left(0, \|\vec{v_c}\| - m^-\right)^2&
\end{aligned} \tag{8}$$

where T_C is a "one-hot vector," which is a vector with a single "1" and all the others "0." If an object of class C is present, the corresponding position equals one. m^+ and m^-, which are penalty terms, are fixed to 0.9 and 0.1, respectively. The λ downweighting (default 0.5) stops the initial learning from shrinking the activity vectors of all the classes. The total loss is simply the sum of the losses of all the category capsules.

B. Caps-TripleGAN Framework Applied to a Small Data Set

As mentioned earlier, GANs have shown good performance in image generation, but for SSL, the existing two-player GANs have two problems.

1) The generator and the categorical discriminator may not be optimal at the same time; in other words, this kind of GAN works well for either categorical discrimination or generation, but cannot achieve an effective generator and classifier simultaneously.
2) The generator cannot learn the latent representation among the different categories exhaustively.

The SSL is aimed at extracting the classification information or other meaningful features from the data itself, with limited supervision, which is impossible for the two-player GANs. The discriminator only takes the data into account while ignoring the label information when discriminating if the data are from real data or a model distribution. Hence, the latent variable cannot learn the label knowledge in the generator during the adversarial training process.

In this paper, TripleGAN is explored for training data generation to improve the performance of CapsNet. Unlike the general two-player GANs, TripleGAN consists of three components, which handle the three distributions jointly.

For hyperspectral image classification, it is critical to assign a high-dimensional spectral pixel with the correct label, which can be represented by the conditional distribution $p_c(y|x)$. As mentioned above, there remains a costly field surveying and labeling task in the small number of labeled samples, which can be solved by learning the class-conditional distribution $p_g(x|y)$.

"Triple" denotes that it consists of three components, a classifier C, a class-condition generator G, and a discriminator D. D is needed to control the class-conditional distribution $p_g(x|y)$ converging to the real data distribution, as in the two-player GANs. Specifically, C is used to describe the conditional distribution $p_c(y|x)$, G to characterize the class-conditional distribution $p_g(x|y)$, and D that detects a data pair (x, y) that comes from the real distribution $p_r(x, y)$ or the model distribution $p_g(x, y)$ or $p_c(x, y)$. Each player is characterized as a neural network. A hyperspectral pixel can be modeled by both $p(x)$ and $p(y)$, and the classifier C can assign a label y given x by the posterior probability of the conditional distribution. Therefore, the joint distribution can be computed as $p_c(x, y) = p(x)p_c(y|x)$. As for $p_g(x, y)$, y is sampled from $p(y)$ and a latent distribution $p_z(z)$ is used to generate $x_g = G(y, z)$. In this paper, a Gaussian distribution is chosen as the latent distribution. After the generation of the fake data pair, the real data pair (x_r, y_r) which is positive training samples and the fake data pairs (x_g, y_g) and (x_c, y_c) which are negative training samples are sent to the supervised learning of discriminator D. The training process is based on an adversarial process, where the adversarial loss is characterized by three players

$$\begin{aligned}\min_{G,C}\max_{D} V(G, C, D) &= E_{(x,y)\sim p(x,y)}[\log D(x, y)] \\ &\quad + \alpha E_{(x,y)\sim p_c(x,y)}[\log(1 - D(x, y))] \\ &\quad + (1-\alpha)E_{(x,y)\sim p_g(x,y)}[\log(1 - D(G(y, z), y))] \\ &\quad + E_{(x,y)\sim p(x,y)}[-\log p_c(y|x)] \qquad (9)\end{aligned}$$

where $V(\cdot)$ denotes the observed value. The training objective is to maximize the probability of assigning the label to the real data and fake data from the generator and classifier while minimizing $\log(1-D(G(y, z), y))$ and $\log(1-D(x, y))$ for the generator and classifier, respectively. $\alpha \in (0, 1)$ is a constant variable that controls the contribution of the generator and classifier. In this paper, the generator and the classifier should be balanced in hyperspectral image classification, so $\alpha = 0.5$ is selected. As in (1), the three players converge if and only if $p(x, y) = \alpha p_c(x, y) + (1 - \alpha)p_g(x, y)$. That is, C and D should converge to the real data distribution at the same time, which cannot be guaranteed. Hence, an additional objective function $E_{(x,y)\sim p(x,y)}[-\log p_c(y|x)]$ should be minimized to ensure a unique optimal solution. It should be mentioned that the validity of (9) is proven in [49]. Training samples in hyperspectral image classification are often insufficient. According to the corollary proposed in [49], adding any divergence between any two of the joint distributions or the conditional distributions or the marginal distributions to the objective function as an additional regularization to be minimized will not change the global equilibrium of the objective function. The discriminative loss from the generated data pair $L_g = E_{p_g}[-\log p_c(y|x)]$ optimizes the C in supervised learning. Therefore, the last term of (9) should be replaced by

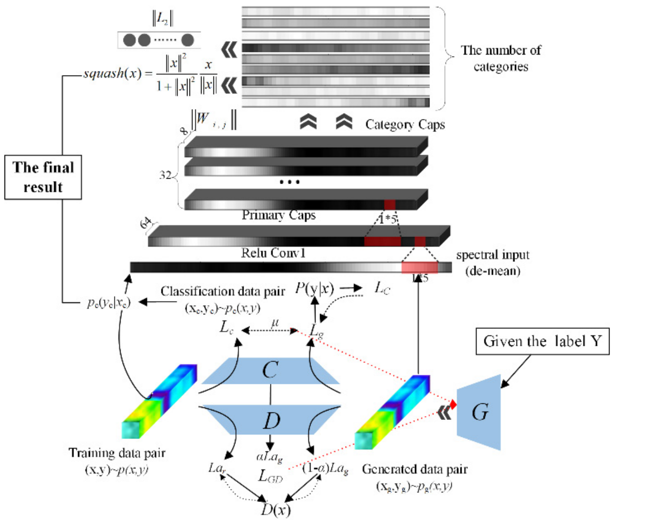

Fig. 3. Structure of the Caps-TripleGAN framework for hyperspectral image classification.

$$E_p[-\log p_c(y|x)] + \mu E_{p_g}[-\log p_c(y|x)]. \qquad (10)$$

Specifically, minimizing L_g with respect to C is equivalent to the minimization of Kullback–Leibler (KL) divergence between $p_g(x, y)$ and $p_c(x, y)$, which cannot be computed with the unknown likelihood ratio $p_g(x, y)/p_c(x, y)$. Here, μ is the weight to control the contribution of the fake data pair in the process of optimization of C.

Finally, the Caps-TripleGAN framework is designed, in which TripleGAN is utilized to generate data pairs, which are then added to CapsNet as the supplementary samples. With the real and supplementary samples, classification results of CapsNet are obtained. By combining the results of TripleGAN and CapsNet, the final classification is achieved. The algorithm flowchart is shown in Fig. 3.

Because of the small size of the training data, the existing labeled data are used to train the three players named a classifier, a discriminator, and a generator. During the adversarial training process, G is maturing to learn the real distribution, and C can achieve the precise label. When the game is converged or the training epochs achieve a certain number, the output from the generator can be regarded as a reliable data pair. CapsNet is then trained by the hybrid of the generated data pairs and the real data pairs. When the training process is finished in CapsNet, its output for an unlabeled sample is used to compare with the label from C. When the results of the two methods are consistent, the final output is fixed. Otherwise, when these two results are different, the largest values of all the categories are compared. Due to the squashing function

TABLE I

PSEUDOCODE OF THE PROPOSED ALGORITHM

```
Algorithm Caps-TripleGAN framework applied to a small dataset
Input: HSI $X \in R^{b*rows*cols}$, training data pairs $T \in \{(x_1,y_1),\ (x_2,y_2) \ldots (x_n,y_n)\}, x_n \in R^{1\times b}$, test data pairs $S \in \{(x_1,y_1),\ (x_2,y_2) \ldots (x_s,y_s)\}, x_n \in R^{1\times b}$
Initialize: Discriminator D, generator G, classifier C and CapsNet.
for number of C, D and G training iterations do:
    Obtain $(x,y)\sim p_c(x,y)$ by C on batch size $m_c$ using T and unlabeled samples.
    Sample $(x,y)\sim p_g(x,y)$ by G on batch size $m_g$ using given label values and latent variable $z$
    Sample $(x,y)\sim p(x,y)$ on batch size $m_d$ from T
    Construct $(x_d,y_d)$ with $(x,y)\sim p_c(x,y)$, $(x,y)\sim p_g(x,y)$ and $(x,y)\sim p(x,y)$ to train D. Assign $(x_c,y_c)$ and $(x_g,y_g)$ with negative label, $(x,y)$ with positive label.
    Update D with gradient:
    $\nabla_{Para_D}\left[\frac{1}{m_d}\left(\sum_{(x,y)\sim p(x,y)} logD(x,y)\right)+\frac{\alpha}{m_c}\left(\sum_{(x,y)\sim p_c(x,y)} log\big(1-D(x,y)\big)\right)+\frac{1-\alpha}{m_g}\left(\sum_{(x,y)\sim p_g(x,y)} log\big(1-D(x,y)\big)\right)\right]$
    Calculate the cost of C on $(x,y)\sim p_g(x,y)$ and $(x,y)\sim p(x,y)$ using equation (10),
    Update C with gradient:
    $\nabla_{Para_C}\left[\frac{\alpha}{m_c}\left(\sum_{(x,y)\sim p_c(x,y)} p_c(y|x)log\big(1-D(x,y)\big)\right)+ E_{p_g}[-logp_c(y|x)]+\mu E_p[-logp_c(y|x)]\right]$
    Update G with gradient:
    $\nabla_{Para_C}\left[\frac{1-\alpha}{m_g}\left(\sum_{(x,y)\sim p_g(x,y)} log\big(1-D(x,y)\big)\right)\right]$
    Generate $(x,y)\sim p_g(x,y)$ by G on batch size $m_g$ using given label values and latent variable $z$
Endfor
Concatenate the $(x,y)\sim p_g(x,y)$ and $(x,y)\sim p(x,y)$ as $T_n$
for number of CapsNet training iterations do:
    Extract the primary feature by conventional convolution operation and Construct capsule $in_i$
    Dynamic routine operation using equation (2) ~ (7)
    Calculate the cost using equation (8) and reconstruct cost
    Update all parameter of CapsNet
Endfor
Output: the final label is depended on the output whose certainty is higher between C and CapsNet.
```

being shrunk to 0–1, which is in accordance with the value of the posterior probability from C, the final result can be obtained. The pseudocode of the proposed Caps-TripleGAN classification algorithm is described in Table I.

C. Spatial Feature Utilization

To verify the proposed Caps-TripleGAN framework, the spectral–spatial information used in the previous step is turned into a vector, as shown in Fig. 4. The new feature consists of the original spectral vector and spatial feature. At first, the hyperspectral image is reduced by the PCA, and then the first three principal components are retained. For the reduced image, patches of a fixed size centered on a certain pixel are extracted as 1-D spatial vector. The spectral–spatial vector is applied to the Caps-TripleGAN framework by concatenating the original spectral vector with the 1-D spatial vector.

IV. EXPERIMENTS

In order to demonstrate the performance of the proposed technique, three real hyperspectral images were utilized. Two of these images are well-known standard hyperspectral image data sets, i.e., the Indian Pines and Pavia University data sets, which were captured by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) (website) and the Reflective Optics System Imaging Spectrometer (ROSIS) (website), respectively.

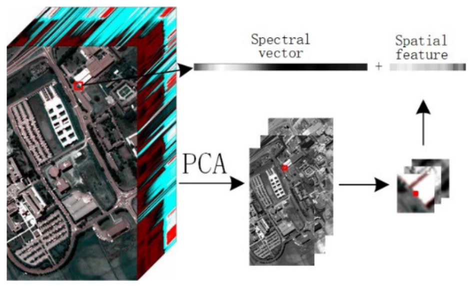

Fig. 4. Combination of spatial and spectral features.

These two data sets were evaluated by various approaches because of the high dimensionality in the spectral domain and the complex spatial structures. The other data set was produced by an airborne HYSPEX hyperspectral camera. A brief description of each data set is given in Section IV-A. The overall accuracy (OA) and kappa coefficient (kappa) are used to report the performance of all the methods on these data sets.

A. Data Set Description

1) Indian Pines AVIRIS Data Set: The Indian Pines data set was collected by the AVIRIS sensor over the northwestern Indiana agricultural test site in June 1992. This data set consists of 145 × 145 pixels with a spatial resolution

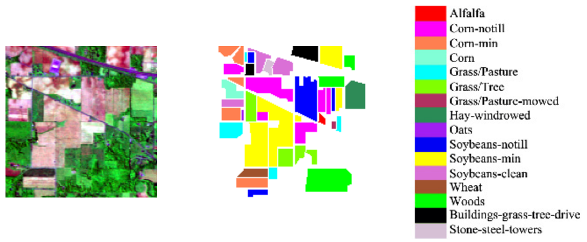

Fig. 5. Pseudocolor composite image and the corresponding ground truth for the Indian Pines AVIRIS data set.

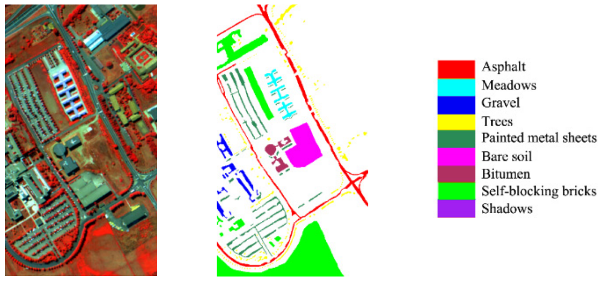

Fig. 6. Pseudocolor composite image and the corresponding ground truth for the Pavia University ROSIS data set.

of 17 m/pixel. The number of spectral bands is 224, ranging from 400 to 2500 nm, and 24 bands which were (near 1400 and 1900 nm) affected by the absorption of water vapor were removed. The remaining 200 spectral channels were used in our experiment. The available training samples cover 16 categories of interest. The pseudocolor composite image and the labeled categories are shown in Fig. 5. On account of its low spatial resolution and small structural size, the Indian Pines data set presents mixed pixels, which cause this data set to be considered a challenging data set for classification.

2) Pavia University ROSIS Data Set: The Pavia University data set was acquired by the ROSIS sensor over the Engineering School of the University of Pavia, Italy, in 2013. This data set consists of 610×340 pixels with a spatial resolution of 1.3 m/pixel. The number of spectral bands is 115, ranging from 430 to 860 nm, and 12 noisy bands were removed because of the absorption of water vapor. The remaining 103 spectral bands were used in the experiments. The data set contains nine categories of interest. The pseudocolor composite image and the labeled categories are shown in Fig. 6. This data set is of a high spatial resolution, which brings great difficulty to the classification.

3) Xuzhou HYSPEX Data Set: The Xuzhou data set was collected by an airborne HYSPEX hyperspectral camera over the Xuzhou periurban site in November 2014. This data set consists of 500×260 pixels, with a very high spatial resolution of 0.73 m/pixel. The number of spectral bands used in the experiment was 436, after removing the noisy bands ranging from 415 to 2508 nm. The scene is periurban and is characterized by nine categories, including crops, vegetation, man-made structures, and coal fields. The pseudocolor composite image and the labeled categories are shown in Fig. 7. The very high spatial resolution and the complex mixed categories make this data set a challenging data set for classification.

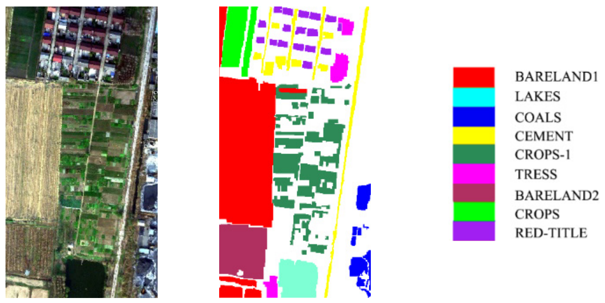

Fig. 7. Pseudocolor images and the corresponding ground truth of the Xuzhou HYSPEX data set.

B. Parameters of CapsNet

Due to the computation of all the capsules in the graphics processing unit, the required storage is larger than for conventional CNN, and the batch size is limited by the graphics memory. Therefore, the batch size is set as 70 with shuffle enabled. The Adam optimization algorithm was also chosen as the optimizer [51]. The number of capsules in the category layer is consistent with the number of categories, which is predetermined explicitly by a specific classifier; hence, the number of category capsules was set as 16, 9, and 9 for the AVIRIS, ROSIS, and HYSPEX data sets, respectively.

The influence of different filter sizes is also studied, which is linked to the size of the receptive field. Moreover, different combinations of capsule number and vector length were utilized to explore the optimal network settings. With regard to the primary capsule number, we chose 24, 32, and 40 as the value scope, but it should be noted this value is the number of capsule groups rather than the total number of primary capsules. The number of category capsules is one of the crucial factors for the final output, so we chose 8, 16, and 24 as the value scope, referring to the baseline given by Sabour *et al.* [20]. The three different parameter combinations were used to form 27 networks, and 81 experiments were performed on the three data sets, as listed in Table II.

The receptive field of CapsNet, which denotes the range of the original data that the neurons can "feel," is determined by the stride and filter size. From Table II, we can see that the optimal size is different for the three data sets. The best size is 1×5 for the ROSIS data set and 1×9 for the AVIRIS and HYSPEX data sets. As for the AVIRIS data, its categories have poor separability and the spectral range is wider. Therefore, a greater receptive field should be considered to increase the range of the features for HYSPEX. With a smaller band range, the ROSIS data set would lose detailed information when a large filter size is chosen, and the results show that the OA declines sharply when the filter size is set as 1×9 on the ROSIS data set. That is to say, the optimal filter size is decided by the category separability and spectral range. The ROSIS data set needs a smaller receptive field size to extract more detailed features, while the other two data sets need greater receptive fields to explore a larger scope of spectral information. As aforementioned, the number of capsules in the primary capsule layer equals $\mathrm{CH}_2 \times N_2$, where the implementation of these capsules is to

TABLE II
DIFFERENT PARAMETERS FOR THE THREE HYPERSPECTRAL DATA SETS

Filter size	Primary capsule	Category vector	AVIRIS OA	AVIRIS Kappa coefficient	ROSIS OA	ROSIS Kappa coefficient	HYSPEX OA	HYSPEX Kappa coefficient
1 × 3	24	8	80.67	0.7835	89.79	0.8667	94.17	0.921
		16	82.62	0.8022	91.69	0.8859	95.13	0.9367
		24	81.91	0.7973	91.36	0.8833	94.74	0.9239
	32	8	81.07	0.7729	88.3	0.8596	96.19	0.9333
		16	81.2	0.7752	89.38	0.8644	94.65	0.9268
		24	79.06	0.7639	87.53	0.842	93.52	0.9141
	40	8	77.32	0.7448	84.38	0.8017	92.38	0.8916
		16	80.04	0.7817	85.42	0.8203	94.23	0.9184
		24	79.57	0.7763	86.24	0.8337	93.93	0.9047
1 × 5	24	8	85.89	0.8303	90.01	0.8831	94.5	0.9267
		16	87.14	0.8547	**94.08(2)**	**0.9157**	95.08	0.9302
		24	86.25	0.8419	93.63	0.9089	95.88	0.9345
	32	8	83.12	0.8026	92.38	0.8973	96.91	0.9278
		16	85.45	0.8322	**94.48(1)**	**0.9166**	97.03	0.9402
		24	85.32	0.8303	93.61	0.9059	**97.31(3)**	**0.9536**
	40	8	81.77	0.7892	91.02	0.8835	93.85	0.9037
		16	82.81	0.8017	90.88	0.8786	94.7	0.9192
		24	80.26	0.773	91.2	0.8842	93.03	0.8956
1 × 9	24	8	86.55	0.8341	89.87	0.871	96.84	0.947
		16	**88.31(1)**	**0.8612**	**93.84(3)**	**0.9027**	97.01	0.9548
		24	**87.72(3)**	**0.8469**	92.61	0.8878	**98.13(1)**	**0.9673**
	32	8	87.31	0.839	89.05	0.8535	95.59	0.9377
		16	**87.94(2)**	**0.8512**	91.36	0.8726	**97.96(2)**	**0.9665**
		24	86.23	0.8375	90.98	0.8639	97.08	0.9428
	40	8	84.02	0.8164	85.36	0.824	94.22	0.9215
		16	82.11	0.7903	87.11	0.8431	96.57	0.9433
		24	80.8	0.7635	85.95	0.8333	95.13	0.9327

*The best results are shown in bold, where (1) represents the best result, (2) represents the second-best result, and (3) represents the third-best result.

define the number of output channels in the previous CH_2 layer and then reconstruct the $CH_2 \times N_2$ capsules. As shown in Table II, the different CH_2 layers have a huge impact on the final performance. The best dimension of CH_2 is 24 for the AVIRIS and HYSPEX data sets and 32 for the ROSIS data set. From all the results, the general trend is that the larger the dimension of CH_2, the lower the classification accuracy. When CH_2 is set as 40, the results obtained with all three data sets become worse in general.

In CapsNet, the dimension of the category capsule represents the contribution of all the extracted features toward the final prediction. The output of the category layer should be calculated by the squashing function (4), whose output denotes the final classification result. The optimal dimension of category capsule is more complicated. Most of the experiments show that a low dimension cannot represent all the features adequately, which is the same for an overlarge dimension. The OA obtained by an 8-D category capsule and a 24-D category capsule is lower than for the 16-D capsule in most cases.

Overall, the best parameters are 1×9, 24, and 16 for the filter size, the number of primary capsule groups, and the dimension of the category capsule, respectively, for the AVIRIS data set, yielding the OA as 88.31% and the kappa coefficient as 0.8612. For the ROSIS data set, the best parameters are 1×9, 32, and 16, resulting in the OA as 94.48% and the kappa coefficient as 0.9166. For the HYSPEX data set, the best parameters are 1×9, 24, and 24, with the OA being 98.13% and the kappa coefficient being 0.9673. Comparing the results of all the experiments, the best parameters were determined as 32 and 16 for the number of primary capsule groups and the dimension of the category capsule, respectively, for all three data sets. The filter size was set as 1×9 for the AVIRIS and HYSPEX data sets and 1×5 for the ROSIS data set. With these settings, the OAs of the AVIRIS, ROSIS, and HYSPEX data sets are 87.94%, 94.48%, and 97.96%, respectively.

C. Comparison With the State of the Art

To the best of our knowledge, none of the previous works have introduced CapsNet into hyperspectral imagery classification. Therefore, in this section, the results of CapsNet are compared with the state of the art.

The architecture of CapsNet for the ROSIS data set was realized as follows: the input size was set as 1×103, which just included the spectral features; the stride of the first layer was set as 1; the kernel size of the ReLU Conv layer and primary capsule layer was set as 1×5; and the number of channels was set as 64 and 32×8 for the first two layers, respectively. In the primary capsule layer, the stride was set as 2 and the number of capsules was set as 32×47, with each capsule an 8-D vector. The number of capsules in the category capsule layer was set to 9, and the length of each capsule was set as 16. The architecture of CapsNet for the AVIRIS data set was realized as follows: the input size was 1×200, which just included the spectral features; the stride of the first layer was set as 1; the kernel size of the ReLU Conv layer and primary capsule layer was set as 1×9; and the number of channels

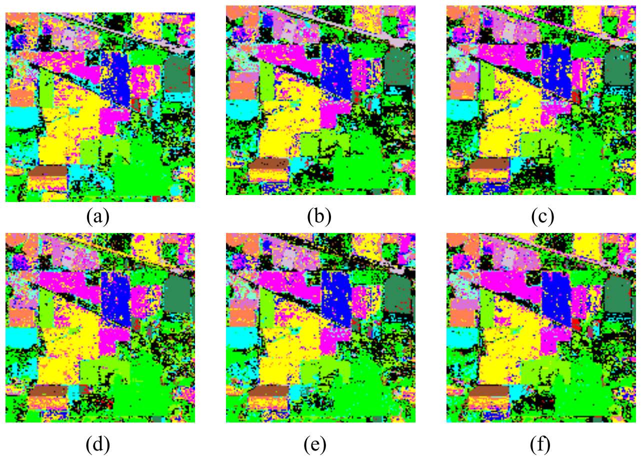
Fig. 8. Results obtained by different approaches for the AVIRIS data set. (a) SVM (linear). (b) SVM (RBF). (c) SDA. (d) DBN. (e) CNN. (f) CapsNet.

was set as 64 and 32 × 8 for the first two layers, respectively. In the primary capsule layer, the stride was set as 2 and the number of capsules was set as 32 × 92, with each capsule an 8-D vector. The number of capsules in the category capsule layer was set as 16 and the length of each capsule was set as 16. The architecture of CapsNet for the HYSPEX data set was realized as follows: the input size was set as 1 × 436, which just included the spectral features; the stride of the first layer was set as 2; the kernel size of the ReLU Conv layer and primary capsule layer was set as 1 × 9; and the number of channels was set as 64 and 32 × 8 for the first two layers, respectively. In the primary capsule layer, the stride was set as 3 and the number of capsules was set as 32 × 68, with each capsule an 8-D vector. The number of capsules in the category capsule layer was set as 9 and the length of each capsule was set as 16. The training data were randomly sampled as 10% of all the labeled data.

The minibatch size for training was set as 80, the learning rate was set to 0.01, and the downweighing factor (λ) was set to 0.5. The conventional classifiers, such as SVM, and the mainstream deep learning algorithms of SDA, DBN, and CNN were chosen as comparative methods. The SVM used a Gaussian radial basis function (RBF) kernel or a linear kernel function. SDA and DBN used two layers and CNN was based on a 1-D convolution operation. The hyperparameters of each network were chosen empirically.

Table III shows that Caps-TripleGAN and CapsNet obtain the best and second-best result of 90.02% and 87.94%, respectively, in the experiment with the AVIRIS data set. Classification maps are shown in Fig. 8. The linear kernel SVM generates the worst result. Each deep learning algorithm offers a better performance than SVM with two different kernel functions. The training time of the CapsNet is longer than that of the CNN, which means that the CapsNet is more computationally expensive. In this experiment, the training process of Caps-TripleGAN is 18 min longer, which is caused by the sequential training process on two parts, one is training of TripleGAN and the other is CapsNet.

For the ROSIS data set, we can see that our proposed Caps-TripleGAN can produce the best performance with 96.31% OA. CapsNet generates the second best result of 94.48%, as shown in Table IV. Classification maps are shown in Fig. 9. The experiments were, in fact, repeated ten times over the randomly split training and test data. During the ten experiments, the performance of DBN was similar to and even surpassed CapsNet. That is, there is no significant advantage for CapsNet in the ROSIS data set when compared with DBN. Nevertheless, the performance of CapsNet is better than that of CNN. The training time of Caps-TripleGAN is the longest and the difference between Caps-TripleGAN and CNN is more significant on AVIRIS data set which is caused by the use of more training samples.

Table V shows that Caps-TripleGAN and CapsNet obtain the best and second-best result of 98.42% and 97.96%, respectively, on the HYSPEX data set. All the methods achieve results above 90% and generate similar classification maps in Fig. 10. This is due to more spectral information in this data set. With the rich spectral characteristics, CapsNet is more effective at excavating efficient features for classification and

TABLE III
OAs and Kappa Coefficients Obtained by Different Approaches for the AVIRIS Data Set

Method	OA	Kappa coefficient	Elapsed time (min)
SVM(linear)	76.49	0.7291	5
SVM(RBF)	78.64	0.7734	8.5
SDA	82.28	0.7833	6.5
DBN	80.11	0.7794	8
CNN	82.39	0.7985	9.5
CapsNet	87.94	0.8512	13
Caps-TripleGAN	90.02	0.8924	27.5

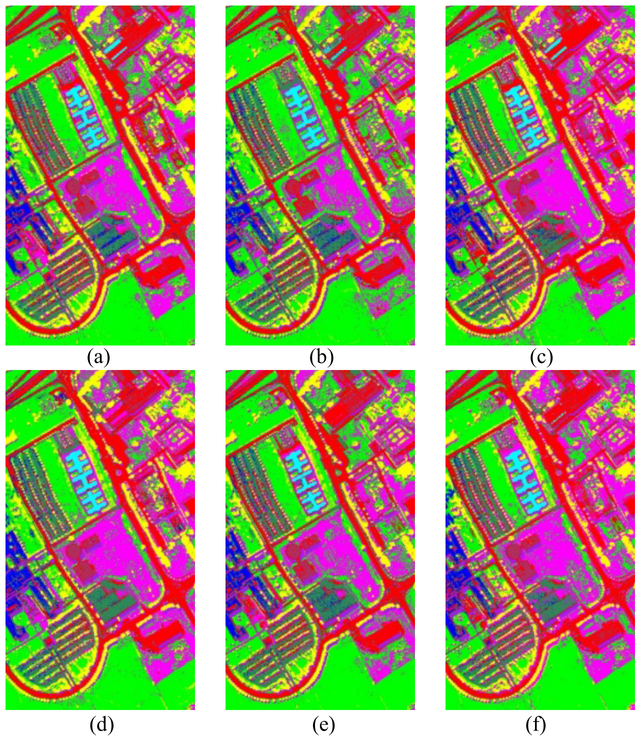
Fig. 9. Results obtained by the different approaches for the ROSIS data set. (a) SVM (linear). (b) SVM (RBF). (c) SDA. (d) DBN. (e) CNN. (f) CapsNet.

TABLE IV

OAS AND KAPPA COEFFICIENTS OBTAINED BY DIFFERENT APPROACHES FOR THE ROSIS DATA SET

Method	OA	Kappa coefficient	Elapsed time (min)
SVM (linear)	90.89	0.8776	7.5
SVM (RBF)	94.24	0.9233	12
SDA	92.27	0.9014	11.5
DBNs	94.38	0.9256	14
CNN	93.29	0.9062	13.5
CapsNet	94.48	0.9166	22
Caps-TripleGAN	96.31	0.9511	38

TABLE V

OAS AND KAPPA COEFFICIENTS OF DIFFERENT APPROACHES FOR THE HYSPEX DATA SET

Method	OA	Kappa coefficient	Elapsed time (min)
SVM (linear)	95.97	0.9412	12.5
SVM (RBF)	97.01	0.9584	19
SDA	96.51	0.9468	18.5
DBN	97.37	0.9620	21.5
CNN	96.68	0.9579	26
CapsNet	97.96	0.9665	42
Caps-TripleGAN	98.42	0.9799	79

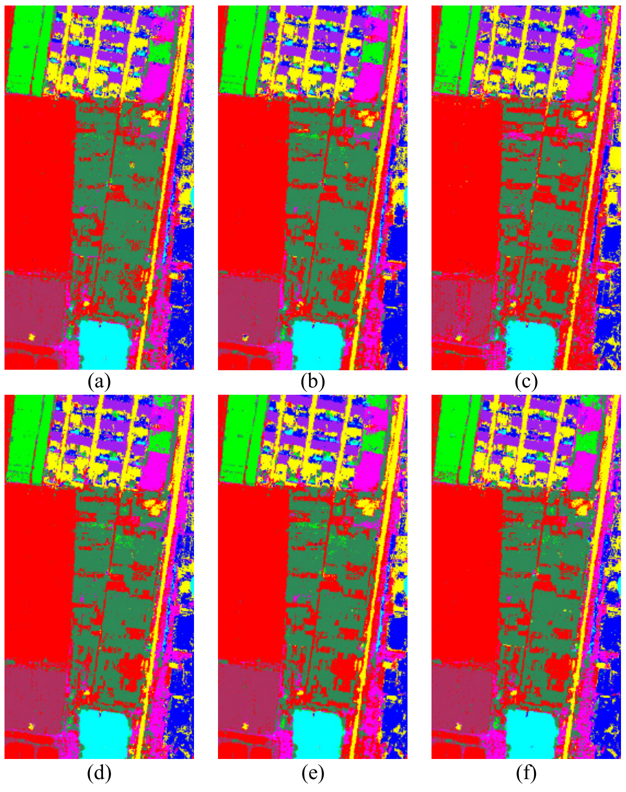

Fig. 10. Results of different approaches for the HYSPEX data set. (a) SVM (linear). (b) SVM (RBF). (c) SDA. (d) DBN. (e) CNN. (f) CapsNet.

Caps-TripleGAN is more faithful in predicting the category labels. The training time of Caps-TripleGAN is three times longer than CNN, and the difference between these two algorithms is the largest on this data set due to the large training set and high spectral dimension.

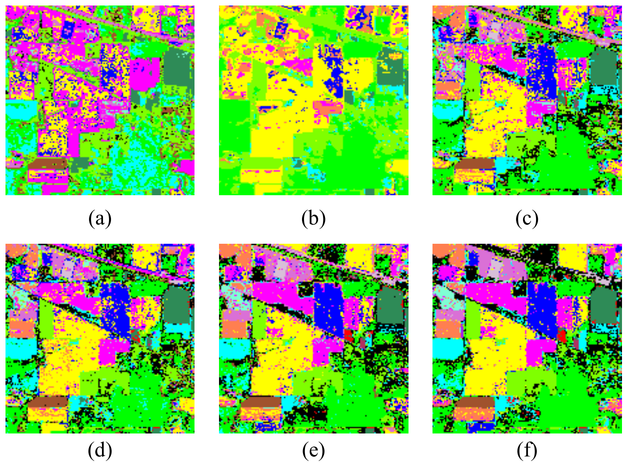

Fig. 11. Results of Caps-TripleGAN for the AVIRIS data set. (a) TripleGAN 1%. (b) TripleGAN 1% + CapsNet. (c) TripleGAN 5%. (d) TripleGAN 5% + CapsNet. (e) TripleGAN 10%. (f) TripleGAN 10% + CapsNet.

TABLE VI

OAS AND KAPPA COEFFICIENTS OBTAINED BY CAPS-TRIPLEGAN FOR THE AVIRIS DATA SET

Method	OA	Kappa
Triple GAN 1%	44.41	0.3943
Triple GAN 1%+CapsNet	32.20	0.3072
Triple GAN 5%	77.87	0.7319
Triple GAN 5%+CapsNet	75.46	0.7032
Triple GAN 10%	86.94	0.8440
Triple GAN 10%+CapsNet	90.02	0.8924

D. Training on a Small Number of Labeled Samples With Caps-TripleGAN

To explore the impact of training size, three experiments were designed with 1%, 5%, and 10% of labeled samples. At the beginning, TripleGAN was trained with the true training data. When the game was converged or the training epochs approached to a certain number, the output from the generator was regarded as a reliable data pair (x, y). CapsNet was then trained by the hybrid of the generated and real data pairs. We have found that the labeled samples were not sufficient in our experiments, so fake labels were generated through the classifier for the unlabeled samples, and assigned these data pairs a positive label when training the D of TripleGAN. That is to say, the positive data pairs were a hybrid of the real data pairs (x_r, y_r) and the assigned data pairs by the classifier (x_c, y_c).

For the AVIRIS data set, Fig. 11 and Table VI show that both CapsNet and TripleGAN provide the worst performance on 1% training data, for which the OAs are 44% and 32%, respectively. Except with 1% and 5% training data, Caps-TripleGAN shows a better performance than just using TripleGAN. Caps-TripleGAN with 10% training samples for the AVIRIS data set generates the best performance. The performance of CapsNet depends on the reliability of the generator in TripleGAN. When training on 1% labeled samples, some categories have inadequate labeled data, which causes poor training of TripleGAN and poor usability of the generated data.

For the ROSIS data set, the results in Fig. 12 and Table VII show that, as with the AVIRIS data and 1% training samples,

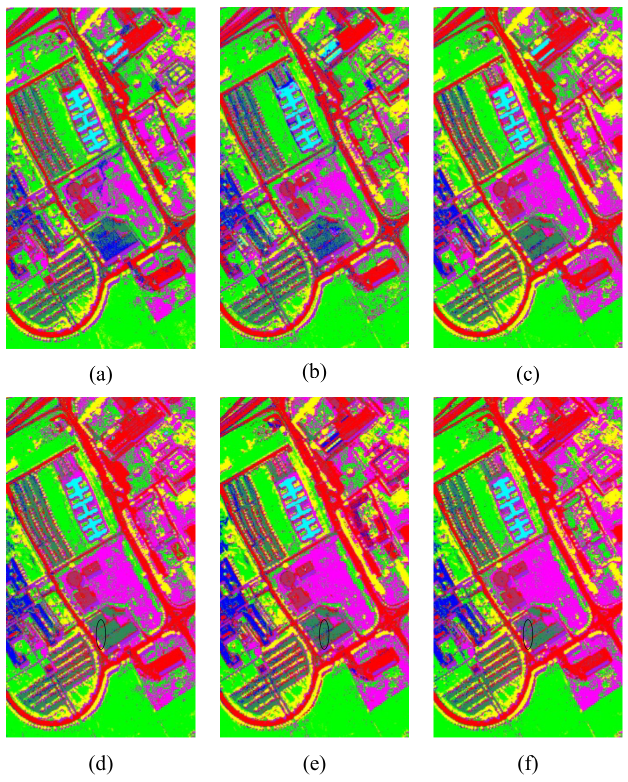

Fig. 12. Results of Caps-TripleGAN for the ROSIS data set. (a) TripleGAN 1%. (b) TripleGAN 1% + CapsNet. (c) TripleGAN 5%. (d) TripleGAN 5% + CapsNet. (e) TripleGAN 10%. (f) TripleGAN 10% + CapsNet.

TABLE VII

OAs and Kappa Coefficients Obtained by Caps-TripleGAN for the ROSIS Data Set

Method	OA	Kappa
Triple GAN 1%	83.36	0.7539
Triple GAN 1%+CapsNet	83.11	0.7163
Triple GAN 5%	90.33	0.8781
Triple GAN 5%+CapsNet	93.58	0.9047
Triple GAN 10%	94.02	0.9108
Triple GAN 10%+CapsNet	96.31	0.9511

TripleGAN obtains a better performance than Caps-TripleGAN. As shown by the black ellipse in Fig. 11, the pixels are not well recognized by TripleGAN (caused by the unbalance of samples between the painted metal sheet and gravel), so the pixels in the black ellipse are wrongly classified on both 5% and 10% training samples; however, this is improved when CapsNet takes part in the classification. In other words, TripleGAN provides sufficient training samples for the capsule training, and CapsNet, in turn, assists TripleGAN.

In the HYSPEX experiment, Fig. 13 and Table VIII show that Caps-TripleGAN offers a better performance than just TripleGAN on 1%, 5%, and 10% training data. The best OA of 98.42% is obtained by Caps-TripleGAN on 10% training data. It is demonstrated that a fully trained TripleGAN can generate reliable labeled samples to assist the training of CapsNet as in other experiments.

E. Comparison With HSI-GANs and HSI-CapsuleNet

In Table I, the proposed Caps-TripleGAN is compared with HS-GAN$_1$ [45], HS-GAN$_2$ [43], HS-GAN$_3$ [46], CapsuleNet-HS [22], InfoGAN [32], ACGAN [33], CatGAN [35],

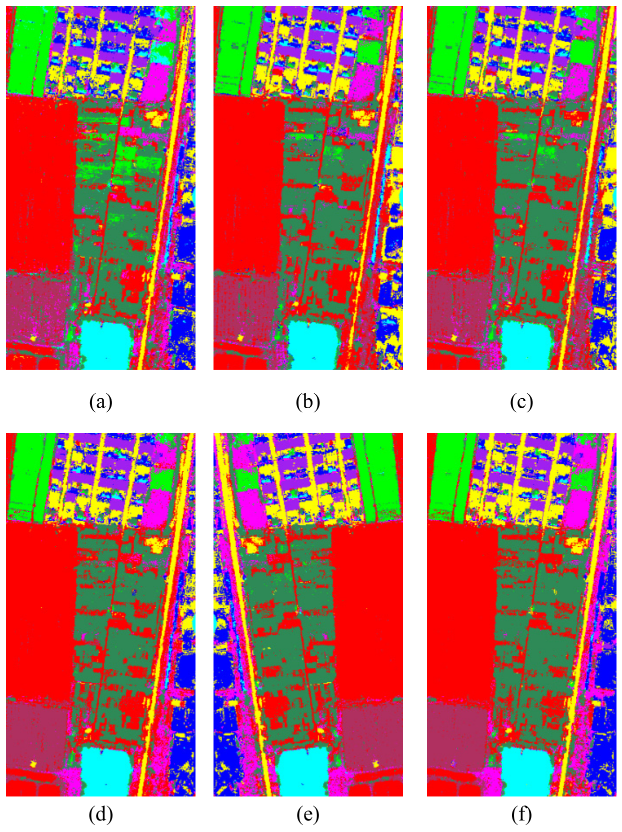

Fig. 13. Results of Caps-TripleGAN for the HYSPEX data set. (a) TripleGAN 1%. (b) TripleGAN 1% + CapsNet. (c) TripleGAN 5%. (d) TripleGAN 5% + CapsNet. (e) TripleGAN 10%. (f) TripleGAN 10% + CapsNet.

TABLE VIII

OAs and Kappa Coefficients Obtained by Caps-TripleGAN for the HYSPEX Data Set

Method	OA	Kappa
Triple GAN 1%	90.15	0.8761
Triple GAN1%+CapsNet	91.57	0.8925
Triple GAN 5%	92.80	0.9085
Triple GAN5%+CapsNet	94.61	0.9369
Triple GAN 10%	97.17	0.9640
Triple GAN 10%+CapsNet	98.42	0.9799

TSVM [24], Graph-based SSL [25], and cotraining [26]. TSVM, Graph-Based, and cotraining used in the experiments are implemented with the scikit-learn package. The other algorithms come with the pytorch platform. Cotraining we utilized is based on single view and multiclassifier. The two classifiers, SVM and KNN, have been employed to construct the cotraining algorithm. HS-GAN$_1$, HS-GAN$_2$, HS-GAN$_3$, and CapsuleNet-HS are carried out using the experimental parameters from references. Due to the lack of real application of ACGAN, InfoGAN, and ACGAN on hyperspectral imagery, the detailed parameters are implemented as the same as the TripleGAN. As shown in Table IX, the InfoGAN shows the worst performance on the three hyperspectral images among these GAN frameworks. CatGAN is better than InfoGAN and ACGAN on AVIRIS and ROSIS data sets. The OAs of TSVM are lower than that of other conventional algorithms on three hyperspectral images. Cotraining provides the best

TABLE IX
OAS AND KAPPA COEFFICIENTS FROM HSI-GANS AND HSI-CAPSULENET ON THE THREE HYPERSPECTRAL DATA SETS

Algorithm	AVIRIS		ROSIS		HYSPEX	
	OA	Kappa	OA	Kappa	OA	Kappa
InfoGAN	72.06	0.7313	89.92	0.8603	93.58	0.8983
CatGAN	77.27	0.7368	91.93	0.88	95.73	0.9247
ACGAN	73.95	0.712	90.57	0.8674	95.81	0.9337
HS-GAN$_1$	74.51	0.7003	92.33	0.9069	97.22	0.9428
HS-GAN$_2$	82.49	0.7932	92.47	0.892	96.34	0.938
HS-GAN$_3$	89.21	0.8731	96.08	0.9433	97.63	95.22
CapsuleNet-HS	83.47	80.15	95.79	0.9147	97.94	0.9618
Caps-Triple	**90.02**	**0.8924**	**96.31**	**0.9511**	**98.42**	**0.9799**
TSVM	76.55	0.736	91.62	0.9098	94.7	0.9244
Graph-based SSL	81.23	0.7936	93.58	0.9083	94.85	0.9142
Co-training	88.16	0.86	95.83	0.9283	98.16	0.9556

*The best results are shown in bold

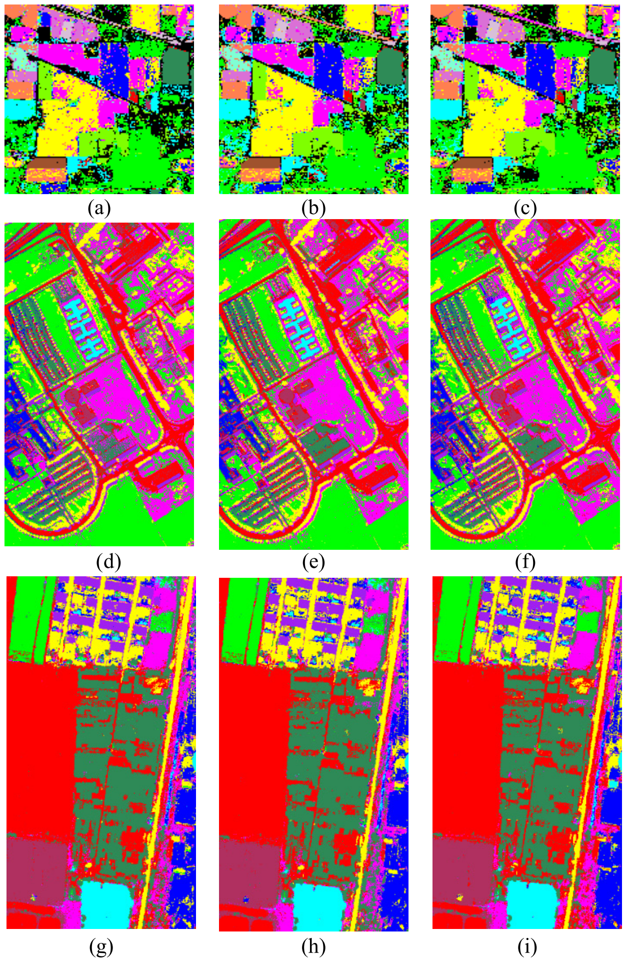

Fig. 14. Results of Caps-TripleGAN with different window sizes. (a) 3 × 3_AVIRIS. (b) 5 × 5_AVIRIS. (c) 9 × 9_AVIRIS. (d) 3 × 3_ROSIS. (e) 5 × 5_ROSIS. (f) 9 × 9_ROSIS. (g) 3 × 3_HYSPEX. (h) 5 × 5_HYSPEX. (i) 9 × 9_HYSPEX.

performance on conventional SSL algorithms, which is lower than Caps-TripleGAN. The Caps-TripleGAN is slightly superior to the cotraining model. In comparison with the recently proposed classifiers, HS-GAN$_3$ obtains the secondary higher accuracy on AVIRIS and ROSIS data sets which is lower than Caps-TripleGAN by less than 1%. HS-GAN$_3$ is improved by introducing the neighborhood majority voting. After spectral classification, the majority voting strategy is carried out to improve classification accuracy by using spatial features. However, the combination of CapsuleNet and TripleGAN is more helpful with performance improvement.

TABLE X
OAS AND KAPPA COEFFICIENTS FROM CAPS-TRIPLEGAN WITH DIFFERENT WINDOW SIZES

Method	OA	Kappa
AVIRIS_no spatial	90.02	0.8924
AVIRIS_3× 3	91.35	0.9031
AVIRIS_5× 5	92.34	0.9140
AVIRIS_9× 9	90.02	0.8861
ROSIS_no spatial	96.31	0.9511
ROSIS_3× 3	96.50	0.9537
ROSIS_5× 5	96.17	0.9490
ROSIS_9× 9	96.55	0.9542
HYSPEX_no spatial	98.42	0.9799
HYSPEX_3× 3	98.71	0.9836
HYSPEX_5× 5	98.97	0.9869
HYSPEX_9× 9	99.02	0.9876

F. Exploring Spatial Information With Caps-TripleGAN.

In this experiment, we explored the use of spatial information in the Caps-TripleGAN framework. The first three principal components of the PCA-analyzed images were utilized to extract the spatial information. Three different window sizes were used to concatenate with the spectral characteristics: 3×3, 5×5, and 9×9. We also note that only 10% of the labeled samples were used in this experiment.

As shown in Fig. 14 and Table X, accuracies are improved by spatial feature utilization for all the three data sets, and the size of the neighborhood has a great influence. The ideal size for the AVIRIS data set is 5×5, for which the OA is 92.34%. When the size continues to increase up to 9×9, the OA is similar to the result without spatial features, but the kappa coefficient is lower. From Fig. 13(a)–(c), the area of blue shows that the salt-and-pepper noise is the minimum at 5×5, and the areas marked with the red ellipse show that the best size is 9×9. That is to say, the optimal window size depends on the size of ground objects. For the ROSIS data results, the improvement is not significant when the size is 5×5, and the OA is lower than with no spatial utilization. The best OA of 96.55% and kappa coefficient of 0.9542 are obtained when the size equals 9×9. The results for the HYSPEX data set show that the best size is 9×9, yielding an OA of 99.02% and a kappa coefficient of 0.9876. Furthermore, we can see that the spatial features can be learned by the Caps-TripleGAN framework, and the optimal window size is independent of the spatial resolution.

V. Conclusion

In this paper, an innovative network named CapsNet was proposed, which take features' locations and directions into account in feature extraction. CapsNet was modified to adapt to hyperspectral image classification. From the results for the three standard hyperspectral data sets, we have shown that CapsNet can outperform other methods. TripleGAN was adopted to boost the performance of CapsNet on small numbers of training samples, and a framework named Caps-TripleGAN, combining TripleGAN and CapsNet, was proposed for hyperspectral image classification with a 1-D CNN. The results showed that the reliable generator in TripleGAN can improve the performance of CapsNet. Moreover, spatial information is used in the Caps-TripleGAN framework and the results showed that spatial features can be learned by the generator and can further improve the classification performance. There are two limitations of Caps-TripleGAN as follows.

1) Spatial–spectral features should be imported integrally in the end-to-end model not using handcrafted features. Its performance will be improved if considering spatial–spectral features simultaneously rather than concatenating the handcrafted spatial and spectral features.
2) The sequential training model of Caps-TripleGAN is inefficient and the thread waiting wastes computing resources.

In our future work, more experiments will be conducted by using spatial–spectral data block, and the spatial–spectral network architecture will be explored on the generator. The distributed training process with GPU will also be utilized for Caps-TripleGAN to improve its efficiency. In addition, more effective sample selection strategy will be investigated to increase the reliability of the created samples.

Acknowledgment

The authors would like to thank Prof. D. Landgrebe, P. Gamba, and K. Tan for providing the data used in the experiments.

References

[1] A. J. Brown *et al.*, "Hydrothermal formation of clay-carbonate alteration assemblages in the Nili fossae region of Mars," *Earth Planetary Sci. Lett.*, vol. 297, pp. 174–182, Aug. 2010.
[2] X. Kang, S. Li, and J. A. Benediktsson, "Spectral-spatial hyperspectral image classification with edge-preserving filtering," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 5, pp. 2666–2677, May 2014.
[3] A. J. Brown, "Spectral curve fitting for automatic hyperspectral data analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 6, pp. 1601–1608, Jun. 2006.
[4] A. J. Brown, B. Sutter, and S. Dunagan, "The MARTE VNIR imaging spectrometer experiment: Design and analysis," *Astrobiology*, vol. 8, no. 5, pp. 1001–1011, 2008.
[5] B. P. Garcia-Salgado and V. Ponomaryov, "Feature extraction scheme for a textural hyperspectral image classification using gray-scaled HSV and NDVI image features vectors fusion," in *Proc. Int. Conf. Electron., Commun. Comput.*, Feb. 2016, pp. 186–191.
[6] Q. Li, S. Qi, Y. Shen, D. Ni, H. Zhang, and T. Wang, "Multispectral image alignment with nonlinear scale-invariant keypoint and enhanced local feature matrix," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 7, pp. 1551–1555, Jul. 2015.
[7] 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.
[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
[9] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017.
[10] 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.
[11] 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.
[12] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[13] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[14] M. H. Rafiei, W. H. Khushefati, R. Demirboga, and H. Adeli, "Supervised deep restricted Boltzmann Machine for estimation of concrete," *ACI Mater. J.*, vol. 114, no. 2, pp. 237–244, 2017.
[15] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[16] M. He, B. Li, and H. Chen, "Multi-scale 3D deep convolutional neural network for hyperspectral image classification," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2018, pp. 3904–3908.
[17] W. Song, S. Li, L. Fang, and T. Lu, "Hyperspectral image classification with deep feature fusion network," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 6, pp. 3173–3184, Jun. 2018.
[18] H. Zhang and Y. Li, "Spectral-spatial classification of hyperspectral imagery based on deep convolutional network," in *Proc. Int. Conf. Orange Technol. (ICOT)*, Dec. 2016, pp. 44–47.
[19] 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.
[20] S. Sabour, N. Frosst, and G. E. Hinton. (2017). "Dynamic routing between capsules." [Online]. Available: https://arxiv.org/abs/1710.09829
[21] F. Deng, S. Pu, X. Chen, Y. Shi, T. Yuan, and S. Pu, "Hyperspectral image classification with capsule network using limited training samples," *Sensors*, vol. 18, no. 9, p. 3153, 2018.
[22] M. E. Paoletti *et al.*, "Capsule networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2145–2160, Apr. 2019.
[23] M. Marconcini, G. Camps-Valls, and L. Bruzzone, "A composite semisupervised SVM for classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 6, no. 2, pp. 234–238, Apr. 2009.
[24] U. Maulik and D. Chakraborty, "Learning with transductive SVM for semisupervised pixel classification of remote sensing imagery," *ISPRS J. Photogram. Remote Sens.*, vol. 77, pp. 66–78, Mar. 2013.
[25] G. Camps-Valls, T. V. B. Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 10, pp. 3044–3054, Oct. 2007.
[26] L. I. Ji-Ming, S. Jia, and Y.-B. Peng, "Hyperspectral data classification with spectral and texture features by co-training algorithm," *Opto-Electron. Eng.*, vol. 39, no. 11, pp. 88–94, 2012.
[27] L. Chapel, T. Burger, N. Courty, and S. Lefevre, "PerTurbo manifold learning algorithm for weakly labeled hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 4, pp. 1070–1078, Apr. 2014.
[28] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 4, 2014, pp. 3581–3589.

[29] P. Liu, L. Xiao, J. Zhang, and B. Naz, "Spatial-Hessian-feature-guided variational model for pan-sharpening," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 4, pp. 2235–2253, Apr. 2016.

[30] Z. He, J. Li, K. Liu, L. Liu, and H. Tao, "Kernel low-rank multitask learning in variational mode decomposition domain for multi-/hyperspectral classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 7, pp. 4193–4208, Jul. 2018.

[31] M. Mirza and S. Osindero. (2014). "Conditional generative adversarial nets." [Online]. Available: https://arxiv.org/abs/1411.1784?context=cs.CV

[32] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. (2016). "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets." [Online]. Available: https://arxiv.org/abs/1606.03657

[33] A. Odena, C. Olah, and J. Shlens. (2016). "Conditional image synthesis with auxiliary classifier GANs." [Online]. Available: https://arxiv.org/abs/1610.09585

[34] A. Radford, L. Metz, and S. Chintala. (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1511.06434

[35] S. Wang and L. Zhang. (2017). "CatGAN: Coupled adversarial transfer for domain generation." [Online]. Available: https://arxiv.org/abs/1711.08904

[36] K. Enomoto *et al.*, "Filmy cloud removal on satellite imagery with multispectral conditional generative adversarial nets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 1533–1541.

[37] D.-Y. Lin, Y. Wang, G.-L. Xu, and K. Fu, "Synthesizing remote sensing images by conditional adversarial networks," in *Proc. Geosci. Remote Sens. Symp.*, Jul. 2017, pp. 48–50.

[38] N. Merkle, P. Fischer, S. Auer, and R. Müller, "On the possibility of conditional adversarial networks for multi-sensor image matching," in *Proc. IGARSS*, Jul. 2017, pp. 2633–2636.

[39] P. Ghamisi and N. Yokoya, "IMG2DSM: Height simulation from single imagery using conditional generative adversarial net," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 5, pp. 794–798, May 2018.

[40] N. Bayramoglu, M. Kaakinen, L. Eklund, and J. Heikkilä, "Towards virtual H&E staining of hyperspectral lung histology images using conditional generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, Oct. 2017, pp. 64–71.

[41] S. Xu, X. Mu, D. Chai, and X. Zhang, "Remote sensing image scene classification based on generative adversarial networks," *Remote Sens. Lett.*, vol. 9, no. 7, pp. 617–626, 2018.

[42] D. Lin, K. Fu, Y. Wang, G. Xu, and X. Sun, "MARTA GANs: Unsupervised representation learning for remote sensing image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 11, pp. 2092–2096, Nov. 2017.

[43] Y. Zhan, D. Hu, Y. Wang, and X. Yu, "Semisupervised hyperspectral image classification based on generative adversarial networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 2, pp. 212–216, Feb. 2018.

[44] Z. He, H. Liu, Y. Wang, and J. Hu, "Generative adversarial networks-based semi-supervised learning for hyperspectral image classification," *Remote Sens.*, vol. 9, no. 10, p. 1042, Oct. 2017.

[45] Z. Lin, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Generative adversarial networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5046–5063, Sep. 2018.

[46] Y. Zhan *et al.*, "Semi-supervised classification of hyperspectral data based on generative adversarial networks and neighborhood majority voting," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2018, pp. 5756–5759.

[47] M. Gong, X. Niu, P. Zhang, and Z. Li, "Generative adversarial networks for change detection in multispectral imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 12, pp. 2310–2314, Nov. 2017.

[48] J. Zhang, L. Chen, L. Zhuo, X. Liang, and J. Li, "An efficient hyperspectral image retrieval method: Deep spectral-spatial feature extraction with DCGAN and dimensionality reduction using t-SNE-based NM hashing," *Remote Sens.*, vol. 10, no. 2, p. 271, 2018.

[49] C. Li, K. Xu, J. Zhu, and B. Zhang. (2017). "Triple generative adversarial nets." [Online]. Available: https://arxiv.org/abs/1703.02291

[50] 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.

[51] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

Xue Wang received the B.S. degree in geographic information system from the China University of Mining and Technology, Jiangsu, China, in 2014. He is currently pursuing the Ph.D. degree in photogrammetry and remote sensing with the China University of Mining and Technology, Xuzhou, China.

His research interests include hyperspectral imagery processing, deep learning, and ecological monitoring.

Kun Tan (SM'16) received the B.S. degree in information and computer science from Hunan Normal University, Hunan, China, in 2004, the joint Ph.D. degree in remote sensing from Columbia University, New York, NY, USA, in 2009, and the Ph.D. degree in photogrammetric and remote sensing from the China University of Mining and Technology, Xuzhou, China, in 2010.

From 2010 to 2018, he was with the Department of Surveying, Mapping and Geoinformation, China University of Mining and Technology. He is currently a Professor with East China Normal University. His research interests include hyperspectral image classification and detection, spectral unmixing, quantitative inversion of land surface parameters, and urban remote sensing.

Qian Du (S'98–M'00–SM'05–F17) received the Ph.D. degree in electrical engineering from the University of Maryland at Baltimore County, Baltimore, MD, USA, in 2000.

She is currently a Bobby Shackouls Professor with the Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS, USA. Her research interests include hyperspectral remote sensing image analysis, pattern recognition, and machine learning.

Dr. Du is a fellow of the International Society for Optics and Photonics (SPIE). She served as the Co-Chair for the Data Fusion Technical Committee of the IEEE Geoscience and Remote Sensing Society (GRSS) from 2009 to 2013 and the Chair for the Remote Sensing and Mapping Technical Committee of the International Association for Pattern Recognition (IAPR) from 2010 to 2014. She currently serves as the Editor-in-Chief for the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SEnsing.

Yu Chen received the master's degree in photogrammetry and remote sensing from the China University of Mining and Technology (CUMT), Xuzhou, China, in 2012, and the Ph.D. degree in earth and planetary science from the University of Toulouse, Toulouse, France, in 2017.

She was an Assistant Researcher with the Géoscience Environnement Toulouse (GET) Laboratory, French National Center for Scientific Research, Paris, France, in 2013. She is currently a Lecturer with CUMT. Her research interests include SAR interferometry, with a particular emphasis on its application for geophysical studies.

Peijun Du (M'07–SM'12) is currently a Professor of photogrammetry and remote sensing with the Department of Geographic Information Sciences, Nanjing University, Nanjing, China, and also the Deputy Director of the Key Laboratory for Satellite Surveying Technology and Applications, National Administration of Surveying and Geoinformation, China. His research interests include remote sensing image processing and pattern recognition; remote sensing applications; hyperspectral remote sensing information processing; multisource geospatial information fusion and spatial data handling; integration and applications of geospatial information technologies; and environmental information science.

Dr. Du is an Associate Editor of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS.