Computing and Informatics, Vol. 39, 2020, 973–993, doi: 10.31577/cai_2020_5_973

DEEP LSTM WITH GUIDED FILTER FOR HYPERSPECTRAL IMAGE CLASSIFICATION

Yanhui Guo, Fuli Qu*, Zhenmei Yu*, Qian Yu

School of Data and Computer Science
Shandong Women's University
2399 Daxue Road, Changqing District
Jinan, China
e-mail: {guoyanhui03, qufuli23}@163.com, {zhenmei_yu, yuqian}@sdwu.edu.cn

Abstract. Hyperspectral image (HSI) classification has been a hot topic in the remote sensing community. A large number of methods have been proposed for HSI classification. However, most of them are based on the extraction of spectral feature, which leads to information loss. Moreover, they rarely consider the correlation among the spectrums. In this paper, we see spectral information as a sequential data which should be relevant to each other. We introduce long short-term memory (LSTM) model, which is a typical recurrent neural network (RNN), to deal with HSI classification. To tackle the problem of overfitting caused by limited labeled samples, regularization strategy is introduced. For unbalance in different classes, we improve LSTM by weighted cost function. Also, we employ guided filter to smooth the HSI that can greatly improve the classification accuracy. And we proposed a method for modeling hyperspectral sequential data, which is very useful for future research work. Finally, the experimental results show that our proposed method can improve the classification performance as compared to other methods in three popular hyperspectral datasets.

Keywords: Recurrent neural network, long short-term memory, guided filter, hyperspectral image classification

* Corresponding authors

1 INTRODUCTION

Remote sensors can acquire hyperspectral images, which has hundreds of spectral data channels of the same pixel. The detailed spectral information can increase the recognition of materials, so as to improve the classification accuracy of materials. HSI classification has generated considerable attention and has been widely used in various areas including land cover, environmental protection, and agriculture. However, there are still two key issues that need to be addressed, curse of dimensionality and limited number of labeled training samples.

It is generally known that the task of HSI classification is to categorize the pixels into one of several classes by their spectral features. A large number of pixel-wise classifiers have been proposed to deal with the HSI classification, including random forests [7], k-Nearest Neighbor [19], support vector machine (SVM) [20], and sparse representation [4]. However, most of these traditional methods suffer from the Hughes phenomenon [18]. To solve the aforementioned problem, feature extraction and feature selection are adopted in these methods. Generally, principal component analysis (PCA) [8] and independent component analysis (ICA) [22] are common methods in feature extraction. Band selection [9] or subspace projection techniques [1] are widely adopted in classification of spectral patterns. Although these methods have improved classification accuracy, both feature extraction and subspace projection can lead to information loss and cannot make full use of hyperspectral features.

In recent years, deep learning has made promising progresses in many fields. Deep learning based methods also are adopted in HSI classification, including the autoencoder [3, 16], convolutional neural network (CNN) [2] and deep belief network (DBN) [5]. The paper [3] proposed a deep learning framework for HSI classification. Autoencoder learns to reconstruct the input vector and reduce the dimension of spectrum. Then a multi-class logistic regression was used to classify the HSI. CNN uses extensive parameter-sharing to tackle the curse of dimensionality and also classifies hyperspectral data directly in the spectral domain. Hu et al. [17] proposed a five-layers network, which can achieve better classification performance. Chen et al. [2] proposed a regularized 3-D CNN-based feature extraction model to extract efficient spectral-spatial features for HSI classification. Additionally, spectral-spatial classification was proposed by many researchers which combines spatial context with spectral information. However, this research is beyond the scope of this paper.

Autoencoder and CNN model obtain better classification accuracy in hyperspectral image, owing to their feature representation. Yet, there is a large number of parameters to be trained in the CNN model. For a hyperspectral image with only a small number of labeled samples, the advantages of CNN model cannot be fully realized. Moreover, all the aforementioned methods view the spectrum as a vector, and can result in information loss. The spectrum of a pixel is regarded as independent of each other. A pixel is considered a point in an orderless feature space. However, hyperspectral data is seen as a continuing spectra sequences in continuous spectrum bands. Recurrent neural network is a typical deep learning model for solving sequential problems. RNN parameters are less, which is more suitable for the case of fewer training sets than CNN. So, we make use of a recurrent neural network (RNN) to characterize the sequential property for classifying the HSI.

Unlike object recognition, it is difficult to train a RNN model for reaching a steady state. We introduce guided filter to smooth the HSI, which greatly improves the HSI classification accuracy. Other filters (bilateral filter, joint bilateral filter) also can be used to smooth the noise. We adopt guided filter due to the ability to preserve edge information and gradient. The detailed steps are as follows:

1. We adopted a recurrent neural network (LSTM) for HSI classification. It is a very novel idea to regard spectral information as an ordered series. We learn the correlation between spectrums and LSTM, which is useful for HSI classification.
2. In order to improve the classification accuracy, we employ the guided filter to smooth HSI. Guided filter can denoise the image and preserve the edge of the image. So, it can greatly improve the classification accuracy.
3. Since the labeled samples are limited, deep learning model tends to overfit. To solve it, we adopt some regularization strategies, such as L2 regularization and dropout.
4. To address the problem of unbalance of the samples in different classes, we implemented a weighted cost function, which can improve the average accuracy of the classification.
5. The proposed methods are applied on two widely used hyperspectral datasets. We do compared experiments with SVM classifier and Autoencoder for three evaluation metrics.

The rest of this paper is organized as follows. Section 2 describes the related methodology and work. Section 3 gives the analysis of HSI classification and the proposed model in detail. Section 4 shows the results of the experiment, which is followed by conclusions in Section 5.

2 RELATED METHODOLOGY AND WORK

This section presents the principle of LSTM and guided filter, which is the foundation of this paper.

2.1 Recurrent Neural Networks

Recurrent neural networks (RNN) are a family of neural networks for processing sequential data, and have been successfully applied in many fields, such as natural language processing [21], speech recognition [12], image recognition [13], etc. RNN consists of an input layer, a hidden layer, and an output layer. Unlike traditional neural networks, there are links between hidden layer units. Given an input sequence

$x = (x_1, x_2, \ldots, x_T)$, a standard RNN computes the hidden vector sequence $h = (h_1, h_2, \ldots, h_T)$ and output vector sequence $y = (y_1, y_2, \ldots, y_T)$ by iterating the following equations from $t = 1$ to T:

$$h_t = g(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \tag{1}$$

$$y_t = W_{hy}h_t + b_y \tag{2}$$

where the W denotes weight matrices (e.g., W_{xh} is input-hidden weight matrix), the b denotes bias vectors (e.g., b_y is output bias vector) and g is the hidden layer function. Generally, g is a bounded function such as a logistic sigmoid function or a hyperbolic tangent function.

While RNN is focused on sequential relationship, and has made promising progresses in many fields, it still encounters difficulty in dealing with long-term sequential data since the gradients tend to vanish. An improved RNN model, named long short-term memory (LSTM) [11], is proposed for the above problem. The key to LSTM is the cell state, which takes the former state and current data as input. LSTM consists of five parts, including input gate, output gate, forget gate, cell input and cell output. The calculation process is as follows.

First, we compute the values for the input gate i_t, and the candidate value $\tilde{C}_t$ for the states of the memory cells at time t:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{3}$$

$$\tilde{C}_t = \tanh(W_{ic}x_t + U_c h_{t-1} + b_c). \tag{4}$$

Then, we compute the forget gate activation f_t at time t:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f). \tag{5}$$

Given the value of the input gate activation i_t, the forget gate activation f_t and the candidate state value $\tilde{C}_t$, we can compute the memory cells' new state C_t:

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1}. \tag{6}$$

Finally, we can compute the value of their output gates and their outputs:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o), \tag{7}$$

$$h_t = o_t * \tanh(C_t) \tag{8}$$

where x_t is the input to the memory cell layer at time t, W, U and V denote the weight matrices, b is bias vector, σ is the activation function.

2.2 Guided Filter

Guided filter[1] was proposed for the first time by He [15]. Guided filter is widely used in noise reduction, image abstraction, etc. Given a guidance I and an input image p, we can obtain an output image q by guided filter. Generally, q is a linear transform of I in a window ω_k centered at the pixel k. If the radius of k is r, the size of local window ω_k is $(2r+1) \times (2r+1)$.

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \tag{9}$$

where a_k is linear coefficient and b_k is a bias. From the model, it is obvious that $\nabla q = a\partial I$, which means that the filtering output q will have similar edge with guidance image I.

To obtain the coefficient and bias, a cost function for minimizing the term of mean error in the window ω_k is applied as follows:

$$E(a_k, b_k) = \Sigma_{i\in\omega_k}((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2). \tag{10}$$

Here, ϵ is a regularization parameter which could affect the blurring for the guided filter. According to the literal [10], formula (10) leads to a solution as follows.

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon}, \tag{11}$$

$$b_k = \bar{p}_k - a_k \mu_k \tag{12}$$

where μ_k and σ_k^2 are the mean and variance of I in ω_k, $|\omega|$ is the number of pixels in the local window, and $\bar{p}_k$ is the mean of p in the window. After obtaining the coefficient a_k, b_k, we can compute the filtering output q_i. Through the above process, we can get a linear transform image q.

3 THE ANALYSIS AND MODELING OF HSI CLASSIFICATION

3.1 The Analysis of HSI Spectrums

LSTM model is better at dealing with sequence data. We analyze the feature of HSI from two dimensions. One is whether spectral information has sequence characteristics. The other is whether the spectral information of different classes is separable. We selected three categories in the KSC dataset (Figure 1). We can see that the hyperspectral data have sequence characteristics and have different spectral characteristics between each other to classify. This is the assumption based on which we examine the proposed idea in this paper.

[1] `http://kaiminghe.com/eccv10/index.html`

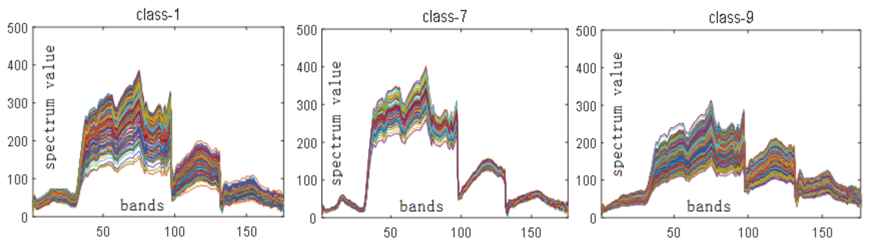


Figure 1. Spectral data of the 3 classes selected from KSC dataset with 176 channels

3.2 The Proposed Method of HSI Classification

We proposed a novel LSTM model for HSI classification with guided filter. First, we obtain a color guidance image from the original HSI by PCA method. Then, we filter the original HSI by guidance image. Finally, filtered HSI was classified by the improved LSTM model. The process is shown in Figure 2.

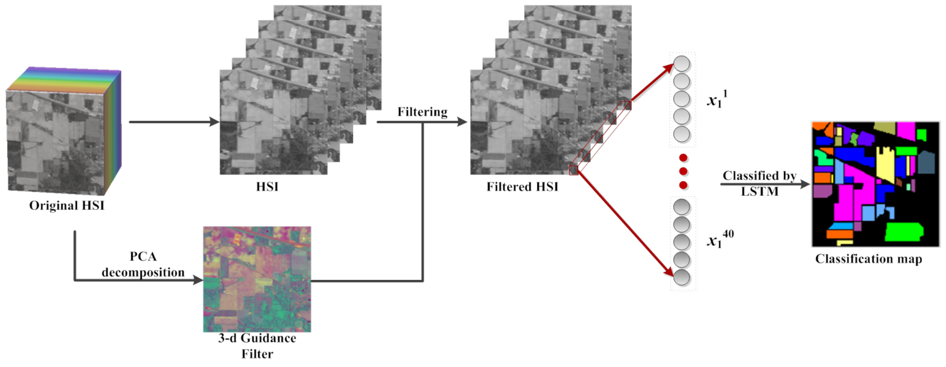


Figure 2. Framework of HSI classification by the proposed method

3.2.1 Filtering the HSI with Guided Filter

From Figure 1 (especially class-1), we can see that the regularity with spectral data is obvious. That is, materials in the same class have similar waveforms. However, there is still a lot of noise, inconsistent with the overall trend. Just like class-7, the right of the waveform is of some confusion. To solve this problem, we introduce guided filter to smooth the noises. Guided filter is an edge-preserving filter, which can be used for edge-aware smoothing. In this paper, the spectral data in Figure 1 were converted into the data in Figure 3 by guided filter. As Figure 3 shows, the spectral data is more obvious after filtering. The implementation of the method is as follows.

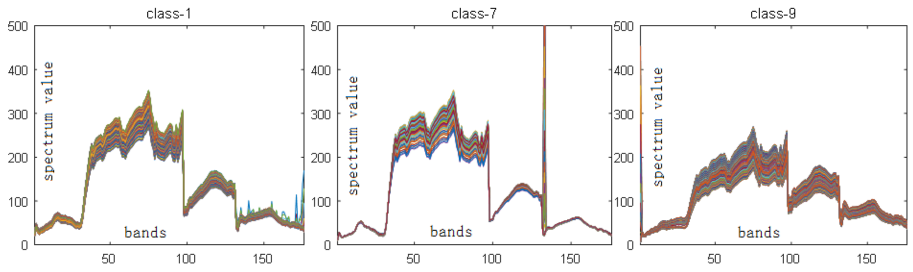


Figure 3. Spectral data of the 3 classes selected from KSC dataset with 176 channels after filtering

First, we need to obtain a guidance image by Principal Component Analysis (PCA). We take the first three principal components as a color guidance image. Given a dataset $D = \{d_1, d_2, \ldots, d_S\}$, we adopt PCA to obtain the following result. Here d_i is the information of the i^{th} band, and S denotes the number of bands.

$$[p_1, p_2, \ldots, p_S] = \text{PCA}(D). \tag{13}$$

So, the guidance image is $G = [p_1, p_2, p_3]$. Then, based on the formula (3), (4), using input image d_1 and guidance image G, we can get the filtering output u_1. By the same way, we can yield all the d_i which construct a new hyperspectral image $U = \{u_1, u_2, \ldots, u_S\}$.

3.2.2 LSTM Model for HSI Classification

The limited number of training samples makes the overfitting a serious problem. To tackle this problem, a regularization strategy based on L2 regularization and dropout is utilized. Meanwhile, HSI samples are in unbalance. The number of samples in some categories is large while in other categories it is small. Previous experiments show that the accuracy of category with the small size is worse than that with the large size. To handle this problem, we implement the weighted cost function, increasing the penalties of small sample misclassification. So, the cross-entropy cost function of LSTM network develops into the following formula:

$$\text{Loss}(T, Y) = -\sum_{i=1}^{n}\sum_{c=1}^{C} w_c * t_{ic} * \log\left(y_{ic} + \frac{\lambda}{2}\|W\|_2\right) \tag{14}$$

where n and C denote the number of samples and categories, respectively. λ is a coefficient of the regularization. We use gradient descent method to learn it. t_{ic} is a true classification label for i^{th} sample in the test set. When the i^{th} sample belongs to the class c, t_{ic} value is one, otherwise, t_{ic} value is zero. y_{ic} is a predicted value of i^{th} sample, which has the same definition with t_{ic}. w_c is the weighted term, obtained by the following formula:

$$w_c = 1 + \frac{n_{max} - n_c}{n_{max}} \times \theta \tag{15}$$

where n_{max} denotes the number of the largest class, the n_c denotes the number of c-class. θ is a parameter which needs to be learnt.

In addition to the improvements mentioned above, the LSTM model is affected by many other factors, such as normalization, modeling, and optimization function, etc. We investigated the influence of different normalization methods on the HSI classification accuracy. We adopt min-max normalization to normalize the raw data to $[0, 1]$. The normalization formula is $x_{norm} = (x - x_{min})/(x_{max} - x_{min})$, where x denotes the raw data, x_{min} and x_{max} are minimum and maximum values, respectively. Experimental results show that the normalization in the spectral data of a pixel is better than that of the spectral band.

Optimization algorithm is the heart of machine learning, which affects the convergence and optimization of the algorithm. We also found that the Adam is faster and better than the stochastic gradient descent (SGD) optimization method.

Moreover, the network structure of the LSTM model plays an important role in the HSI classification, including input nodes, steps, and hidden layer nodes. We only discuss the influence of input nodes and steps, in this section. Hidden layer nodes are discussed in Section 4. Taking Indian Pines dataset as an example, we study the influence of the number of different input nodes on the classification accuracy. Parameters of the input node and step are set by the following groups, including (2, 100), (5, 40), (8, 25), (10, 20). In which, the product of the number of input nodes and steps is equal to the numbers of bands. The overall accuracy of different modeling methods can be seen from Figure 4. Although (2, 100) looks better at the result, it takes longer training time due to having more time steps. Considering the stability and convergence, we choose (5,40) as the experimental modeling method.

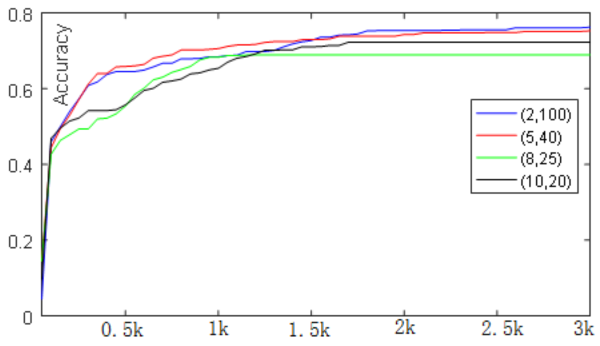


Figure 4. Accuracy of different modeling methods

4 EXPERIMENTS AND RESULTS

All our programs are implemented using Python 2 language and Tensorflow 1.0 library. We implemented all the methods on our PC with 32 GB memory and 8-core CPUs. LSTM methods and AE methods were implemented with a GTX1060 GPU for acceleration. In the following section, we first show the experimental setup, and then describe the experimental process and results in detail.

4.1 Experimental Setup

4.1.1 Datasets

Three hyperspectral data, including Indian Pines and Kennedy Space Center (KSC), are employed to evaluate the effectiveness of the proposed method. The Indian Pines dataset was gathered by Airborne Visible Infrared Imaging Spectrometer (AVIRIS) sensor over the Indian Pines test site in Northwest Indiana. This dataset consists of 145×145 pixels with 200 spectral bands in the wavelength range from 0.4 to 2.5 μm. In this scene, there are 16 categories to be classified, including woods, grass-pasture, etc.

KSC data was also collected by the AVIRIS sensor, acquired in 224 bands of 10 nm with center wavelengths from 0.4 to 2.5 μm. After removing water absorption and low SNR bands, 175 bands were used for the analysis. There are 13 categories to be classified.

Salinas scene was collected by the 224-band AVIRIS sensor over Salinas Valley, and it is characterized by high spatial resolution. It contains 512×217 pixels in all. We also discarded the 20 water absorption bands and selected 200 bands for experiments.

Detailed information of categories and samples is shown in Table 1.

4.1.2 Evaluation Metrics

To evaluate the performance of different HSI classification algorithms, we apply three widely used quality indexes, including the overall accuracy (OA), the average accuracy (AA), and the kappa coefficient (KA). OA shows the number of hyperspectral pixels that are classified correctly, divided by the number of all test samples. AA is the mean of the classification accuracies of all classes. KA is a statistical measurement of agreement, based on the confusion matrix of different classes. If we have a confusion matrix M, where m_{ij} is the element in row i, column j. OA, AA and KA also can be computed by formulas (16), (17), (18), where N and C denote the number and classes of samples.

No.	Indian Pines		KSC		Salinas	
	Categories	**smp**	**Categories**	**smp**	**Categories**	**smp**
C1	Alfalfa	46	Scrub	761	Brocoli_G_W_1	2 009
C2	Corn-N	1 428	Willow swamp	243	Brocoli_G_W_2	3 726
C3	Corn-M	830	CP hammock	256	Fallow	1 976
C4	Corn	237	CP/Oak	252	Fallow_R_P	1 394
C5	Grass-M	483	Slash pine	161	Fallow_smooth	2 678
C6	Grass-T	730	Oak/Broadleaf	229	Stubble	3 959
C7	Grass-P-M	28	Hardwood	105	Celery	3 579
C8	Hay-W	478	Graminoid	431	Grapes_untrained	11 271
C9	Oats	20	Spartina	520	Soil_V_D	6 203
C10	Soybean-N	972	Cattail marsh	404	Corn_S_G_W	3 278
C11	Soybean-M	2 455	Salt marsh	419	Lettuce_R_4wk	1 068
C12	Soybean-C	593	Mud flats	503	Lettuce_R_5wk	1 927
C13	Wheat	205	Water	927	Lettuce_R_6wk	916
C14	Woods	1 265			Lettuce_R_7wk	1 070
C15	Build-G-T-D	386			Vinyard_untrained	7 268
C16	Stone-S-T	93			Vinyard_V_T	1 807
	Total	10 249	Total	5 211	Total	54 129

Table 1. Categories and samples of three datasets

$$OA = \frac{\sum_{i=1}^{C} m_{ii}}{N}, \tag{16}$$

$$AA = \frac{\sum_{i=1}^{C} \frac{m_{ii}}{m_{ii}}}{N}, \tag{17}$$

$$KA = \frac{N\sum_{i=1}^{C} m_{ii} - \sum_{i=1}^{C} m_{i+}m_{+i}}{N^2 - \sum_{i=1}^{C} m_{i+}m_{+i}}. \tag{18}$$

4.2 Influence Factors of the Experiments

4.2.1 Analysis of Sampling Proportion

The proportion of sampling is an important factor affecting the training model. Indian Pines dataset was taken as a case. We exacted samples as training set at the ratio of 5 %, 10 %, 15 %, 20 %, 25 %, and 30 %. We tested three methods including SVM, Autoencoder, and general LSTM, on the above six cases. The experimental results are shown in Figure 5. It illustrates the changes in the accuracy over the proportion from 5 % to 30 %. The increasement of SVM and Autoencoder methods become slow gradually. However, LSTM seems to have a large increase to promote. This shows that LSTM is more dependent on sample size. This is consistent with the fact that deep learning requires a large number of training samples. In order to accommodate less labeled samples, we choose 10 % as the experimental ratio in this paper.

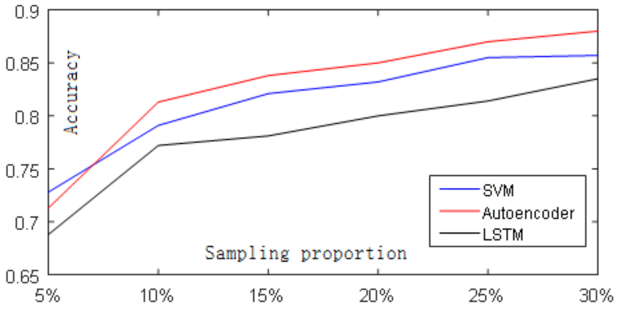


Figure 5. Analysis of sampling proportion in Indian Pines

4.2.2 Analysis on the Number of Hidden Layer Nodes and Iterations

How many iterations does the LSTM model need to reach a stable state? What is the number of hidden nodes to achieve the best classification accuracy? Experiments are done on three datasets. The number of hidden layer nodes are set by 100, 150, 200, 250, and 300, respectively. The results of LSTM with different hidden nodes on three datasets are shown in Figure 6. We can see from the chart, when the number of nodes is 200, the accuracy is the best on Indian Pines dataset. For Kennedy Space Center (KSC) dataset, the number of hidden nodes is 150 for the best result. Salinas dataset choose 200 hidden nodes for our experiment, which is not affected by the number of hidden nodes.

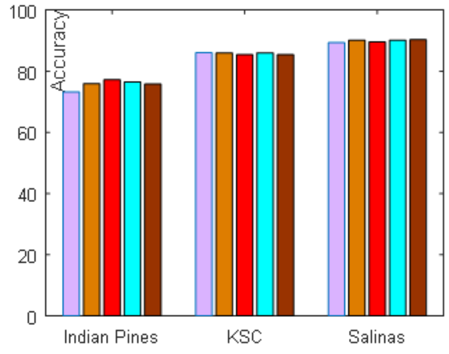


Figure 6. Accuracy of LSTM with different nodes on three datasets

The results of LSTM with different hidden nodes on Indian Pines are shown in Figure 7. It is clear that the accuracy grows sharply for the first one thousand iterations. They get stable after three thousand iterations. So, we set the number of iterations to 3×103 in the following experiments. Surprisingly, we found that the less nodes the hidden layer has, the slower the initial growth rate is, which is different from our expectations. Another finding is that the network structure with fewer nodes is more prone to underfitting and is more unstable than that with more nodes.

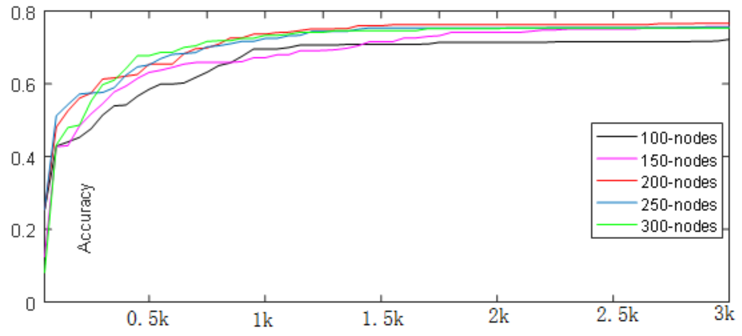


Figure 7. Accuracy of LSTM with different nodes on three datasets

4.3 Experimental Results

To examine the effectiveness of our proposed methods, we do experiments on three widely used datasets. Ten percent of the dataset is taken as training set, and the remaining 90 % as the test set. In this section, the proposed methods are compared with two widely used classification methods, SVM [20] and autoencoder [3], which are typical examples of traditional methods and deep learning methods, respectively.

4.3.1 Experiment on Indian Pines Dataset

Parameter Settings. In this experiment, we use libSVM[2] designed by Lin [6]. The libSVM has two mainly parameters C and g to be set. The C and g are determined by cross validation. C changes from 10^{-2} to 10^4, g ranges from 2^{-1} to 2^4. The parameters of SVM are set by the same way in the following two experiments.

The Autoencoder [3] is based on radius r and regularization parameter ϵ. Radius r is used to express the range of smooth. ϵ is used to control the degree of smooth. We set $r = 3$ and $\epsilon = 0.001$ in this paper. It has the same settings on next two experiments. So, we do not represent them anymore in the following sections.

LSTM, RG-LSTM and GF-LSTM has the same network structure in our experiments. They have three layers, including input layer, hidden layer and output layer,

[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
[3] `https://github.com/hantek/deeplearn_hsi`

which are set to 5, 200, and 16, respectively. The step of LSTM is 40. The number of training is set to 5×10^3. RG-LSTM and GF-LSTM have two extra parameters. They are weight coefficient θ and regularization coefficient λ, which are set to 10^{-2} and 10^{-3}, respectively. Besides, radius r and regularization parameter ϵ of guided filter are two important parameters for GF-LSTM. In our previous work [14], the performance of methods is the best when radius r is set to 3 on Indian Pines. In this experiment, we set $r = 3$ and $\epsilon = 10^{-3}$.

Experimental Results. To evaluate our methods, we compared the performance of the methods using the quantitative index of OA, AA and KA. The detailed performance of these methods is shown in Table 2. It can be seen from Table 2 that the OA results of SVM and Autoencoder are better than LSTM by 4 %. LSTM with regularization and weight (RG-LSTM) has improved the classification accuracy of LSTM, which is better than SVM and Autoencoder. Obviously, LSTM with guided filter (GF-LSTM) outperforms all the above methods on all the indexes. Especially in the OA index, GF-LSTM reaches 90.15 %, which is 10 % higher than other methods. Also, overall classification maps of different methods are illustrated in Figure 8. It can be seen from this figure that the proposed methods (RG-LSTM, GF-LSTM) achieve better classification performance than other compared approaches. Besides, the classification results of C9 are poor, especially with Autoencoder. That is because the sample size of C9 is only 20. This explains that sample size has a greater impact on Autoencoder.

For all the methods, the value of OA is higher than that of AA, which means that the class with large samples has a better performance than that with small samples. This problem is more serious in the LSTM. Compared the LSTM, RG-LSTM and GF-LSTM are improved by regularization strategy, which has partly solved the problem of unbalance.

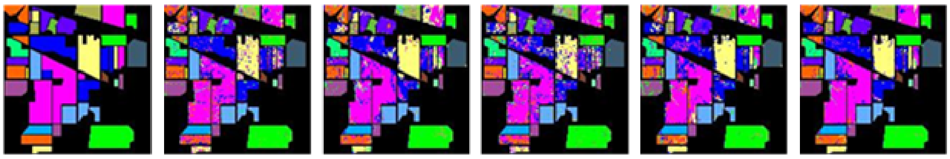


Figure 8. Qualitative results on Indian Pines dataset

4.3.2 Experiment on KSC Dataset

Parameter Settings. For this dataset, there are 4 layers for Autoencoder including two hidden layers. Each hidden layer has 60 nodes. The iteration of pre-training is set to 33×102. Compared to Indian Pines, the hidden layer is reduced, and the iteration is increased. That is because there are too few samples of KSC

	SVM	**Autoencoder**	**LSTM**	**RG-LSTM**	**GF-LSTM**
C1	83.33	70.52	60.71	76.92	91.30
C2	76.34	74.50	66.53	70.44	84.95
C3	71.73	70.17	73.31	78.14	86.84
C4	48.68	53.77	47.49	50.90	64.63
C5	87.38	85.05	82.34	79.86	90.61
C6	95.06	95.50	89.45	93.05	99.23
C7	86.66	90.00	80.00	68.75	74.71
C8	99.02	95.62	98.75	99.74	100.00
C9	52.63	30.56	52.63	48.57	51.67
C10	75.38	79.41	67.71	74.01	86.00
C11	83.97	69.16	77.56	89.75	93.00
C12	71.12	87.36	69.23	77.33	76.96
C13	93.75	93.07	87.66	88.44	95.59
C14	94.26	94.83	95.22	90.95	98.36
C15	59.77	88.73	55.58	73.23	85.24
C16	86.79	77.48	73.02	91.67	88.00
OA	81.01	81.48	77.02	81.76	90.15
AA	79.12	78.94	74.46	77.99	87.19
KA	78.29	77.29	74.20	79.17	88.24

Table 2. Classification accuracy of methods on the Indian Pines dataset (%)

dataset. Other parameters of Autoencoder are the same as the previous experiment.

For LSTM model, we set three layers to train, input layer, hidden layer and output layer. The size of hidden layer is 150, and the input size is 5. The size of time step in the KSC dataset is 35. That is, how many times it carries out the forecast. For the RG-LSTM and GF-LSTM, we set coefficient of the regularization $\lambda = 10^{-4}$, and weight coefficient $\theta = 10^{-2}$. The other settings are the same as the previous experiment.

Experimental Results. The KSC dataset only has about 5 thousand samples. This makes the results different from the previous experiment. The qualitative results are indicated in Figure 9. The results of different methods are shown in Table 3. It can be seen from Table 3, comparing the OA results of SVM, Autoencoder and LSTM, SVM is the best, followed by LSTM, and Autoencoder is the worst, which is 10 % worse than the other two methods. It can be attributed to the small sample size, which makes Autoencoder not fully trained. However, SVM is more suitable for small sample classification. The improved LSTM (RG-LSTM, GF-LSTM) is better than the first three method at OA, AA, and KA. In particular, OA of GF-LSTM increases by 10 % compared with other methods. All these prove that our proposed methods are very effective for HSI classification.

Comparing OA and AA for all the methods, we can see that the gap of Autoencoder is the biggest, which means that Autoencoder is greatly affected by the sample size and sample imbalance. LSTM is less affected by sample imbalance than SVM and Autoencoder. Let us take a look at the classification results of C7, which has the least samples. Autoencoder only has an accuracy about 10.47 %, and LSTM has an accuracy about 57.94 %. With regluarization and weighted cost function, the accuracy of C7 upgrades to 78.79 % by RG-LSTM. Then, with the guided filter, the accuracy is up to 100 % by GF-LSTM, which effectively solves the small size and imbalance of the samples.

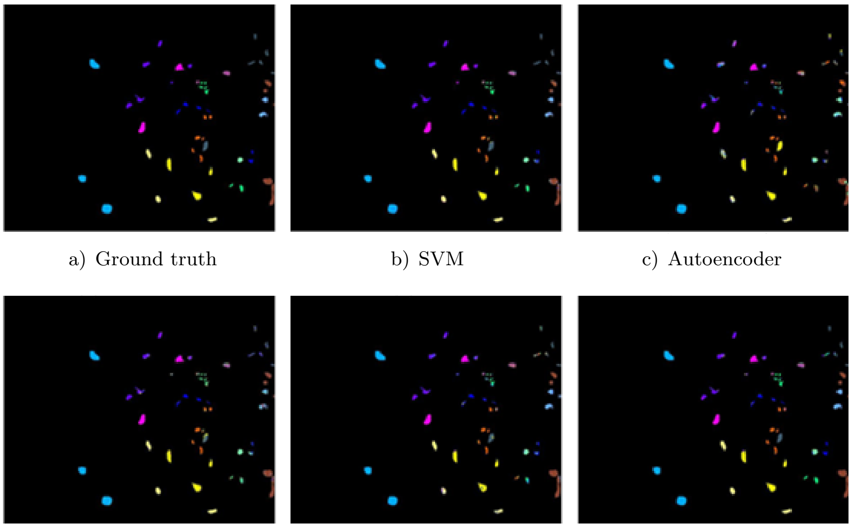


Figure 9. Qualitative results on KSC dataset

4.3.3 Experiment on Salina Dataset

Parameter Settings. For this dataset, there are still 6 layers for Autoencoder as Indian Pines has. Each hidden layer has 60 nodes. Other parameters of Autoencoder are the same as the previous experiment in Indian Pines dataset.

For LSTM model, we set three layers to train, input layer, hidden layer and output layer. The size of hidden layer is 200, and the input size is 5. The size of time step is 40. That is, how many times it carries out the forecast. We set coefficient of the regularization $\lambda = 10^{-4}$, and weight item $\theta = -1 \times 10^{-2}$.

Experimental Results. The last experiment is performed on the Salinas dataset, which is the biggest one we have chosen. The qualitative results are shown in

	SVM	**Autoencoder**	**LSTM**	**RG-LSTM**	**GF-LSTM**
C1	95.59	89.49	90.43	100.00	100.00
C2	73.39	72.94	74.76	71.30	87.20
C3	92.23	27.15	80.08	78.91	98.48
C4	65.47	21.12	61.85	73.10	82.72
C5	76.24	43.38	66.67	72.85	84.03
C6	58.76	24.60	61.39	75.01	97.41
C7	64.58	10.47	57.94	78.79	100.00
C8	79.30	61.92	83.62	90.34	95.58
C9	84.01	78.40	87.91	98.01	100.00
C10	96.87	76.35	83.25	84.87	89.69
C11	85.45	98.90	93.28	94.92	96.41
C12	92.09	58.88	87.78	87.33	90.53
C13	100.0	99.75	98.86	95.87	99.66
OA	87.38	71.86	86.06	88.46	95.48
AA	81.84	58.72	84.37	85.12	93.98
KA	85.87	68.51	79.06	87.08	94.94

Table 3. Classification accuracy of methods on the KSC dataset (%)

Figure 10. It is apparent from this figure that the GF-LSTM obtains the best results, which has the fewest noise points. The detailed results are illustrated in Table 4. All the methods perform well on this dataset. The worst is about 87.4 % by Autoencoder. The OA results of LSTM, RG-LSTM, GF-LSTM are all over 90 %, especially GF-LSTM reaches 98.15 %, outperforming other methods greatly. It can be seen that LSTM with guided filter can significantly improve the classification accuracy.

Comparing AA and OA, we see interesting phenomena that the results of AA are higher than those of OA, which is different from the previous two experiments. This means that when the sample size reaches a certain extent, increasing the sample does not improve the classification accuracy. The gap between AA and OA for Autoencoder is bigger than with other methods. Maybe the number of samples is the main influence factor of Autoencoder. LSTM has better robustness than SVM and Autoencoder. LSTM with regularization (RG-LSTM) can improve the imbalance to some extent. Then, LSTM with guided filter, the results are further improved, whose OA and AA are consistent. Take C8 for example, which is the biggest category, the accuracy is 79.47 % by LSTM. However, the accuracy is up to 88.25 % by RG-LSTM (with regluarization and weighted cost function), and the accuracy reach the highest 97.75 % by GF-LSTM (with guided filter).

From the three experiments we can reach the conclusion that Autoencoder is greatly affected by the sample size. It cannot obtain good results if the size is too large or too small. SVM and LSTM have better robustness. LSTM performs better than SVM on big dataset. RG-LSTM solves the problem of small size and unbalance of samples to a certain extent. GF-LSTM outperforms all the methods at all

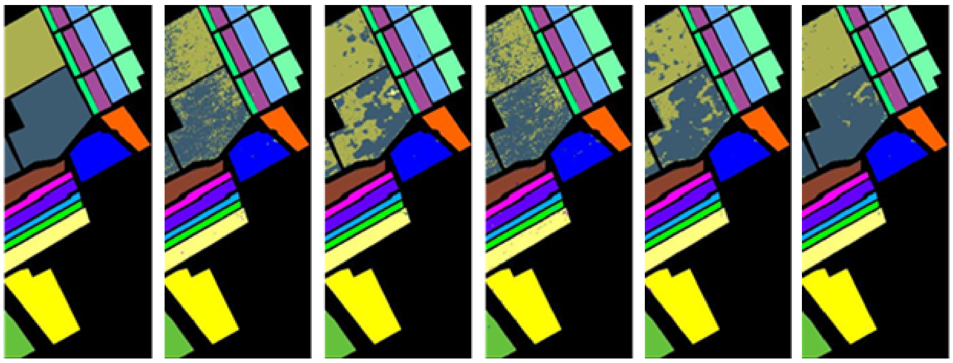


Figure 10. Qualitative results on Salinas dataset

the datasets for all indexes. So, GF-LSTM is an effective way to HSI classification.

	SVM	**Autoencoder**	**LSTM**	**RG-LSTM**	**GF-LSTM**
C1	99.94	99.27	99.34	100.00	100.00
C2	99.24	99.75	99.17	100.00	99.85
C3	92.16	95.36	98.49	98.34	99.70
C4	97.22	99.58	99.15	97.77	99.33
C5	98.62	94.67	99.44	99.53	99.32
C6	100.0	99.97	99.86	99.97	99.86
C7	98.93	99.80	99.14	99.20	99.23
C8	80.13	97.13	79.47	88.25	97.75
C9	99.54	99.94	98.95	98.98	99.91
C10	84.39	94.77	96.36	98.67	99.01
C11	85.38	92.03	96.29	97.32	99.59
C12	96.35	100.00	98.45	99.09	99.94
C13	94.97	99.87	95.24	95.00	100.00
C14	94.20	95.63	91.28	96.68	96.31
C15	63.83	18.38	70.67	75.76	93.37
C16	96.15	97.61	94.80	95.38	96.45
OA	88.09	87.40	90.30	91.88	98.15
AA	92.57	92.74	94.76	96.23	98.72
KA	86.65	85.85	89.09	90.89	97.93

Table 4. Classification accuracy of methods on the Salinas dataset (%)

4.3.4 Analysis of Time Cost

In order to compare time cost with the same platform, we reproduced SVM and Autoencoder methods for three datasets. We take the average running time of five times as the time cost, and the results are shown in Table 5. We can see that the running time of LSTM and its improved models are more expensive than SVM and Autoencoder on Indian Pines. For other datasets, LSTM and its improved models are better than SVM on running time cost. Although time cost of Autoencoder is the best, the classification accuracy is the worst. SVM is more sensitive to the size of data, and its running cost will rise rapidly when the amount of data increases. Autoencoder, LSTM, RG-LSTM and GF-LSTM are not particularly sensitive to data size, which is mainly affected by the number of model parameters.

	SVM	**Autoencoder**	**LSTM**	**RG-LSTM**	**GF-LSTM**
Indian Pines	4.011	0.234	4.4548	4.4098	4.4706
KSC	28.507	1.517	1.5554	1.5748	1.5868
Salinas	38.464	0.928	22.5939	23.6033	23.2382

Table 5. Time cost of methods on three datasets

5 CONCLUSION

We proposed a guided filter based LSTM model for HSI classification, under the assumption that the spectral data can be regarded as an ordered sequence. To tackle the problem of unbalance and limited labeled samples, we introduce weighted cost function and regularization strategy. Compared with SVM classifier and Autoencoder model, the proposed model could achieve higher accuracy at all the experimental datasets, with a 10 % samples to train.

Our work is an exploration of using LSTM for HSI classification. The modeling of HSI classification by LSTM can be a useful reference for further research. It is noteworthy that we only take into account the spectral data in this paper, therefore, some spatial-spectral techniques can be employed to improve the LSTM model for classification in the future.

Acknowledgements

This research was funded by the Science and Technology Project for the Universities of Shandong Province (grant No. J18KB171), Open Research Fund of Shandong Provincial Key Laboratory of Infectious Disease Control and Prevention, Shandong Center for Disease Control and Prevention (grant No. 2017KEYLAB01), Shandong Women's University High Level Scientific Research Project Cultivation Fund (grants No. 2020GSPGJ08 and 2019GSPGJ07), Discipline Talent Team Cultivation Program of Shandong Women's University (grant No. 1904).

REFERENCES

[1] Bioucas-Dias, J. M.—Nascimento, J. M. P.: Hyperspectral Subspace Identification. IEEE Transactions on Geoscience and Remote Sensing, Vol. 46, 2008, No. 8, pp. 2435–2445, doi: 10.1109/tgrs.2008.918089.

[2] Chen, Y.—Jiang, H.—Li, C.—Jia, X.—Ghamisi, P.: Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. IEEE Transactions on Geoscience and Remote Sensing, Vol. 54, 2016, No. 10, pp. 6232–6251, doi: 10.1109/tgrs.2016.2584107.

[3] Chen, Y.—Lin, Z.—Zhao, X.—Wang, G.—Gu, Y.: Deep Learning-Based Classification of Hyperspectral Data. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Vol. 7, 2014, No. 6, pp. 2094–2107, doi: 10.1109/jstars.2014.2329330.

[4] Chen, Y.—Nasrabadi, N. M.—Tran, T. D.: Hyperspectral Image Classification via Kernel Sparse Representation. IEEE Transactions on Geoscience and Remote Sensing, Vol. 51, 2013, No. 1, pp. 217–231, doi: 10.1109/TGRS.2012.2201730.

[5] Chen, Y.—Zhao, X.—Jia, X.: Spectral-Spatial Classification of Hyperspectral Data Based on Deep Belief Network. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Vol. 8, 2015, No. 6, pp. 2381–2392, doi: 10.1109/TGRS.2012.2201730.

[6] Chang, C. C.—Lin, C. J.: LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology, Vol. 2, 2011, No. 3, Art. No. 27, doi: 10.1145/1961189.1961199.

[7] Dalponte, M.—Ørka, H. O.—Gobakken, T.—Gianelle, D.—Næsset, E.: Tree Species Classification in Boreal Forests with Hyperspectral Data. IEEE Transactions on Geoscience and Remote Sensing, Vol. 51, 2013, No. 5, pp. 2632–2645, doi: 10.1109/TGRS.2012.2216272.

[8] Fauvel, M.—Chanussot, J.—Benediktsson, J. A.: Kernel Principal Component Analysis for the Classification of Hyperspectral Remote Sensing Data over Urban Areas. EURASIP Journal on Advances in Signal Processing, Vol. 1, 2009, Art. No. 783194, doi: 10.1155/2009/783194.

[9] Feng, J.—Jiao, L. C.—Zhang, X.—Sun, T.: Hyperspectral Band Selection Based on Trivariate Mutual Information and Clonal Selection. IEEE Transactions on Geoscience and Remote Sensing, Vol. 52, 2014, No. 7, pp. 4092–4105, doi: 10.1109/tgrs.2013.2279591.

[10] Hastie, T.—Tibshirani, R.—Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Vol. 1, Springer Series in Statistics, New York, 2001.

[11] Graves, A.: Supervised Sequence Labelling. Supervised Sequence Labelling with Recurrent Neural Networks. Springer, Berlin, Heidelberg, Studies in Computational Intelligence, Vol. 385, 2012, pp. 5–13, doi: 10.1007/978-3-642-24797-2_2.

[12] Graves, A.—Mohamed, A.-r.—Hinton, G.: Speech Recognition with Deep Recurrent Neural Networks. 2013 IEEE International Conference on Acoustics,

Speech and Signal Processing, Vancouver, BC, Canada, 2013, pp. 6645–6649, doi: 10.1109/icassp.2013.6638947.

[13] Gregor, K.—Danihelka, I.—Graves, A.—Rezende, D. J.—Wierstra, D.: DRAW: A Recurrent Neural Network for Image Generation. arXiv preprint arXiv:1502.04623, 2015.

[14] Guo, Y.—Cao, H.—Han, S.—Sun, Y.—Bai, Y.: Spectral-Spatial Hyperspectral Image Classification with k-Nearest Neighbor and Guided Filter. IEEE Access, Vol. 6, 2018, pp. 18582–18591, doi: 10.1109/access.2018.2820043.

[15] He, K.—Sun, J.—Tang, X.: Guided Image Filtering. In: Daniilidis, K., Maragos, P., Paragios, N. (Eds.): Computer Vision – ECCV 2010. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6311, 2010, pp. 1–14, doi: 10.1007/978-3-642-15549-9_1.

[16] Hinton, G. E.—Salakhutdinov, R. R.: Reducing the Dimensionality of Data with Neural Networks. Science, Vol. 313, 2006, No. 5786, pp. 504–507, doi: 10.1126/science.1127647.

[17] Hu, W.—Huang, Y.—Wei, L.—Zhang, F.—Li, H.: Deep Convolutional Neural Networks for Hyperspectral Image Classification. Journal of Sensors, Vol. 2015, 2015, Art. No. 258619, doi: 10.1155/2015/258619.

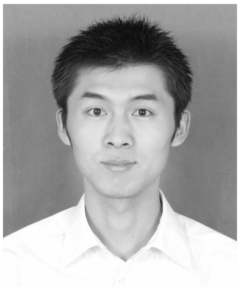
[18] Hughes, G.: On the Mean Accuracy of Statistical Pattern Recognizers. IEEE Transactions on Information Theory, Vol. 14, 1968, No. 1, pp. 55–63, doi: 10.1109/tit.1968.1054102.

[19] Ma, L.—Crawford, M. M.—Tian, J.: Local Manifold Learning-Based k-Nearest-Neighbor for Hyperspectral Image Classification. IEEE Transactions on Geoscience and Remote Sensing, Vol. 48, 2010, No. 11, pp. 4099–4109, doi: 10.1109/tgrs.2010.2055876.

[20] Melgani, F.—Bruzzone, L.: Classification of Hyperspectral Remote Sensing Images with Support Vector Machines. IEEE Transactions on Geoscience and Remote Sensing, Vol. 42, 2004, No. 8, pp. 1778–1790, doi: 10.1109/TGRS.2004.831865.

[21] Mikolov, T.—Karafiát, M.—Burget, L.—Černocký, J.—Khudanpur, S.: Recurrent Neural Network Based Language Model. Eleventh Annual Conference of the International Speech Communication Association (INTERSPEECH 2010), Makuhari, Chiba, Japan, 2010, pp. 1045–1048.

[22] Wang, J.—Chang, C.-I.: Independent Component Analysis-Based Dimensionality Reduction with Applications in Hyperspectral Image Analysis. IEEE Transactions on Geoscience and Remote Sensing, Vol. 44, 2006, No. 6, pp. 1586–1600, doi: 10.1109/tgrs.2005.863297.



Yanhui Guo received his B.Sc. degree in information management and information system from the Xi'an University of Finance and Economics, Xi'an, China in 2006, and his M.Sc. degree in computer software and theory from the Shaanxi Normal University, Xi'an, China in 2009, then he received his Ph.D. degree with the School of Computer Science of Shaanxi Normal University in 2020. Since 2009, he has been with the School of Information Technology, Shandong Women's University, Jinan, China, where he is currently Professor. His research interests include computer vision and machine learning.

Fuli Qu received her Master's degree in computational mathematics at Shandong University. She is Assistant Professor at the Institute of Data Science and Computing. Her research focuses on numerical methods of differential equations. She published her thesis at applied mathematics and mechanics.

Zhenmei Yu received her M.Sc. degree from the School of Management Science and Engineering at Shandong Normal University. She is now Professor at the School of Data and Computer Science, Shandong Women's University. Her main research interests include machine learning and artificial intelligence.

Qian Yu is currently Associate Professor in the School of Data and Computer Science, Shandong Women's University, China. She received her Ph.D. degree from the Department of Computer Science of Nanjing University in 2020. Her research interests include computer vision and medical image analysis.