Semi-supervised classification of hyperspectral images based on two branch autoencoder

Z Lei$^1$, Z Yi$^1$, L Peng$^2$ and S X Hui$^1$

1. Beijing Forestry University, Beijing 100083, China
2. Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100094, China

The corresponding author’s e-mail zengyi@bjfu.edu.cn

Abstract. In this paper, we propose a novel semi-supervised method based on two branch autoencoder (TBAE) for hyperspectral images. A branch is classification function, and another branch is decoder function. Both of functions promote each other in training process. The encoder is fit for both classifier and decoder. The encoder and the decoder guide feature extraction from unlabeled samples. The encoder and the classifier guide classification for all labeled samples. We can get more information from unlabeled samples for classification. Finally, we use convolutional autoencoder (CAE) to extend TBAE and obtain TBCAE. Compared with ANN and CNN, TBAE and TBCAE have a better performance in the case of a small number of labeled samples. Use different datasets to verify these methods and print visualization of hidden layer and the reconstructed data. The results demonstrate that the proposed framework obtains credible results with a small number of labeled samples.

1. Introduction

Hyperspectral images (HSIs) gather detailed spectral feature from ground image by using a mass of narrow bands [1]. HSI has been applied by many applications, such as hyperspectral unmixing [2], classification [3], super resolution [4], target detection [5], band selection [6]. The classification of HSI is a common application. There are three main classification methods for HSIs: unsupervised learning, supervised learning, and semi-supervised learning (SSL) [7]. Unsupervised learning methods are insensitive to the priori knowledge, but relation ship between classes and clusters is not ensured [8]. Unsupervised learning methods focuses on feature representation in hyperspectral images, which can transform original high dimensional data into an appropriate low-dimensional space [9], such as principal component analysis (PCA) [10], independent component analysis (ICA) [11], sparse graph learning [12]. On the contrast, supervised learning methods have better performance in classification with using lots of training samples, because priori knowledge from labeled samples guide classifier to extract feature which is benefit for classification, such as support vector machines (SVM) [13], K nearest neighbor (KNN) [14], convolutional neural network (CNN) [15], logistic regression (LR) [16].

However, the vast labeled samples are collected for classification is difficult, time-wasting, and expensive [17]. Only a small number of samples are used to train the model can lead to model overfitting. Plentiful information form unlabeled samples are not exploited by supervised learning methods. SSL methods not only utilize a small amount labeled samples but also use a large number of unlabeled samples to alleviate overfitting phenomenon, such as graph [18], extended label propagation and rolling guidance filtering [19], fusion evidence entropy [20].
In recent years, the development of deep learning is unprecedented in many applications. Generative adversarial network (GAN) is introduced by paper [21], which includes generator and discriminator. GAN has been used in the SSL for classification of HSI. Paper [6] proposed HSGAN, the framework enables the automatic extraction of spectral features for HSI classification. Using unlabeled hyperspectral data to train HSGAN, the generator can generate the same dimension hyperspectral samples as the real data, whose discriminator contains the features from real samples and unlabeled samples, which can be used to classify HSIs with only a small number of labeled samples. Paper [22] proposed a SSL method, which used idea of paper [23] which use classifier about Softmax to replace the last layer of discriminator in GAN and add a fake category in classifier for generated data. Some research also proposed some SSLs with autoencoder. These papers [24][25] proposed stacked autoencoders (SAEs), whose unsupervised learning method is pre-training is carried combined Softmax classifier with unlabeled samples. Then, fine-tuning the network model with small labeled samples, the SAE-based classifier can learn implicit feature of HSI and achieve classification of HSI. These methods have a trait that they all did pre-training by unlabeled samples after that they only fine tuning the network by labeled samples, which is two steps for classification.

In this paper, we proposed a new method about autoencoder. To modify the structure autoencoder, we add a classifier with Softmax into hidden layer of autoencoder. We can train decoder by unlabeled samples and train classifier by labeled samples. We can cross training with these two structures whose loss functions include cross-entropy for classifier and mean-square error (MSE) for decoder. Therefore, we can take full advantage of unlabeled samples in this structure, and utilize a small number of samples to classify HSI. The features of encoder have a trait which not only be fit for classifier but also be fit for decoder. The decoder and the encoder promote each other. The structure of autoencoder has two branch named TBAE. Finally, we use convolutional autoencoder (CAE) to extend TBAE and obtain TBCAE.

The main contributions of this paper are as follow: (1) This paper proposes a novel semi-supervised framework TBAE for classification of HSI, which can utilize unlabeled samples to obtain more features and a small number of samples to classify HSI. (2) This paper offers detailed structure of TBAE. We use the University of Pavia data and Indian Pines data to verification compared with other methods, the results show that our method is obvious effect in a small number of samples. (3) We also use convolutional autoencoder to extend TBAE and obtain TBCAE. The experiment shows that this method also is fit for TBCAE.

The remainder of this paper is organized as follows. Section 2 briefly introduce the structure of TBAE and TBCAE. Section 3 mainly analyze experiments, which include manifestation compared with CNN and ANN, and visual experimental results. Section 4 discusses the experimental conclusion.

2. The classification by two branch autoencoder
The section discusses the structure of TBAE and TBCAE, which is applied to extract features. Then, we describe two loss functions in TBAE. Finally, we use convolution operation to replace the full connect and obtain TBCAE.

2.1. The structure of two branch autoencoder
Generally, autoencoder is unsupervised learning, which include encoder and decoder. The input of autoencoder is equal to the output of this autoencoder. The function of the encoder has a trait that the dimension of training data is compressed to low dimension space. The process is feature extraction. But these features may be not applied to classify HSI, we can add a classifier to guide this process for autoencoder. The structure of two branch autoencoder as Figure 1.
Figure 1. The schematic diagram of TBAE.

The input of encoder is full-band data, the output of decoder is reconstructed band, and the output of classifier is score of the corresponding category of a sample. The formula of encoder is as follow:

\[ x_{i+1} = f(w_i, x_i + b) \]  

where \( x_i \) is the input of the \( i \)-th layer in encoder, \( x_{i+1} \) is the output of the \( i \)-th layer, and also is the input of the \( (i+1) \)-th layer. \( w_i \) and \( b_i \) are weight of the \( i \)-th layer. The input of encoder is \( x_0 \). \( f(\cdot) \) is an activation function, which can adopt three typical activation functions which include Sigmoid, ReLU and tanh. The output of encoder is named \( z \), which is the input of decoder and also is the input of classifier. The formula of decoder is as follow:

\[ \hat{x}_{i+1} = f(\hat{w}_i, \hat{x}_i + \hat{b}_i) \]  

where \( \hat{x}_i \) is the input of the \( i \)-th layer in decoder, \( \hat{x}_{i+1} \) is the output of the \( i \)-th layer, and also is the input of the \( (i+1) \)-th layer. \( \hat{w}_i \) and \( \hat{b}_i \) are weight of the \( i \)-th layer. The input of decoder \( z \) is equal to \( \hat{x}_0 \). The output of decoder is \( \hat{x}_i \) which is very similar to \( x_0 \). The formula of classifier is as follow:

\[ \hat{y}_{i+1} = f(\hat{w}_i, \hat{x}_i + \hat{b}_i) \]  

where \( \hat{x}_i \) is the input of the \( i \)-th layer in classifier, \( \hat{x}_{i+1} \) is the output of the \( i \)-th layer, and also is the input of the \( (i+1) \)-th layer. \( \hat{w}_i \) and \( \hat{b}_i \) are weight of the \( i \)-th layer. The input of decoder \( z \) is equal to \( \hat{x}_0 \). The output of classifier connects with Softmax which is a multi-classifier.

2.2. Two loss function of TBAE

This section discusses two loss functions to decoder and classifier. MSE is used for decoder. cross entropy is used for classifier. The formula of MSE is as follow:

\[ MSE = \frac{1}{m} \sum_{i=1}^{m} (\hat{x}_i - x'_i)^2 \]  

where \( x'_0 \) is a value of the \( i \)-th location of \( x_0 \). \( \hat{x}_i \) also is a value of the \( i \)-th location of \( x_i \). \( m \) is the dimension of full-band data. The formula of cross entropy is as follow:

\[ loss = -\frac{1}{n} \sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \]  

where \( y \) is the expected output.
where $y_i$ is the $i$-th actual label in this batch. $\hat{y}_i$ is the $i$-th score of correct classification. $n$ is the number of a batch data. According to one-hot encoding, $y_i$ is “1”.

### 2.3. The structure of TBCAE

In this section, the theory of TBCAE is same as TBAE. The is a difference which is convolution autoencoder to replace the autoencoder. The formula of convolution is as follow:

$$
\bar{x}_{i+1} = f(\omega_i * \bar{x}_i + \bar{b}_i) \quad \{i = 0,1,2,...\}
$$

(6)

where $\bar{x}_i$ is the input of the $i$-th layer in a CNN, $\bar{x}_{i+1}$ is the output of the $i$-th layer in a CNN and $\bar{b}_i$ is the input of the $(i+1)$-th layer, $\omega_i$ is parameter of the $i$-th layer, and $\bar{b}_i$ is bias of the $i$-th layer. $*$ is convolution operation.

### 3. Experiment results

#### 3.1. Data sets

**3.1.1. University of Pavia data**

In our experiment, we adopt the University of Pavia data set, which is acquired by the ROSIS sensor and has 103 bands. The size of Pavia University is $610 \times 340$. But a large of samples in image contain no information and have to be discarded before the analysis. The number of useful samples is 42776. The spatial resolution is 1.3 meters. This data has a spectral coverage from 0.43 to 0.86 $\mu$m. Image ground truth is divided into 9 categories.

**3.1.2. Indian Pines data**

Indian Pines data was gathered by AVIRIS sensor over the Indian Pines test site in North-western Indiana and consists of $145 \times 145$ pixels and 224 spectral reflectance bands in the wavelength range the visible to infrared spectrum. This data is a subset of a larger one. The ground truth available is designated into sixteen classes and is not all mutually exclusive. The number of bands has been reduced to 200 to 200 by removing bands covering the region of water absorption: [104-108], [150-163], 220.

#### 3.2. Experimental setting

**3.2.1. The architectures of BTAE and BTCAE for Pavia University data**

Table 1 is our architecture of BTAE for Pavia University data. In the architecture, E is encoder consisting of layer 1 through 3, D is decoder consisting of layer 4 through 6, and C is classifier consisting of layer 7 through 9. The experiments are trained by the Keras library based on the TensorFlow framework. Traditional neural networks adopt Sigmoid activation function, so we also adopt it in Table 1. Our hidden layer has 50 neurons which is the input of D and C, and also is the output of E. In backpropagation process, we adopt adaptive moment estimation (Adam) [26] which use the default parameters. In training process, we combine E and D to obtain an autoencoder (E-D), and combine E and C to obtain a classifier (E-C). Then, we joint training E-D and E-C. The parameters of E are shared by E-D and E-C. Finally, we only use E-C to classify HSI.

Table 2 designs a BTCAE, which does not use fully connected layer (FC). Our architecture also has 3 modules which are encoder (E), decoder (D) and classifier (C). E consists of layer 1 though 6, D consists of layer 7 though 11, and C consists of layer 12 though 15. The training process is same as TBAE. Because TBCAE has the 4-th operation and the 8-th operation in Table 2 to lead to dimensional change and the dimension of Pavia University data is 103, we remove the last dimension of Pavia University to remain 102 dimensions.
Table 1. The architectures of BTAE for Pavia University data.

| No. | Operation     | Neurons size | Activation function |
|-----|---------------|--------------|---------------------|
| 1   | E-input       | $103 \times 1$ | No                  |
| E   | FC            | $50 \times 1$ | Sigmoid             |
| 3   | E-output      | $50 \times 1$ | No                  |
| 4   | D-input       | $50 \times 1$ | No                  |
| D   | FC            | $103 \times 1$ | Sigmoid             |
| 6   | D-output      | $103 \times 1$ | No                  |
| 7   | C-input       | $50 \times 1$ | No                  |
| C   | FC            | $9 \times 1$  | No                  |
| 9   | C-output      | $9 \times 1$  | Softmax             |

Table 2. The architectures of BTCAE for Pavia University data.

| No. | Operation     | Kernel size | Feature Map | Activation function |
|-----|---------------|-------------|-------------|---------------------|
| 1   | E-input       | $102 \times 1$ | 1           | No                  |
| 2   | Conv          | $3 \times 1$  | 32          | Relu                |
| 3   | Conv          | $3 \times 1$  | 32          | Relu                |
| E   | MaxPooling    | $2 \times 1$  | 32          | Relu                |
| 5   | Conv          | $3 \times 1$  | 32          | Relu                |
| 6   | E-output      | $51 \times 1$ | 32          | No                  |
| 7   | D-input       | $51 \times 1$ | 32          | No                  |
| 8   | UpSampling    | $2 \times 1$  | 32          | Relu                |
| D   | Conv          | $3 \times 1$  | 32          | No                  |
| 10  | Conv          | $3 \times 1$  | 1           | Relu                |
| 11  | D-output      | $102 \times 1$ | 1           | No                  |
| 12  | C-input       | $51 \times 1$ | 32          | No                  |
| C   | Conv          | $3 \times 1$  | 32          | Relu                |
| 14  | FC            | $9 \times 1$  | 1           | Softmax             |
| 15  | C-output      | $9 \times 1$  | 1           | No                  |

3.2.2. The architectures of BTAE and BTCAE for Indian Pines data

Table 3 is architectures of BATE for Indian Pines data, which is similar to Pavia University data. E is encoder consisting of layer 1 through 3, D is decoder consisting of layer 4 through 6, and C is classifier consisting of layer 7 through 9.

Table 4 also designs a TBCAE for Indian Pines data. E consists of layer 1 though 7, D consists of layer 8 though 12, and C consists of layer 13 though 15. We also train this model by Adam.

In training process, we combine E and D to obtain an autoencoder (E-D), and combine E and C to obtain a classifier (E-C). Then, we joint training E-D and E-C. The parameters of E is shared by E-D and E-C.
### Table 3. The architectures of BTAE for Indian Pines data.

| No. | Operation | Neurons size | Activation function |
|-----|-----------|--------------|---------------------|
| E 1 | E-input   | 200×1        | No                  |
| E 2 | FC        | 100×1        | Sigmoid             |
| E 3 | E-output  | 100×1        | No                  |
| D 4 | D-input   | 100×1        | No                  |
| D 5 | FC        | 200×1        | Sigmoid             |
| D 6 | D-output  | 200×1        | No                  |
| C 7 | C-input   | 100×1        | No                  |
| C 8 | FC        | 16×1         | Softmax             |
| C 9 | C-output  | 16×1         | No                  |

### Table 4. The architectures of BTCAE for Indian Pines data.

| No. | Operation    | Kernel size | Feature Map | Activation function |
|-----|--------------|-------------|-------------|---------------------|
| 1   | E-input      | 200×1       | 1           | No                  |
| 2   | Conv         | 3×1         | 32          | Relu                |
| 3   | Maxpooling   | 2×1         | 32          | Relu                |
| E 4 | Conv         | 3×1         | 32          | Relu                |
| E 5 | Conv         | 3×1         | 32          | Relu                |
| E 6 | Conv         | 3×1         | 32          | Relu                |
| E 7 | E-output     | 100×1       | 32          | No                  |
| D 8 | D-input      | 100×1       | 32          | No                  |
| D 9 | Conv         | 3×1         | 32          | Relu                |
| D 10| UpSampling   | 2×1         | 32          | No                  |
| C 11| Conv         | 3×1         | 1           | Relu                |
| D 12| D-output     | 200×1       | 1           | No                  |
| C 13| C-input      | 100×1       | 32          | No                  |
| C 14| FC           | 16×1        | 1           | Softmax             |
| C 15| C-output     | 16×1        | 1           | No                  |

### 3.3. Classification results and Visualization

#### 3.3.1. Classification results of University

In this experiment, to test whether TBAE works in this paper, we only train the E-C by training data to obtain a classifier (ANN), which is also an artificial neural network for classification. Meanwhile, we can test whether TBCAE works compared with CNN which also is E-C. For University Pavia data, as shown in Table 5, we set training samples of different proportions, which includes 0.2%, 0.5%, 1%, 5%. The number of training epoch is 5000 for TBAE and TBCAE. Because we joint training E-D and E-C, different combinations of training for E-D and E-C are different effective. The number of training E-D is excessive is not benefit for E-C. Ultimately, in every epoch, we randomly choose 5 samples from unlabeled samples to train E-D model in every time, and the number of choices is 10
times. Then, we randomly choose 50 samples from labeled samples to train E-C in every time, and the number of choices is 4 times.

The proposed methods are compared with CNN and ANN which only use labeled data and are E-C. Our accuracy scores are presented in Table 5 in the from of the over accuracy (OA), average accuracy (AA), and Cohen’s Kappa (K). A training-to-total sample ratio (TTR) was randomly selected for training, with the remainder used for testing, and each class has at least one sample. The results show that our methods have an advantage over CNN and ANN when training sample proportions are 0.2% and 0.5%. But when the proportion of labeled samples is 1% and 5%, BTAE and BTCAE do not much difference compared with ANN and CNN, and the improvement is not obvious. Table 5 also shows that our methods are sensitive on AA, which is better than other accuracy scores when training sample proportions are 0.2% and 0.5%.

| TTR Criterion | CNN | ANN | TBAE | TBCE |
|---------------|-----|-----|------|------|
| OA (%)        | 78.74 | 77.23 | 80.07 | 80.56 |
| 0.2% AA (%)   | 74.27 | 63.49 | 74.60 | 76.57 |
| K             | 0.71 | 0.68 | 0.73 | 0.73 |
| OA (%)        | 81.32 | 80.00 | 83.24 | 85.19 |
| 0.5% AA (%)   | 73.46 | 71.36 | 77.84 | 81.09 |
| K             | 0.75 | 0.73 | 0.77 | 0.81 |
| OA (%)        | 88.08 | 87.02 | 87.60 | 88.83 |
| 1% AA (%)     | 85.04 | 82.92 | 83.70 | 85.50 |
| K             | 0.84 | 0.83 | 0.83 | 0.85 |
| OA (%)        | 92.36 | 90.22 | 90.05 | 92.59 |
| 5% AA (%)     | 89.28 | 87.38 | 87.32 | 90.30 |
| K             | 0.89 | 0.87 | 0.87 | 0.90 |

3.3.2. Classification results of Indian Pines data

In Table 6, we use Indian Pines data to test these methods. Table 6 also adopts different proportion samples, which includes 5%, 10%, and 15%. The number of training epoch is 5000 on TBAE and TBCAE. After a lot of testing, in every epoch, we randomly choose 20 samples from unlabeled data to train E-D model in every time, and the number of choices is 10 times. Then, we randomly choose 100 samples from labeled samples to train E-C in every time, and the number of choices is 5 times. Table 6 shows that OA of TBCAE is better than CNN in three kinds of proportion of training samples. TBAE also has an advantage over ANN in different proportions. Particularly, when we select 5% training samples, the performance of TBCAE is more obvious than CNN, and TBAE also is far above ANN.

3.3.3. Effect of different size of unlabeled data

In Table 7, we use different proportion unlabeled samples to test these methods, which includes 100%, 50%, 25% and 10%. We can discover that the results of different proportion unlabeled data have no obvious change on OA, AA, and K in the experiment. When the proportion of unlabeled data is lesser, the OA, AA and K may be low. OA, AA and K decrease as the unlabeled sample decreases on BTAE for Indian Pines data and Pavia University data. But some results may be promoted on OA and AA when the number of unlabeled samples is lesser, for example, the AA of BTAE for Indian Pines is promoted by 10% unlabeled samples. The phenomenon may be that not all unlabeled samples were used to train E-D in every epoch when the number of unlabeled samples is more and the less
proportion of unlabeled samples may be used repeatedly, because the total number of updates to the parameter is fixed in all epochs.

### Table 6. Results on the Indian Pines data.

| TTR Criterion | CNN | ANN | TBAE | TBCAE |
|---------------|-----|-----|------|-------|
| OA (%)        | 75.47 | 68.91 | 73.82 | 77.86 |
| 5% AA (%)     | 61.19 | 55.99 | 62.58 | 64.32 |
| K             | 0.72  | 0.64 | 0.70 | 0.75 |
| OA (%)        | 79.86 | 74.70 | 75.56 | 80.55 |
| 10% AA (%)    | 64.42 | 70.24 | 71.75 | 65.63 |
| K             | 0.77  | 0.71 | 0.72 | 0.80 |
| OA (%)        | 82.35 | 77.20 | 78.65 | 83.86 |
| 15% AA (%)    | 67.85 | 74.50 | 77.52 | 68.03 |
| K             | 0.80  | 0.74 | 0.76 | 0.81 |

### Table 7. Effect of different size of unlabeled data.

| TTR Criteria | unlabeled-to-total sample ratio | 100% | 50% | 25% | 10% |
|--------------|---------------------------------|------|-----|-----|-----|
| Indian Pines (5%) |                                |      |     |     |     |
| BTAE         | AA (%)                         | 62.58 | 57.35 | 56.26 | 55.77 |
|              | K                              | 0.70  | 0.66 | 0.65 | 0.65 |
| OA (%)       | 77.86                           | 77.05 | 76.59 | 77.19 |
| BTCAE        | AA (%)                         | 64.32 | 61.92 | 63.85 | 68.58 |
|              | K                              | 0.75  | 0.74 | 0.74 | 0.74 |
| OA (%)       | 83.24                           | 81.60 | 81.05 | 80.26 |
| Pavia University (0.5%) |                                |      |     |     |     |
| BTAE         | AA (%)                         | 77.84 | 75.88 | 73.93 | 73.34 |
|              | K                              | 0.77  | 0.74 | 0.74 | 0.74 |
| OA (%)       | 85.19                           | 85.15 | 85.22 | 85.99 |
| BTCAE        | AA (%)                         | 81.09 | 80.75 | 82.09 | 81.35 |
|              | K                              | 0.81  | 0.80 | 0.80 | 0.82 |

### 3.3.4. Visualization of reconstructed data and the hidden layer

The visualization of reconstructed data is shown in Figure 2. With the increase of epoch, the reconstructed data is getting better and better. In the experiment of Figure 2, for Pavia University data, we selected 0.5% data as labeled data for E-C, and rest of data is unlabeled samples for E-D. (a) is the reconstructed data of BTCAE. (c) is the reconstructed data of BTAE. For Indian Pines data, 5% data was selected as labeled data, and the rest of data is unlabeled samples. (b) is the reconstructed data of BTCAE. (d) is the reconstructed data of BTAE. We can discover that the reconstructed data is getting better and better after more epochs. The effect of reconstructed data of BTCAE is better performance than BTAE.

In Figure 3, the output of hidden layer of BTAE is the output of E of BTAE. We only train E-D named AE. When completing AE training, the output of E is the top of each diagram for each category of Pavia University data. We can see that each category is similar to other categories, so the features
may not be benefit for classification. The bottom of each diagram is the output of E of BTAE for each category, which is different form other categories. Therefore, the output may be benefit for classification.

4. CONCLUSION
In this paper, we proposed a novel semi-supervised framework based on two branch autoencoder to lead with the unlabeled samples. The method has two functions which are classification (C) and decode (D). The parameters of E are shared by C and D. We can obtain the reconstructed band in D. When completing the cross training, we can obtain a better classifier which combines E and C compared with only label samples for CNN or ANN. Finally, we obtain visualization of reconstructed data and the output of hidden layer, which are visualized. The experimental results demonstrate that the proposed approach can outperform CNN and ANN in the case of a small number of samples. In the future, we can use different combinations of training to further improve the accuracy.

Figure 2. The reconstructed data after different epochs: (a) BTCAE of Pavia University data.(b) BTCAE of Indian Pines data.(c)BTAE of Pavia University data.(d) BTAE of Indian Pines data.
Figure 3. The output of hidden layer of BTAE. The data set was Pavia University data. At the top of each diagram is the output of hidden layer of AE. At the bottom of each diagram is the output of hidden layer of BTAE.
References

[1] Chang, C. I. (Ed.) 2007 Hyperspectral data exploitation: theory and applications. John Wiley & Sons.

[2] José M. Bioucas-Dias, Plaza, A., Dobigeon, N., Parente, M., & Chanussot, J. 2012 Hyperspectral unmixing overview: geometrical, statistical, and sparse regression-based approaches. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 5 2.

[3] Lee, H., & Kwon, H. (2017). Going deeper with contextual cnn for hyperspectral image classification. IEEE Trans Image Process. 26 10, 4843-4855.

[4] Li, X., Zhang, L., & You, J. (2019). Domain Transfer Learning for Hyperspectral Image Super-Resolution. Remote Sensing. 11 6, 694.

[5] Zhang, Y., Xu, M., Du, B., Wu, K., Hu, X., & Dong, Y 2018, July Multi-Priori Learning Algorithm for Hyperspectral Target Detection. In IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium (pp. 2757-2760). IEEE.

[6] Zhan, Y., Hu, D., Wang, Y., & Yu, X. 2018 Semisupervised hyperspectral image classification based on generative adversarial networks. IEEE Geoscience and Remote Sensing Letter. 15(2), 212-216.

[7] Zhou, T. B. D., & Camps-Valls, G. 2006 Semi-supervised hyperspectral image classification with graphs. IEEE International Geoscience and Remote Sensing Symposium, IGARSS’2006. IEEE.

[8] Yuan, Y., Zheng, X., & Lu, X. (2016). Discovering diverse subset for unsupervised hyperspectral band selection. IEEE Transactions on Image Processing, 26 1, 51-64.

[9] Feng, J., Yu, H., Wang, L., Cao, X., Zhang, X., & Jiao, L. 2019 Classification of hyperspectral images based on multiclass spatial-spectral generative adversarial networks. IEEE Transactions on Geoscience and Remote Sensing.

[10] Rodarmel, C., & Shan, J. 2002 Principal component analysis for hyperspectral image classification. Surveying and Land Information Science. 62 2, 115-122.

[11] Chiang, S. S., Chang, C. I., & Ginsberg, I. W 2000, July Unsupervised hyperspectral image analysis using independent component analysis. In IGARSS 2000. IEEE 2000 International Geoscience and Remote Sensing Symposium. Taking the Pulse of the Planet: The Role of Remote Sensing in Managing the Environment. Proceedings (Cat. No. 00CH37120) (Vol. 7, pp. 3136-3138). IEEE.

[12] Chen, P., Jiao, L., Liu, F., Gou, S., Zhao, J., & Zhao, Z. (2016). Dimensionality reduction of hyperspectral imagery using sparse graph learning. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 10 3, 1165-1181.

[13] Tarabalka, Y., Fauvel, M., Chanussot, J., & Benediktsson, J. A. (2010). SVM-and MRF-based method for accurate classification of hyperspectral images. IEEE Geoscience and Remote Sensing Letters. 7 4, 736-740.

[14] Huang, K., Li, S., Kang, X., & Fang, L 2016 Spectral–spatial hyperspectral image classification based on KNN. Sensing and Imaging, 17 1, 1.

[15] Hu, W., Huang, Y., Wei, L., Zhang, F., & Li, H 2015 Deep convolutional neural networks for hyperspectral image classification. Journal of Sensors.

[16] Khodadadzadeh, M., Li, J., Plaza, A., & Bioucas-Dias, J. M 2014 A subspace-based multinomial logistic regression for hyperspectral image classification. IEEE Geoscience and Remote Sensing Letters. 11 12, 2105-2109.

[17] Li, J., Bioucas-Dias, J. M., & Plaza, A. 2009 Semi-supervised hyperspectral image classification based on a Markov random field and sparse multinomial logistic regression. Geoscience and Remote Sensing Symposium, 2009 IEEE International, IGARSS 2009. IEEE.

[18] Ma, L., Ma, A., Ju, C., & Li, X 2016 Graph-based semi-supervised learning for spectral-spatial hyperspectral image classification. pattern recognition letters. 83, 133-142.

[19] Cui, B., Xie, X., Hao, S., Cui, J., & Lu, Y 2018 Semi-supervised classification of hyperspectral images based on extended label propagation and rolling guidance filtering. Remote Sensing. 10 4, 515.

[20] Wang, C., Xu, Z., Wang, S., & Zhang, H 2018 Semi-supervised classification framework of hyperspectral images based on the fusion evidence entropy. Multimedia Tools and Applications. 77 9, 10615-10633.
[21] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Bing, X., Warde-Farley, D., & Ozair, S., et al. 2014 Generative adversarial nets. *International Conference on Neural Information Processing Systems*.

[22] He, Z., Liu, H., Wang, Y., & Hu, J. 2017 Generative adversarial networks-based semi-supervised learning for hyperspectral image classification. *Remote Sensing*. 9 10, 1042.

[23] Odena, A. . 2016 . Semi-supervised learning with generative adversarial networks.

[24] Fu, Q., Yu, X., Wei, X., & Xue, Z. 2016 Semi-supervised classification of hyperspectral imagery based on stacked autoencoders (Eighth International Conference on Digital Image Processing. International Society for Optics and Photonics).

[25] Zabalza, J., Ren, J., Zheng, J., Zhao, H., & Marshall, S. 2015 Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. *Neurocomputing*, 185(C), 1-10. Zabalza, J., Ren, J., Zheng, J., Zhao, H., & Marshall, S. . (2015). Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. *Neurocomputing*, 185 C, 1-10.

[26] Ruder, S. 2016 An overview of gradient descent optimization algorithms.

**Acknowledgments**

This work was supported by the Key National Research and Development Program of China (No. 2017YFD0600906), and National Natural Science Foundation of China (No.61731022, 31872240 and 41971397).