remote sensing

MDPI

Article

Self-Attention-Based Conditional Variational Auto-Encoder Generative Adversarial Networks for Hyperspectral Classification

Zhitao Chen [1,2], Lei Tong [1,2,*], Bin Qian [3], Jing Yu [1] and Chuangbai Xiao [1]

[1] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; gid_chen@emails.bjut.edu.cn (Z.C.); jing.yu@bjut.edu.cn (J.Y.); cbxiao@bjut.edu.cn (C.X.)
[2] Engineering Research Center of Intelligent Perception and Autonomous Control, Ministry of Education, Beijing 100124, China
[3] Traffic Management Research Institute of the Ministry of Public Security, Wuxi 214151, China; 311062198@njust.edu.cn
* Correspondence: lei_tong@bjut.edu.cn

Abstract: Hyperspectral classification is an important technique for remote sensing image analysis. For the current classification methods, limited training data affect the classification results. Recently, Conditional Variational Autoencoder Generative Adversarial Network (CVAEGAN) has been used to generate virtual samples to augment the training data, which could improve the classification performance. To further improve the classification performance, based on the CVAEGAN, we propose a Self-Attention-Based Conditional Variational Autoencoder Generative Adversarial Network (SACVAEGAN). Compared with CVAEGAN, we first use random latent vectors to obtain more enhanced virtual samples, which can improve the generalization performance. Then, we introduce the self-attention mechanism into our model to force the training process to pay more attention to global information, which can achieve better classification accuracy. Moreover, we explore model stability by incorporating the WGAN-GP loss function into our model to reduce the mode collapse probability. Experiments on three data sets and a comparison of the state-of-art methods show that SACVAEGAN has great advantages in accuracy compared with state-of-the-art HSI classification methods.

Keywords: Generative Adversarial Network (GAN); hyperspectral classification; self-attention

Citation: Chen, Z.; Tong, L.; Qian, B.; Yu, J.; Xiao, C. Self-Attention-Based Conditional Variational Auto-Encoder Generative Adversarial Networks for Hyperspectral Classification. *Remote Sens.* **2021**, *13*, 3316. https://doi.org/10.3390/rs13163316

Academic Editors: Ajmal Mian, Jun Zhou, Naveed Akhtar, Pedram Ghamisi, Antonio Robles-Kelly and Tat-Jun Chin

Received: 24 June 2021
Accepted: 18 August 2021
Published: 21 August 2021

1. Introduction

With the development of remote sensing technology, hyperspectral images (HSI) have made great breakthroughs in earth observation. Different from three-channel color images, HSI can simultaneously collect images in hundreds of spectral bands, providing rich spectral information. Therefore, HSI are widely used in many fields [1–6], such as satellite remote sensing, agriculture observation, and mineral exploration.

For images classification problems, many algorithms [7–10] have be used, such as kernel extreme learning machine (KELM) and kernel principal component (KPCK) [7], wavelet coefficients [8], and neural network [9]. Many traditional methods have also been used to solve HSI classification problems, such as support vector machine (SVM) [11–13], K nearest neighbor (KNN) [14,15], maximum likelihood [16], neural network [17], and logistic regression [18,19]. However, these algorithms usually require manual designing of spectral and spatial features. Therefore, the classification results may be affected by manual features instead of algorithms. Moreover, traditional HSI classification algorithms usually encounter the Huges phenomenon [20], which severely affects the classification results.

Recently, deep learning methods [21–31] have received attention in HSI classification. Chen et al. [21] first applied the convolution network to solve the HSI classification problem. To augment the training data, CNN-PPF [26] was proposed by pairing selected samples.

In this way, the training data can be expanded to obtain better classification results. In Reference [31], a two-channel neural network was proposed, which extracts features from spectral and spatial domains. In Reference [30], a novel local spatial sequence method based on recurrent neural network (RNN) was proposed, which can extract local information and semantic information for HSI classification. On the other hand, Hamida et al. [22] proposed a 3D convolution network to extract the spatial-spectral features of hyperspectral data for the classification and has achieved excellent results. However, most CNN-based HSI classification methods use fixed-size patches to train the model, which cannot reflect the complex spatial structure information in HSI. Therefore, MS-DenseNet [28] uses multi-scale patches to solve this problem.

Although deep learning methods have made great progress in HSI classification, they still face the problem of limited labeled data. To solve the problem, some scholars have proposed Generative Adversarial Networks (GAN) [32] and other models [33–40] to generate virtual samples to augment the training data. Zhu et al. [36] proposed a GAN-based [32] classification method for HSI and discussed its practicability and effectiveness in HSI classification. This method generates virtual samples through the GAN [32] network to alleviate the lack of labeled data in HSI and to improve the accuracy of the model. Multi-class spatial-spectral GAN (MSGAN) [34] was also proposed; it generates samples of spatial and spectral information by designing two generators and then uses a discriminator to extract spatial-spectral features and to output the final category. DropBlock structure (DBGAN) [41] was also applied to a GAN [40], which allowed the CNN to learn features by discarding continuous regions in the feature map to improve performance. Progressive growing GAN (PG-GAN) [42] and Wasserstein GAN gradient penalty (WGAN-GP) [43] were also proposed; they gradually increase the depth of the network, making the training process faster. Capsule network (CapsNet) [44] and convolutional long short-term memory (ConvLSTM) [45] were combined in Reference [39]. It designed a new discriminator that extracts low-level features and combines them with local spatial sequence information to form high-level context features. CVAEGAN [37] was also proposed for hyperspectral classification; it makes the classification process more stable and obtains better performance.

In this paper, we propose a new HSI classification model called SACVAEGAN. Compared with the original CVAEGAN [37] model, our proposed SACVAEGAN can generate virtual samples from the generated and random latent vector and can generate more high-quality training data. We also apply the self-attention mechanism [46] to the VAE and discriminator so that the model focuses more on global features rather than local features, thus making the virtual samples generated by the model more realistic. In the discriminator, the loss function in WGAN-GP [43] is applied to the optimization process, which makes the model training more stable. Moreover, in the encoder and classifier, we extract the features of HSI data from the spectral and spatial dimensions, which can effectively enhance the performance.

The rest of this paper is organized as follows. Section 2 presents the CVAEGAN [47] model and self-attention method [46]. Section 3 presents each module of SACVAEGAN in detail. Section 4 carries out a comprehensive evaluation of the proposed method. Section 5 analyzes each module of the self-attention mechanism, the WGAN-GP loss function, and the virtual samples. Finally, conclusions are drawn in Section 6.

2. Related Work

In this section, the CVAEGAN [47] and the self-attention mechanism [46,48] are introduced, which is the basis of our proposed method.

2.1. Conditional Variational Auto-Encoder Generative Adversarial Networks

In recent years, generative models have been roughly divided into VAE and GAN. Both have their strengths and weaknesses. The image generated by the VAE is normal but blurry. The GAN generates clear images, but the GAN model is hard to converge in the

training stage and the samples generated from GAN are far from natural. The CVAEGAN network combines the advantages of the two to improve the effect of generating images.

CVAEGAN [47] is a network structure that combines VAE [49] and GAN [32]. It consists of four parts: (1) an encoder network E, used to learn the relationship between a latent vector space and the real image space; (2) the generator network G, which generates the corresponding virtual sample through the given latent vector; (3) the discriminator network D, used to judge whether a given sample is a real sample or a virtual sample; and (4) the classifier network C, used to classify a given sample.

The encoder network E transfers the data x to a latent vector space with a Gaussian distribution $N(0,1)$ through the learned distribution. It outputs the mean value μ and covariance ϵ of the latent vector corresponding to the data sample x. Then, through $z = \mu + r * exp(\epsilon)$, $r \in N(0,1)$, the latent vector z can be obtained. At the same time, the distance between $p(z)$ and $q(z|x)$ can be reduced through KL divergence, where p represents the true latent vector distribution and q represents the latent vector distribution predicted by the model. The following is the equation:

$$KL(p||q) = \sum(p(x)log(\frac{p(x)}{q(x)})) \tag{1}$$

$$L_{KL} = \frac{1}{2}(\mu^T\mu + \sum(\exp(\epsilon) - \epsilon - 1)) \tag{2}$$

The classifier takes the data x as input and outputs the posterior probability $P(c|x)$. Classifier C minimizes the following loss function as follows:

$$L_C = -E_{x\sim P_{data}(x)}[logP(c|x)] \tag{3}$$

Suppose the generator G generates data samples $G(z)$ corresponding to the latent vector by receiving the latent vector z. The discriminator is a binary classifier to judge the true or false of a given data sample. The loss function of the discriminator is as follows:

$$L_D = -E_{x\sim P_{data}(x)}[logD(x)] - E_{x\sim P_z(z)}[log(1 - D(G(z,c)))] \tag{4}$$

In the generator, to solve the problem of unstable training and to synthesize more realistic data samples, CVAEGAN [47] adds a feature matching method to the generator. The generator G minimizes the following loss function:

$$L_G = \frac{1}{2}(\|x - G(z,c)\|_2^2 + \|f_D(x) - f_D(G(z,c))\|_2^2 + \|f_C(x) - f_C(G(z,c))\|_2^2) \tag{5}$$

$$L_{GD} = \frac{1}{2}(\|E_{x\sim P_r}f_D(x) - E_{z_r\sim P_{z_r}}f_D(G(z_r,c))\|_2^2 \tag{6}$$

$$L_{GC} = \frac{1}{2}(\|E_{x\sim P_r}f_C(x) - E_{z_r\sim P_{z_r}}f_C(G(z_r,c))\|_2^2 \tag{7}$$

where f_D and f_C represent the middle layer features of the discriminator and classifier, respectively; $G(z,c)$ represents the sample generated by the latent vector generated by the encoder; and $G(z_r,c)$ represents the sample generated by the latent vector generated randomly.

2.2. Self-Attention

The purpose of the self-attention [46,48] mechanism is to improve the classification performance of the classifier. Most GAN-based methods [32] used self-attention in their encoder or generator to enhance the performance. Figure 1 shows the structure of self-attention. $f(x)$, $g(x)$, and $h(x)$ are all ordinary 1×1 convolutions, and the difference lies in the different output channels. The output of $f(x)$ is transposed and multiplied by the output of $g(x)$, and the attention map is obtained after the softmax activation function.

After multiplying the obtained attention map by the output of $h(x)$ pixel by pixel, the final attention feature map can be obtained. The specific process is as follows.

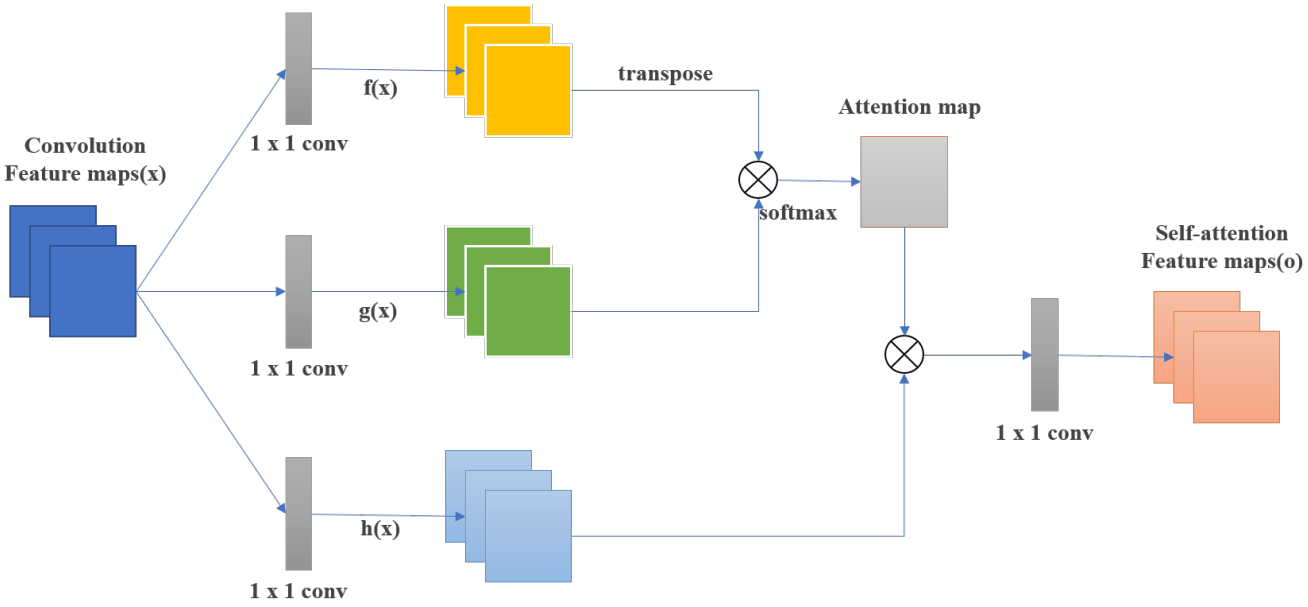


Figure 1. The self-attention module for SACVAEGAN. The $\otimes$ denotes matrix multiplication.

Suppose that the image features $x \in R^{C\times N}$ in the previous hidden layer are transformed into two features spaces f and g to calculate attention. Among them, $f(x) = W_f x, g(x) = W_g x$. $\beta_{j,i}$ represents the degree of participation in the i area when the model synthesizes the image content of the j area, that is, the correlation.

$$\beta_{j,i} = \frac{exp(s_{ij})}{\sum_{i=1}^{N} exp(s_{ij})}, \qquad s_{ij} = f(x_i)^T g(x_j) \tag{8}$$

The output of the attention layer is $o = (o_1, o_2, \ldots, o_j, \ldots, o_N \in R^{C\times N})$, C is the number of channels, and N is the number of feature positions of the previous hidden layer feature.

$$o_j = v(\sum_{i=1}^{N} \beta_{j,i} h(x_i)), \qquad h(x_i) = W_h x_i, \qquad v(x_i) = W_v x_i \tag{9}$$

Finally, the output of the attention layer is multiplied by a scale parameter, and the input feature map is added. Therefore, the final output is as follows:

$$y_i = \gamma o_i + x_i \tag{10}$$

To allow the network to pay attention to the neighborhood information, γ is a scalar that can be learned in the self-attention training process with the initial value of 0, and then, the weight is slowly assigned to other long-distance features.

3. Methodology

In this section, we introduce the details of the proposed SACVAEGAN method. As shown in Figure 2, the network structure consists of three modules: discriminator, VAE, and classifier. The module of the discriminator is used to determine whether the input samples are real samples or virtual samples. The VAE module is divided into two parts: encoder and generator. The encoder transfers real samples to the latent vector space, and the generator uses the latent vectors generated by the encoder and the random latent

vectors to generate virtual hyperspectral samples. The classifier module classifies the input real samples and virtual samples.

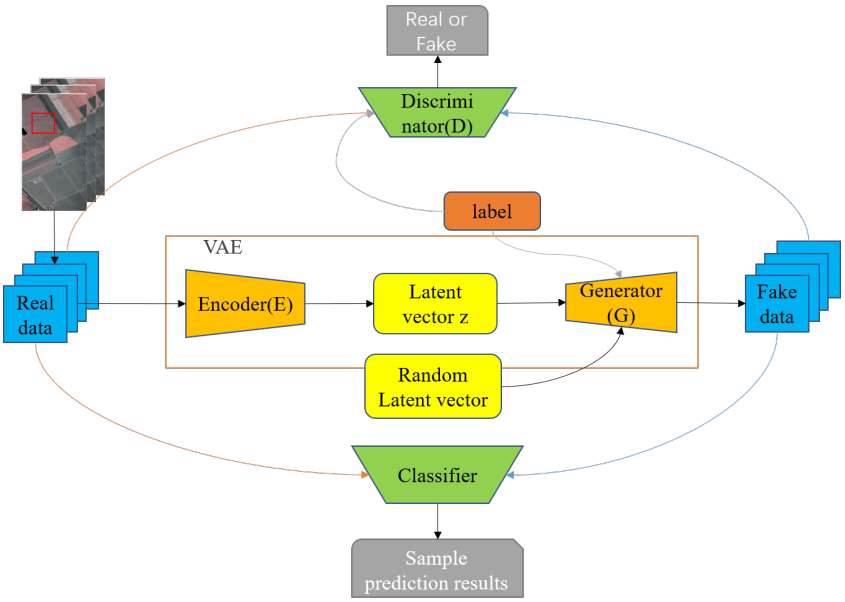


Figure 2. The structure of the proposed SACVAEGAN.

3.1. Discriminator

As shown in Figure 3, it is the module of the discriminator. It consists of four convolution layers. For each layer, the kernel size of each layer is 3×3. The self-attention mentioned in Section 2 is applied after the first and second convolution layers. After the last layer, we reshape the data into a feature vector. According to Reference [50], the label information should also be input to the discriminator to make the model more stable. Therefore, we reshape the label to a vector through a full connection layer and concatenate it with the feature vector. Then, a full connection layer is applied to reduce the dimension. At last, the sigmoid function is used to determine whether the data are real.

The loss function of the discriminator D is as follows:

$$L_D = L_{D_{GP}} + L_{D_{G(z_{random}|y)}} \tag{11}$$

The first term is the loss in WGAN-GP between x and $G(z|y)$, which can make the model more stable, and the second term is the loss of the discriminator, which judges whether $G(z_{random}|y)$ is false.

Among them:

$$L_{D_{GP}} = E_{z\sim P_z}[D(G(z|y)|y)] - E_{x\sim P(x_{real})}[D(x|y)] + \lambda E_{z\sim P_z}[(\| \bigtriangledown_{G(z|y)} D(G(z|y)|y)\|_2 - 1)^2] \tag{12}$$

$$L_{D_{x_{z_{random}}}} = -E_{z\sim p_{z_{random}}}[logD(G(z_{random}|y)|y)] \tag{13}$$

where z represents the latent vector generated by the encoder, z_{random} represents the latent vector generated randomly, x_{real} represents real samples, $G(z|y)$ represents the virtual samples generated by the generator according to z and the corresponding label, $G(z_{random}|y)$ represents the virtual sample generated by the generator according to z_{random} and the corresponding label, and y represents the label.

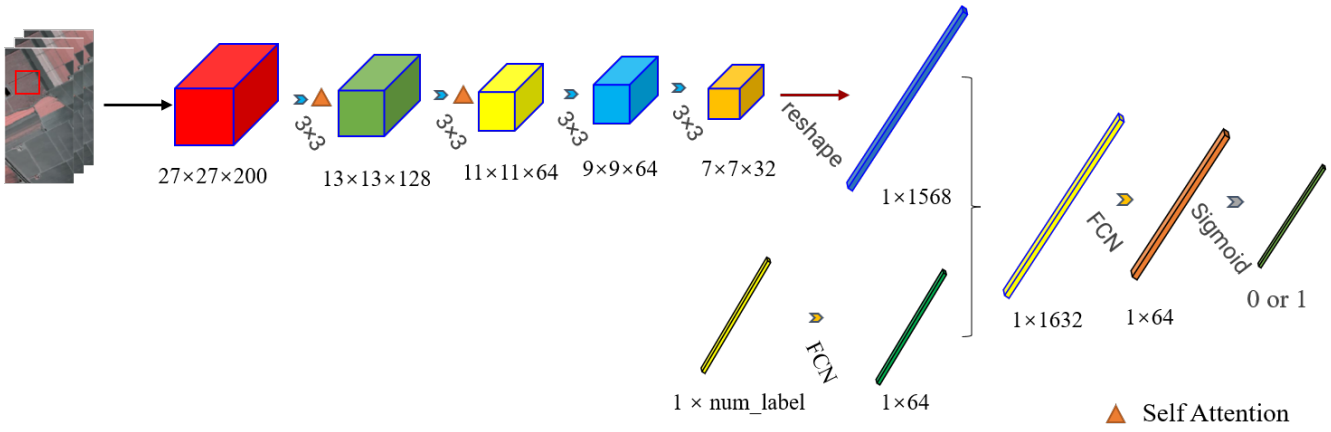


Figure 3. The structure of the discriminator D.

3.2. Variational Auto-Encoder

The module of the Variational Auto-Encoder (VAE) is shown in Figure 4. The VAE consists of two parts: the encoder and the generator. The encoder E is used to transfer the real samples to the latent vector space, while the generator G uses the latent vector to generate the virtual hyperspectral sample. We can see from Figure 4 that the encoder E is divided into two spectral-spatial feature extraction networks: one is used to get the mean vector μ, and the other is for the covariance ϵ of the latent vector space. For each feature extraction network, the network structure is the same. It consists of spectral and spatial feature extraction networks. For the spectral feature extraction network, it consists of 4 1-D convolution layers with 5×1 kernel. Self-attention is also applied to the first and second layers. For the spatial feature network, there are four 2-D convolution layers with 3×3 kernel and self-attention in the first and second layers. After the spectral and spatial feature extraction network, we concatenate the spectral-spatial feature together and use a full connection layer for dimension reduction. After we obtain the mean vector μ and the covariance ϵ, we use the following equation to obtain the latent vector:

$$z = \mu + r * exp(\epsilon) \tag{14}$$

The goal of generator G is to learn the distribution of training data and to generate virtual hyperspectral samples. For the generator G, it consists of two full connection layers and four transposed convolution layers. After obtaining the latent vector and the corresponding label, it uses two full connection layers to reshape the vector. Then, the vector is reshaped into a 3-D data cube, which is sent to the transposed convolution layers. The kernel size of the transposed convolution layer is 3×3. At last, we can obtain a virtual hyperspectral sample.

The loss function of VAE is as follows:

$$L_{VAE} = L_{kl} + L_G + L_{vae_D} + L_{vae_C} \tag{15}$$

The first term is the KL divergence, which is used to reduce the difference between the distribution of the obtained latent vector and the assumed latent vector distribution. The second term is the l_2 reconstruction loss between x and $G(z|y)$. The third term is the sum of the pair-wise feature matching loss between x and G(z | y) in the discriminator D and the loss of the discriminator in judging whether $G(z_{random}|y)$ is true. The last term is the sum of the pair-wise feature matching loss between x and $G(z|y)$ in the classifier C and the loss of the classification result of the classifier.

Among them:

$$L_{kl} = \frac{1}{2}(\mu^T \mu + sum(exp(\epsilon) - \epsilon - 1)) \tag{16}$$

$$L_G = \frac{1}{2}(\|x - G(z|y)\|_2^2) \tag{17}$$

$$L_{vae_D} = \|f_D(x) - f_D(G(z|y))\|_2^2 + E[log(1 - D(G(z_{random}|y)))] \tag{18}$$

$$L_{vae_C} = \|f_C(x) - f_C(G(z|y))\|_2^2 - E[logP(y|G(z_{random}|y))] \tag{19}$$

where μ and ϵ represent the mean value and covariance generated by the encoder, respectively; f_D represents the features of the middle layer in the discriminator D; f_C represents the features of the middle layer of the classifier; the real samples is represented as x_{real}; the virtual sample generated according to the latent vector generated by the encoder is $G(z|y)$; the virtual sample generated according to the randomly generated latent vector is $G(z_{random}|y)$; and y represents the label.

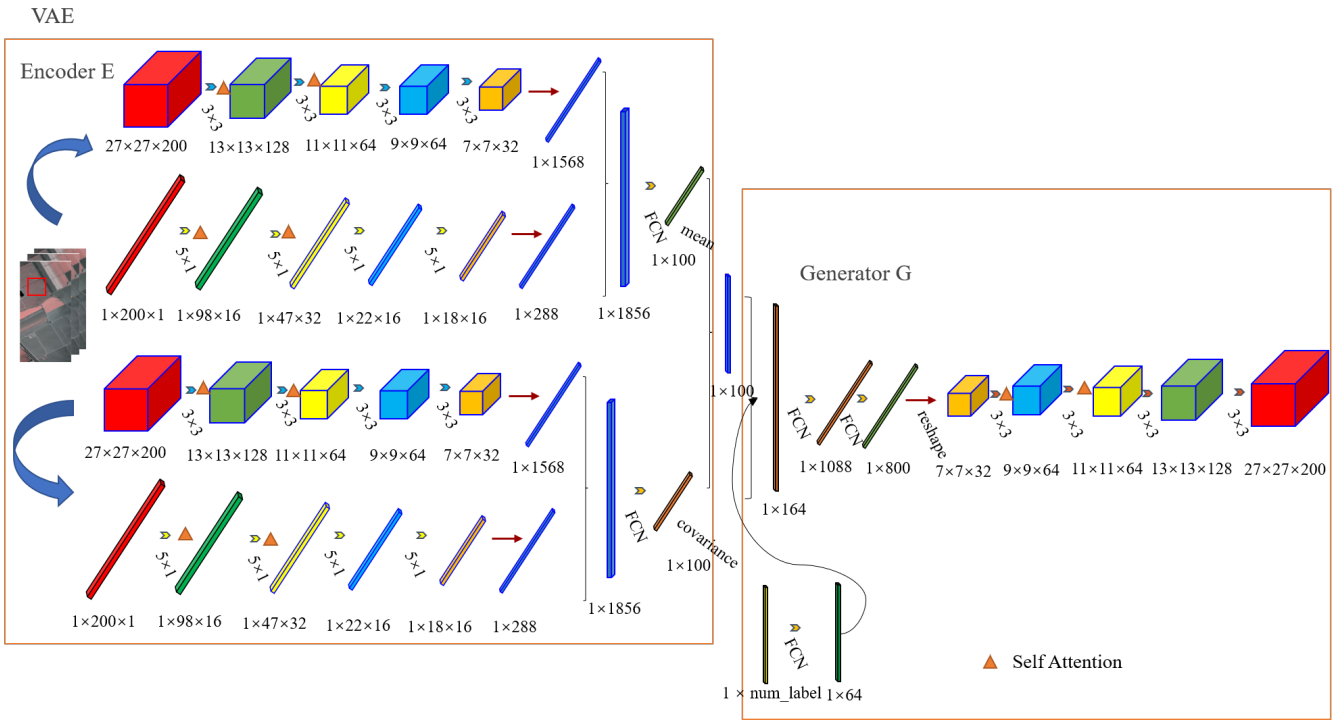


Figure 4. The structure of the *VAE*.

3.3. Classifier

The module of classifier C is shown in Figure 5. The purpose of classifier C is to obtain the classification results. For classifier C, it also consists of spectral-spatial feature extraction networks. For the spectral feature extraction network, there are five 1-D convolution layers with the kernel of 1×5, and for the spatial feature extraction network, it also consists of five 2-D convolution layers with 3×3 kernel. Finally, we concatenate the spectral and spatial features together and then send them to two full connection layers to obtain the final results. The loss function L_C is as follows:

$$L_C = L_{C_{x_{real}}} + \lambda_1 L_{C_{x_z}} + \lambda_2 L_{C_{x_{z_{random}}}} \tag{20}$$

where the first term is the loss of the result obtained by classifying x. The second term is the sum of the pair-wise feature matching loss between x and $G(z|y)$. The last term is the loss of the result obtained by classifying $G(z_{random}|y)$. λ_1 and λ_2 are the weights of $L_{C_{x_z}}$ and $L_{C_{x_{x_{random}}}}$ loss, respectively.

$$L_{C_{x_{real}}} = -E[logP(c|x_{real})] \tag{21}$$

$$L_{C_{x_z}} = \|f_C(x_{real}) - f_C(x_z)\|_2^2 \tag{22}$$

$$L_{C_{z_{random}}} = -E[logP(y|x_{z_{random}})] \tag{23}$$

where f_C represents the characteristics of the middle layer of the classifier, x_{real} represents the real samples, x_z represents the virtual sample generated by inputting the latent vector z generated by the encoder into the generator, $x_{z_{random}}$ represents the virtual sample generated by inputting the randomly generated latent vector z_{random} into the generator, and y represents the label.

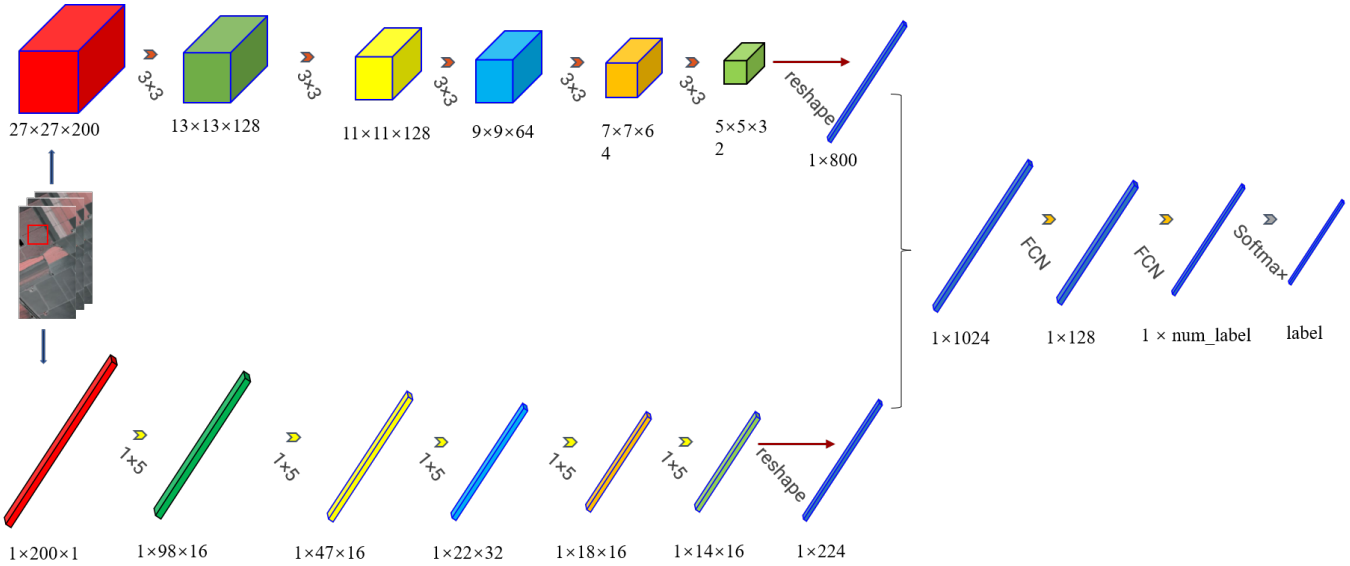


Figure 5. The structure of the classifier *C*.

The procedure of the proposed SACVAEGAN is summarized in Algorithm 1.

Algorithm 1: Optimization procedure of SACVAEGAN.

Input: the training samples x_{train}=$(x_i)_{i=1}^N$ and test samples $x_{test} = (x_i)_{i=1}^S$ from C classes, the onehot lables $(y_i)_{i=1}^N$ of training samples, batch size n, the number of training epochs E, noise dimension d, the updating times k of the discriminator

Initialize all the weight matrices and biases
while *every epoch* **do**
  sample d dimensional noises $z_{random} = (z_j)_{j=1}^d$;
  generate n latent vectors $z = (z_j)_{j=1}^d$ by Encoder E
  generate n virtual samples $G(z|y)$ through the generator according to the latent vector z ;
  generate n virtual samples $G(z_{random}|y)$ through the generator according to the latent vector z_{random};
  for *k steps* **do**
    updating parameters of the D by minimizing L_D
  end
  updating parameters of the VAE by minimizing L_{VAE}
  updating parameters of C by minimizing L_C
end

Output: the labels of the test samples x_{test} classified by the trained sample classifier in SACVAEGAN

4. Experiments

To evaluate the accuracy of our proposed SACVAEGAN, we conducted training and evaluation on three data sets, namely the Indian Pines data set, the PaviaU data set,

and the Salinas data set. We used three performance metrics: OA, AA, and K. OA is the overall classification accuracy, which represents the ratio between the categories that are properly classified and the total number of categories. AA is the average classification accuracy, which represents the average accuracy between categories. K represents the kappa coefficient for different weights in the confusion matrix.

4.1. Hyperspectral Data Sets

4.1.1. Indian Pines

The Indian Pines data set was collected by airborne visible infrared imaging spectrometer (AVRIS) sensors in Indian Pines testing in northwest Indiana. The AVIRIS sensor captured images in 0.4 to 2.5 microns. The data consist of 145 $\times$ 145 pixels and 224 spectral reflection bands. By deleting the bands covering the water absorption area, the number of spectral was reduced to 200. There are 16 categories in total, and the distribution of samples in each category is extremely unbalanced.

4.1.2. Paviau

The PaviaU data set was acquired by Reflective Optics System Imaging Spectrometer (ROSIS) sensors in Pavia region of northern Italy. The data consist of 610 $\times$ 340 pixels and 115 spectral reflection bands. By deleting 12 bands affected by noise, 103 spectral signatures remained. The size of the data is 610 $\times$ 340. However, most of them are background pixels, with only 42,776 pixels being foreground pixels in nine categories, including trees, Asphalt, Bricks, Meadows, etc.

4.1.3. Salinas

The Salinas data set was also collected by the AVIRIS sensor in the Salinas Valley, California, with a spatial resolution of 3.7 m. The data consist of 512 $\times$ 217 pixels and 224 spectral reflection bands. By deleting the bands covering the water absorption area, the spectrum was reduced to 204. The image size is 512 $\times$ 217 with 16 classes.

We divided the labeled samples in the three data sets into two parts: the training set and the testing set. Among them, for the Indian Pines data set, we chose 5% data for training. For the PaviaU data set, we chose 2% points as the training samples. For the Salinas data set, we chose 1% points as the training samples. Tables 1–3 show the number of training samples and testing samples for each category on the Indian Pines data set, PaviaU data set, and Salinas data set, respectively.

Table 1. Training and testing data in the Indian Pines data set.

No.	Class	Training	Test
1	Alfalfa	2	44
2	Corn-notill	71	1357
3	Corn-mintill	41	789
4	Corn	12	225
5	Grass-pasture	24	459
6	Grass-trees	37	693
7	Grass-pasture-mowed	1	27
8	Hay-windrowed	24	454
9	Oats	1	19
10	Soybean-notill	49	923
11	Soybean-mintill	123	2332
12	Soybean-clean	30	563
13	Wheat	10	195
14	Woods	63	1202
15	Buildings-Grass-Trees-Drives	19	367
16	Stone-Steel-Towers	5	88
Total		512	9737

Table 2. Training and testing data in the PaviaU data set.

No.	Class	Training	Test
1	Asphalt	132	6499
2	Meadows	373	18,276
3	Gravel	42	2057
4	Trees	61	3003
5	Painted metal sheets	27	1318
6	Bare Soil	100	4929
7	Bitumen	27	1303
8	Self-Blocking Bricks	74	3608
9	Shadows	19	928
Total		855	41,921

Table 3. Training and testing data in the Salinas data set.

No.	Class	Training	Test
1	Brocoli_green_weeds_1	20	1989
2	Brocoli_green_weeds_2	37	3689
3	Fallow	20	1956
4	Fallow_rough_plow	14	1380
5	Fallow_smooth	27	2651
6	Stubble	39	3920
7	Celery	36	3543
8	Grapes_untrained	113	11,158
9	Soil_vinyard_develop	62	6141
10	Corn_senesced_green_weeds	33	3245
11	Lettuce_romaine_4wk	11	1057
12	Lettuce_romaine_5wk	19	1908
13	Lettuce_romaine_6wk	9	907
14	Lettuce_romaine_7wk	11	1059
15	Vinyard_untrained	72	7196
16	Vinyard_vertical_trellis	18	1789
Total		541	53,588

4.2. Parameter Analysis

Before training the model, some important factors should be analyzed, which may affect the results. Due to the introduction of WGAN-GP, we used RMSprop as the optimizer in the VAE module and the discriminator module. The Adam optimizer was used in the classifier module. We mainly discuss three factors: the different size of patch sizes, the network regularization method, and the parameters of λ_1 and λ_2.

4.2.1. Analysis of the Size of Patches

To analyze the influence of the size of patches, we calculated the overall accuracy (OA) on three data sets using different sizes of patches. As shown in the Table 4, the size of the patches is $19 \times 19, 23 \times 23, 27 \times 27, 31 \times 31$. It can be seen from Table 4 for the Indian Pines and the PaviaU data sets. When the patch size was 27, the proposed method obtained the best performance, while for the Salinas data set, when the size was set to 31, it obtained the best results. Therefore, in the experiments, we set the patch size as 27, with better overall accuracy (OA).

Table 4. Overall accuracy of different patch sizes in the Indian pines, PaviaU, and Salinas data sets.

Patch Size	Indian Pines	PaviaU	Salinas
19 × 19	92.78	98.91	97.05
23 × 23	94.78	99.01	97.80
27 × 27	**95.94**	**99.32**	98.82
31 × 31	95.02	99.10	**99.19**

4.2.2. Analysis of the Network Regularization Method

To analysis the regularization methods, we used different methods to regularize our proposed method, including batch normalization (BN) and dropout. Batch normalization can normalize the input data and can solve the problem of the gradient disappearing and the gradient exploding. It can also accelerate the convergence speed. Dropout enhances the generalization of the model by making the activation value have a certain probability p during forward propagation. As shown in Table 5, when the proposed method uses dropout and batch normalization, it obtains the best performance. Therefore, we applied dropout and Batch Normalization (BN) in our proposed method.

Table 5. Network regularization method in the Indian Pines, PaviaU, and Salinas data sets.

Proposeed	Indian Pines	PaviaU	Salinas
None	94.01	98.37	97.70
Dropout	95.26	98.26	98.00
BN	95.50	98.63	98.21
Both	**95.94**	**99.32**	**98.82**

4.2.3. Analysis of the λ_1 and λ_2 in the Classifier Loss Function

In order to analyze the influence of λ_1 and λ_2 in Equation (17), we conducted experiments with different values of λ_1 and λ_2. We set λ_1 from 0 to 0.5 and λ_2 from 0.5 to 1 with the interval of 0.1. We calculated the performance on three data sets. As can be seen in Tables 6–8, when $\lambda_1 = 0.8$ and $\lambda_2 = 0.2$, our model obtains the best average classification accuracy. Therefore, we set the value of λ_1 to 0.8, and λ_2 to 0.2 in our experiments.

Table 6. Overall accuracy in the Indian Pines data set with different λ_1 and λ_2.

λ_2 \ λ_1	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
0	94.50	94.82	95.23	95.73	95.67	95.39
0.1	94.51	94.71	95.35	95.32	95.71	95.32
0.2	93.56	94.75	95.46	**95.94**	95.76	95.44
0.3	93.72	93.32	94.67	95.55	95.10	95.59
0.4	93.85	94.10	94.43	94.57	94.66	94.91
0.5	93.43	93.87	94.22	94.01	94.62	94.68

Table 7. Overall accuracy in the PaviaU data set with different λ_1 and λ_2.

λ_2 \ λ_1	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
0	98.99	99.01	99.05	99.24	99.10	98.82
0.1	98.83	98.74	99.21	99.28	99.14	99.07
0.2	98.91	99.12	99.27	**99.32**	99.29	99.19
0.3	98.61	98.67	99.01	99.13	99.07	98.91
0.4	98.69	98.43	98.87	98.57	98.76	98.59
0.5	98.34	98.31	97.89	98.46	98.39	98.21

Table 8. Overall accuracy in the Salinas data set with different λ_1 and λ_2.

λ_2 \ λ_1	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
0	98.47	98.51	98.57	98.65	98.41	98.61
0.1	98.38	98.42	98.58	98.66	98.72	98.67
0.2	98.46	98.32	98.78	**98.82**	98.81	98.79
0.3	98.12	98.09	98.41	98.31	98.53	98.37
0.4	97.75	98.18	98.43	98.29	98.05	98.17
0.5	97.37	97.51	98.25	97.93	98.11	98.03

4.3. Classification Results

To illustrate the effectiveness of our proposed method, SACVAEGAN was compared with several different hyperspectral image classification methods, including SVM-Radial Basis Function (RBF), Two-CNN [31], 3D-CNN [22], DCGAN [36], DBGAN [41], and CVAEGAN [37]. SVM-Radial Basis Function uses a Gaussian kernel function, and the parameter gamma of the kernel function was set to 0.125. The hyperparameters of the deep learning method were all set to the parameters mentioned in the corresponding paper. We used the classifier module for the final classification. Tables 9–11 show the comparison of the methods we proposed on the Indian Pines data set, the PaviaU data set, and the Salinas data set. All of the classification methods were tuned to the best settings. The training set and test set are shown in Tables 1–3.

Tables 9–11 show the results of the proposed methods on the Indian Pines data set, the PaviaU data set, and the Salinas data set, respectively. From the tables, we can see that the performance of the deep learning methods is better than the traditional method. This is because the deep learning method can extract better features. Moreover, for the GAN-based method, the performance is better than other deep learning methods because the GAN-based method could generate more training data, which could help the training. From these tables, we can see that our proposed method obtained the best performance. This may be due to the following reasons: First, compared with DCGAN and DBGAN, the VAE module is applied to our proposed method, which makes it higher. Second, compared with CVAEGAN, the conditional GAN is also applied to our proposed method, which could generate more high-quality training data and can enhance the performance.

Figures 6–8 show the false color images of the three HSI data sets, the corresponding ground truth maps, and the classification maps of each method. It can be seen that the classification result diagram is consistent with the classification results shown in Tables 9–11. From these figures, we can see that the classification maps of our proposed method are clearer, less noisy, and closer to the ground truth. This indicates that our proposed method has a better classification capability than other methods.

Table 9. Classification results on the Indian Pines data set.

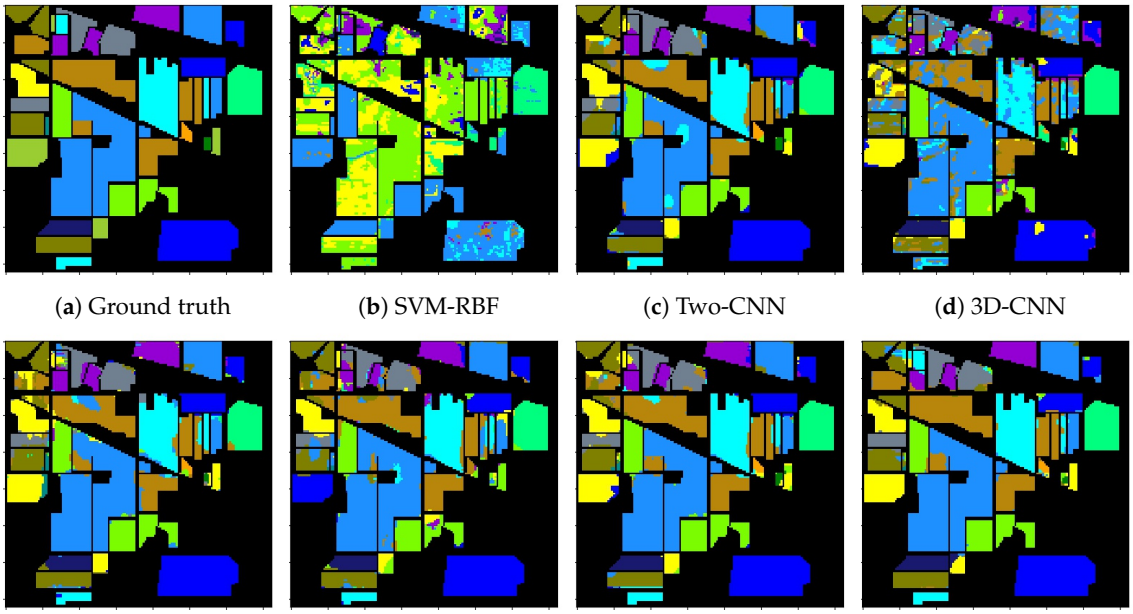
Class	SVM−RBF	Two−CNN	3D−CNN	DCGAN	DBGAN	CVAEGAN	Proposed
1	0	69.78 ± 1.98	17.70 ± 1.11	**75.97 ± 0.04**	37.83 ± 7.44	71.01 ± 2.37	66.67 ± 3.29
2	28.07 ± 2.63	84.78 ± 0.26	68.96 ± 0.84	85.60 ± 0.01	77.23 ± 1.38	90.13 ± 0.27	**97.15 ± 0.38**
3	0.33 ± 0.99	88.39 ± 0.59	68.11 ± 1.46	90.72 ± 0.01	81.16 ± 1.26	88.76 ± 0.91	**91.04 ± 1.63**
4	1.11 ± 2.90	94.68 ± 0.42	67.75 ± 3.67	**95.36 ± 0.01**	89.37 ± 3.95	90.44 ± 2.88	92.55 ± 1.80
5	2.11 ± 4.08	80.68 ± 0.75	87.52 ± 0.84	81.78 ± 0.02	24.43 ± 5.88	84.82 ± 2.06	**91.58 ± 1.05**
6	92.58 ± 2.35	95.04 ± 0.32	88.26 ± 0.65	95.62 ± 0.01	91.84 ± 0.97	**99.45 ± 0.11**	97.99 ± 0.31
7	0	90.00 ± 2.66	65.31 ± 6.28	**81.07 ± 0.05**	14.29 ± 9.16	79.76 ± 1.94	42.86 ± 2.92
8	99.21 ± 1.27	97.05 ± 0.22	99.16 ± 0.21	97.47 ± 0.00	98.42 ± 0.32	**100 ± 0**	98.60 ± 0.82
9	0	36.00 ± 0.63	60.00 ± 6.06	**80.00 ± 0.04**	29.50 ± 6.65	45.00 ± 4.08	38.33 ± 5.93
10	0.35 ± 1.04	84.44 ± 0.47	68.90 ± 0.77	87.14 ± 0.01	83.70 ± 0.69	89.27 ± 0.93	**96.02 ± 1.3**
11	94.99 ± 1.76	93.18 ± 0.27	80.14 ± 0.72	96.99 ± 0.00	94.12 ± 0.75	97.65 ± 0.64	**98.56 ± 0.31**
12	0.23 ± 0.54	82.48 ± 0.38	54.85 ± 2.36	83.98 ± 0.01	73.88 ± 2.44	85.83 ± 1.33	**89.71 ± 0.68**
13	0.29 ± 30.79	94.44 ± 0.46	98.47 ± 0.59	96.59 ± 0.01	95.41 ± 0.48	99.02 ± 0.61	**99.84 ± 0.13**
14	98.66 ± 0.38	96.17 ± 0.19	92.94 ± 0.36	98.18 ± 0.01	97.53 ± 0.40	**99.31 ± 0.04**	98.58 ± 0.78
15	2.70 ± 4.24	95.52 ± 0.62	69.73 ± 1.18	**97.93 ± 0.02**	87.42 ± 2.30	92.57 ± 2.20	95.51 ± 0.78
16	82.61 ± 4.37	84.95 ± 1.84	80.49 ± 2.10	88.17 ± 0.02	50.86 ± 8.51	78.49 ± 0.51	**92.47 ± 2.63**
OA (%)	51.69 ± 0.73	90.10 ± 0.21	77.80 ± 0.46	91.11 ± 0.45	84.42 ± 0.37	93.47 ± 0.11	**95.94 ± 0.20**
AA (%)	33.27 ± 1.75	85.41 ± 0.35	73.02 ± 0.73	**87.53 ± 0.55**	67.00 ± 0.29	86.97 ± 0.24	86.71 ± 0.69
Kappa (%)	41.46 ± 0.92	88.73 ± 0.24	74.68 ± 0.52	89.87 ± 0.51	82.14 ± 0.42	82.54 ± 0.13	**95.36 ± 0.22**

Table 10. Classification results on the PaviaU data set.

Class	SVM−RBF	Two−CNN	3D−CNN	DCGAN	DBGAN	CVAEGAN	Proposed
1	91.49 ± 1.00	94.81 ± 0.16	96.91 ± 0.18	92.70 ± 0.35	96.03 ± 0.37	96.31 ± 0.53	**99.60 ± 0.15**
2	96.80 ± 2.76	99.51 ± 0.07	99.15 ± 0.08	99.40 ± 0.16	99.67 ± 0.09	99.98 ± 0.02	**99.99 ± 0.01**
3	1.89 ± 3.50	79.15 ± 0.28	85.88 ± 0.85	72.87 ± 3.10	80.98 ± 3.09	90.41 ± 0.68	**97.09 ± 1.01**
4	63.57 ± 12.59	94.14 ± 0.25	94.73 ± 0.45	91.33 ± 1.52	97.90 ± 0.33	95.96 ± 0.22	**97.96 ± 0.44**
5	98.87 ± 0.16	98.74 ± 0.34	99.72 ± 0.06	99.48 ± 0.16	98.23 ± 0.64	**100 ± 0**	**100 ± 0**
6	14.84 ± 1.79	96.88 ± 0.38	95.32 ± 0.49	96.34 ± 0.91	98.90 ± 0.31	99.25 ± 0.37	**100 ± 0**
7	0	84.09 ± 0.29	86.46 ± 0.99	85.41 ± 3.62	90.79 ± 1.29	94.29 ± 2.94	**96.54 ± 1.95**
8	91.90 ± 2.59	92.15 ± 0.24	94.41 ± 0.33	83.13 ± 0.20	93.83 ± 1.04	**98.75 ± 0.26**	98.23 ± 0.68
9	**99.82 ± 0.08**	92.73 ± 0.73	97.02 ± 0.54	86.91 ± 0.37	92.67 ± 1.79	90.11 ± 1.02	97.04 ± 0.85
OA (%)	76.00 ± 0.49	95.80 ± 0.09	96.55 ± 0.17	93.88 ± 0.39	96.99 ± 0.34	98.07 ± 0.04	**99.32 ± 0.06**
AA (%)	62.13 ± 1.16	92.47 ± 0.14	94.40 ± 0.33	89.19 ± 0.23	95.33 ± 0.21	96.12 ± 0.37	**98.49 ± 0.14**
Kappa (%)	66.28 ± 0.48	94.41 ± 0.12	95.42 ± 0.23	91.86 ± 0.51	96.00 ± 0.45	97.43 ± 0.05	**99.10 ± 0.08**

Table 11. Classification results on the Salinas data set.

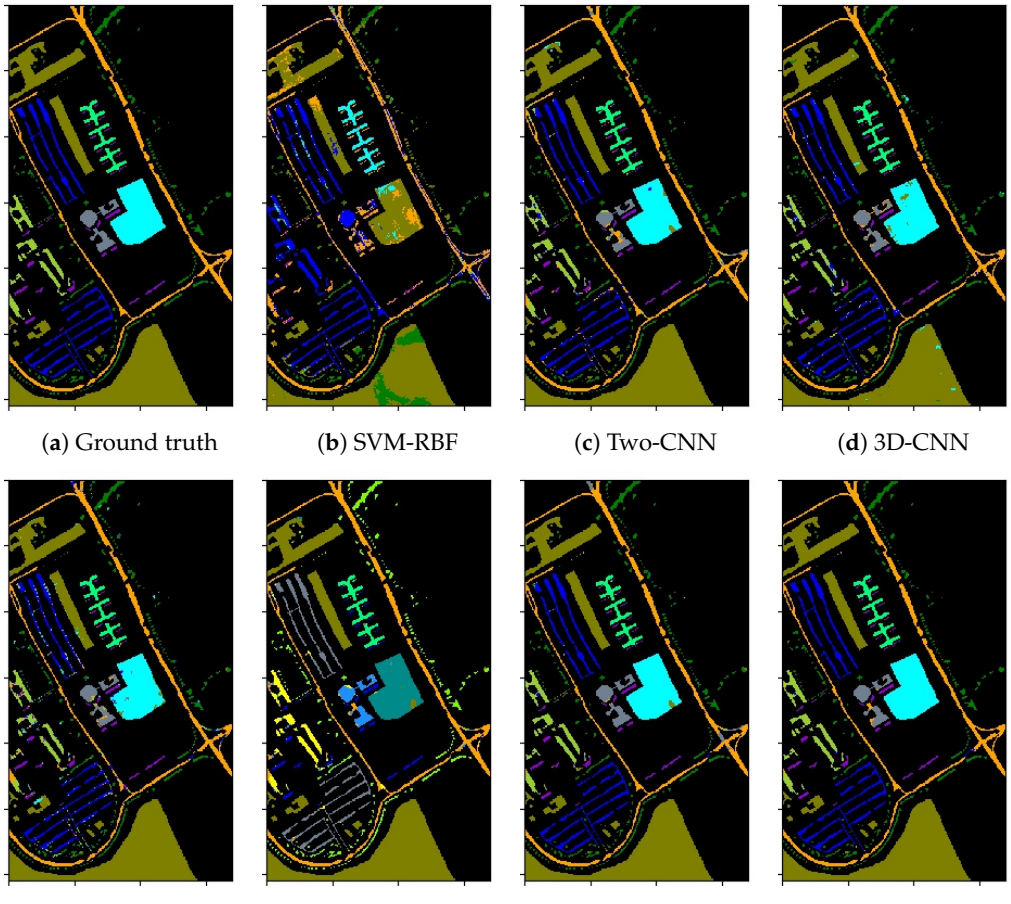
Class	SVM−RBF	Two−CNN	3D−CNN	DCGAN	DBGAN	CVAEGAN	Proposed
1	94.92 ± 1.43	93.16 ± 0.61	97.58 ± 0.14	96.33 ± 1.10	23.48 ± 0.19	**99.93 ± 0.05**	99.30 ± 0.31
2	95.36 ± 1.43	98.79 ± 0.15	99.84 ± 0.04	92.39 ± 1.65	97.82 ± 1.49	**100 ± 0**	98.60 ± 0.57
3	60.68 ± 4.33	99.65 ± 0.16	98.46 ± 0.31	92.02 ± 1.36	98.50 ± 0.83	99.19 ± 0.45	**99.85 ± 0.10**
4	98.44 ± 0.36	96.90 ± 0.37	**98.76 ± 0.24**	97.20 ± 0.40	97.30 ± 0.58	97.87 ± 0.59	98.02 ± 0.77
5	96.83 ± 0.56	96.15 ± 0.20	87.56 ± 0.93	97.53 ± 0.50	96.03 ± 1.25	98.32 ± 0.33	**99.22 ± 0.20**
6	98.25 ± 0.79	98.65 ± 0.18	98.86 ± 0.24	91.02 ± 0.20	99.78 ± 0.11	**99.99 ± 0.01**	99.92 ± 0.03
7	99.05 ± 0.17	96.12 ± 0.32	98.10 ± 0.31	96.68 ± 0.58	98.70 ± 0.32	**100 ± 0**	99.95 ± 0.03
8	96.77 ± 2.87	94.22 ± 0.28	83.66 ± 0.69	81.53 ± 0.96	96.06 ± 0.58	97.23 ± 0.89	**97.79 ± 0.36**
9	98.65 ± 0.36	98.36 ± 0.13	97.29 ± 0.15	99.18 ± 0.33	98.82 ± 0.50	**99.80 ± 0.17**	99.44 ± 0.29
10	61.48 ± 11.90	99.78 ± 0.09	92.75 ± 0.53	97.87 ± 0.46	98.45 ± 0.36	99.30 ± 0.30	**99.82 ± 0.13**
11	0.11 ± 0.34	90.69 ± 0.24	89.96 ± 0.97	92.08 ± 1.63	63.55 ± 0.30	93.07 ± 0.54	**96.47 ± 0.57**
12	60.40 ± 7.68	91.99 ± 0.14	98.71 ± 0.26	91.26 ± 0.82	95.99 ± 0.80	97.61 ± 1.15	**99.12 ± 0.47**
13	38.82 ± 39.97	90.97 ± 0.54	**98.76 ± 0.14**	95.82 ± 0.92	94.07 ± 0.48	95.49 ± 1.28	96.98 ± 0.49
14	87.97 ± 1.71	93.90 ± 0.29	94.90 ± 0.35	96.50 ± 0.96	95.95 ± 0.78	97.69 ± 0.45	**98.57 ± 0.45**
15	2.91 ± 5.88	92.06 ± 0.19	75.98 ± 1.05	78.98 ± 1.68	97.93 ± 0.28	97.67 ± 0.30	**98.81 ± 0.55**
16	53.00 ± 3.16	**99.23 ± 0.15**	94.23 ± 1.14	81.08 ± 2.10	97.34 ± 0.80	98.73 ± 0.63	97.79 ± 0.47
OA (%)	75.25 ± 1.23	95.79 ± 0.07	91.05 ± 0.24	90.39 ± 0.35	94.13 ± 0.94	98.49 ± 0.01	**98.82 ± 0.11**
AA (%)	71.48 ± 3.01	95.66 ± 0.06	94.09 ± 0.19	92.80 ± 0.30	90.61 ± 2.34	98.24 ± 0.29	**98.72 ± 0.08**
Kappa (%)	71.92 ± 1.46	95.31 ± 0.07	90.04 ± 0.27	89.30 ± 0.39	93.46 ± 1.04	98.69 ± 0.12	**98.80 ± 0.01**



(**a**) Ground truth (**b**) SVM-RBF (**c**) Two-CNN (**d**) 3D-CNN

(**e**) DCGAN (**f**) DBGAN (**g**) CVAEGAN (**h**) SACVAEGAN

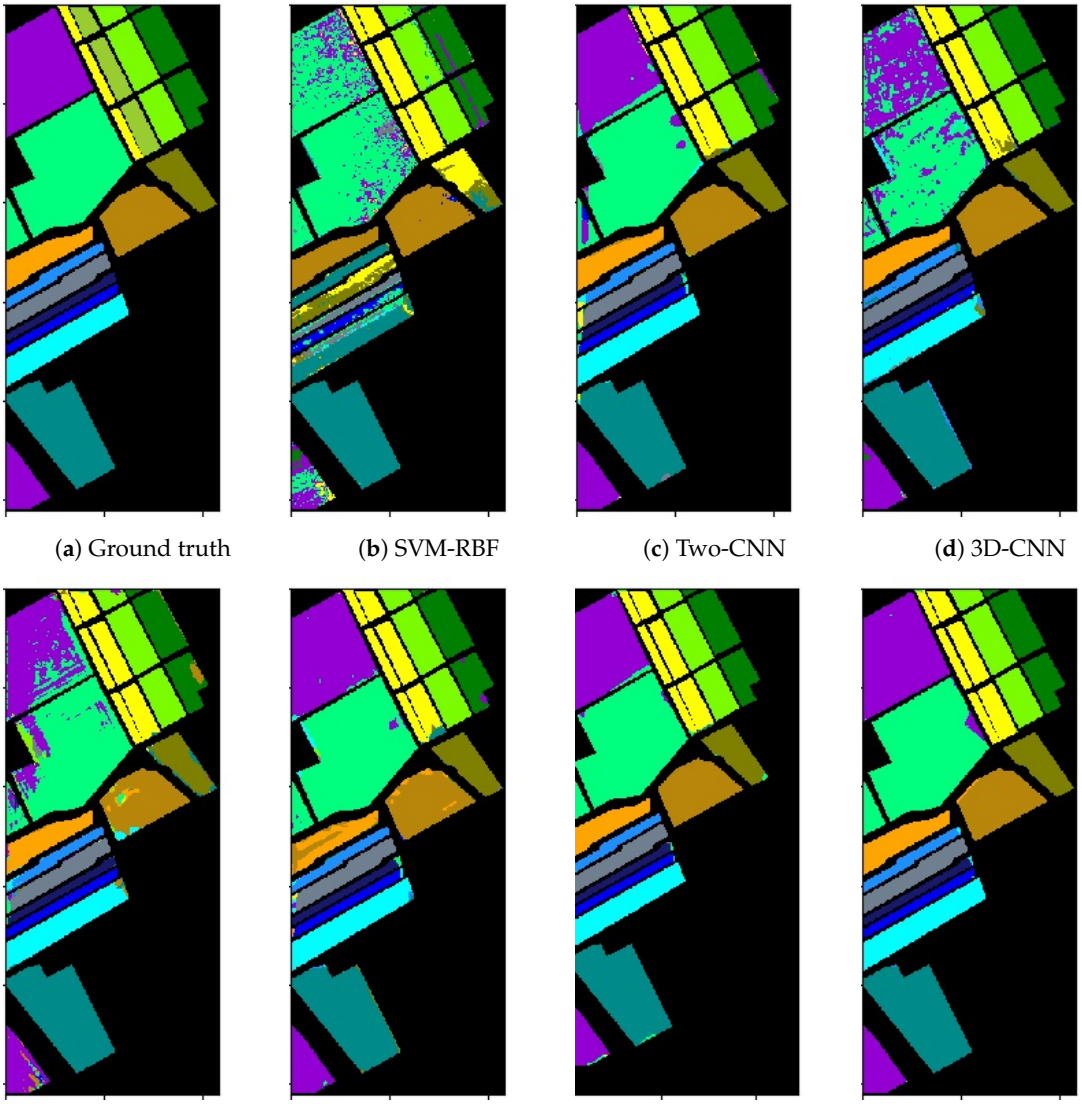
Figure 6. Illustration of the classification results on the Indian Pines data set. (**a**) Ground truth, (**b**) SVM-RBF (51.69%), (**c**) Two-CNN (90.10%), (**d**) 3D-CNN (77.80%), (**e**) DCGAN (91.11%), (**f**) DBGAN (84.42%), (**g**) CVAEGAN (93.47%), and (**h**) SACVAEGAN (95.98%).



(**a**) Ground truth (**b**) SVM-RBF (**c**) Two-CNN (**d**) 3D-CNN

(**e**) DCGAN (**f**) DBGAN (**g**) CVAEGAN (**h**) SACVAEGAN

Figure 7. Illustration of the classification results on the PaviaU data set. (**a**) Ground truth, (**b**) SVM-RBF (76.00%), (**c**) Two-CNN (95.80%), (**d**) 3D-CNN (96.55%), (**e**) DCGAN (93.88%), (**f**) DBGAN (96.99%), (**g**) CVAEGAN (98.07%), and (**h**) SACVAEGAN (98.30%).



(**a**) Ground truth (**b**) SVM-RBF (**c**) Two-CNN (**d**) 3D-CNN

(**e**) DCGAN (**f**) DBGAN (**g**) CVAEGAN (**h**) SACVAEGAN

Figure 8. Illustration of the classification results on the PaviaU data set. (**a**) Ground truth, (**b**) SVM-RBF (75.25%), (**c**) Two-CNN (95.79%), (**d**) 3D-CNN (91.05%), (**e**) DCGAN (90.39%), (f) DBGAN (94.13%), (**g**) CVAEGAN (98.49%), and (**h**) SACVAEGAN (98.92%).

4.4. Analysis of the Size of the Training Set

To verify the robustness of SACVAEGAN for different sizes of training data, we selected different proportions of samples for training. For the three data sets, we selected 1 to 5% data for training. We show the results in Figure 9. From Figure 9, we can see that our proposed method obtains a better performance than other methods. This demonstrates that our proposed method is more stable compared with other methods and robust to different sizes of training data.

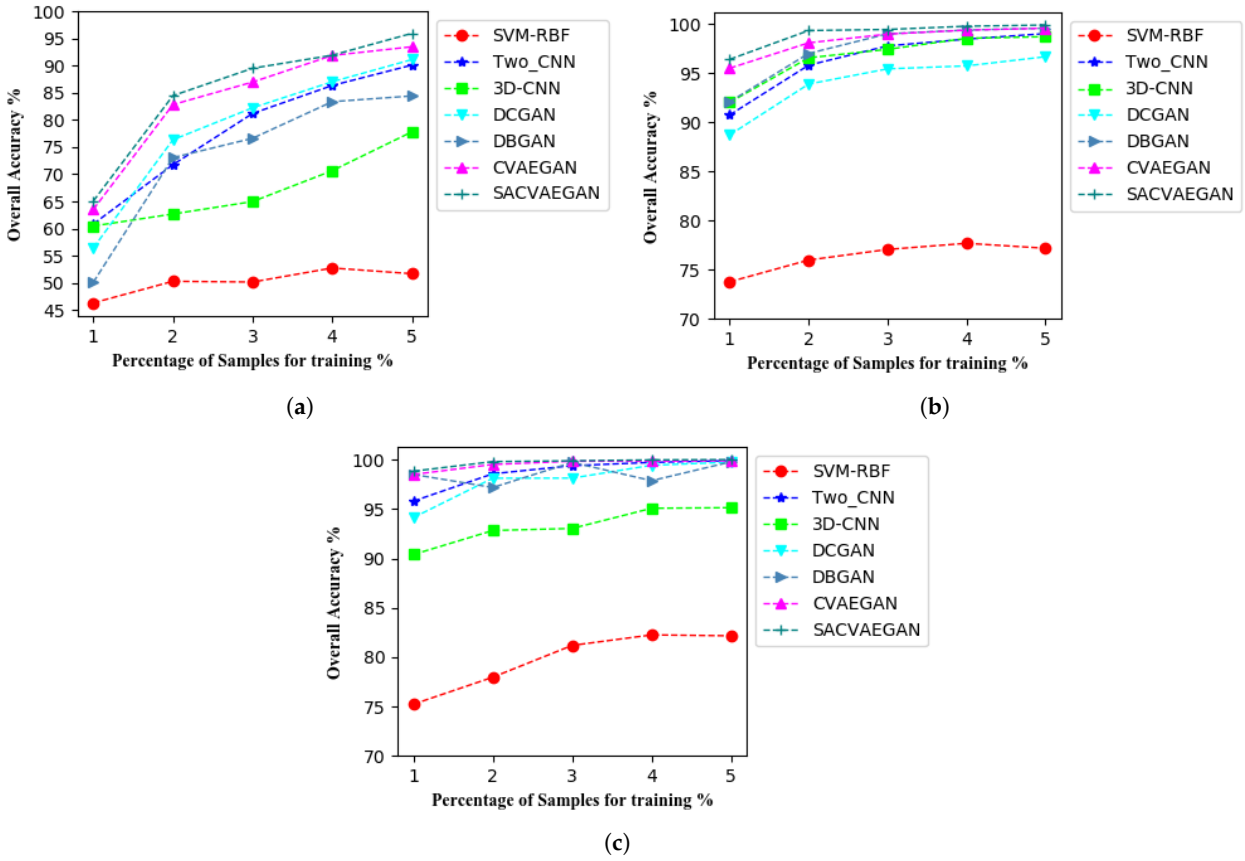


Figure 9. OA(%) of different methods with different training percentages of samples. (**a**) Indian Pines. (**b**) PaviaU. (**c**) Salinas.

4.5. Investigation on the Run Time

To evaluate the effectiveness of our proposed method, we show the training and testing time of seven different methods in Table 12. The experiments were run with an NVIDIA GTX 1660 GPU and an Inter i5-9300H 2.40-GHz CPU with 16 GB of RAM. It can be seen from Table 12 that the traditional method is much faster than the deep learning method. The deep learning method using the generative adversarial network is slower in training speed than and similar in test speed to the other deep learning methods. The method proposed in this paper requires a relatively long training time on the three data sets. The reason is that our proposed method extracts spectral and spatial features for training to enhance the performance, so more training time is required.

Table 12. Training and testing time on the three data sets.

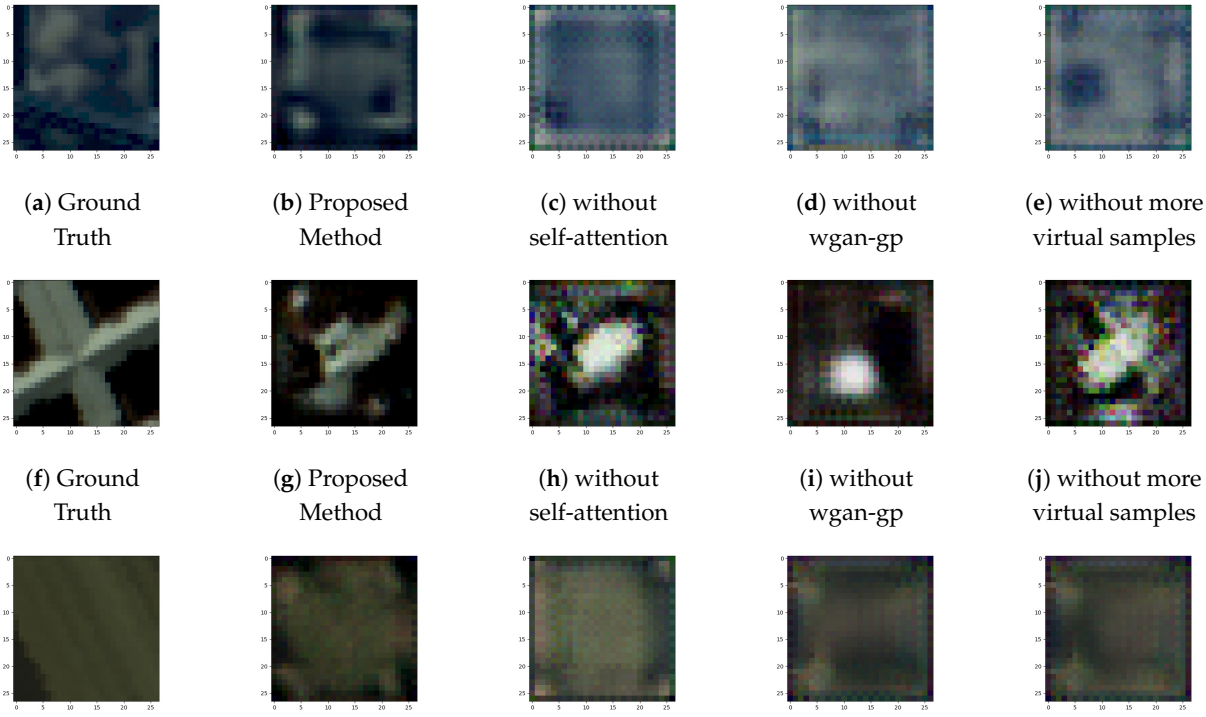
Algorithms	Time	Indian Pines	PaviaU	Salinas
SVM−RBF	Train(s)	0.0098	0.0080	0.0043
	Test(s)	0.1366	0.4065	0.5430
Two−CNN	Train(s)	2023.64	4306.78	10,957.61
	Test(s)	22.47	48.75	140.63
3D−CNN	Train(s)	448.00	1565.82	2559.34
	Test(s)	5.27	15.12	21.38
DCGAN	Train(s)	7019.69	14,053.51	10,273.40
	Test(s)	7.28	25.63	31.62
DBGAN	Train(s)	3485.74	7922.47	6551.87
	Test(s)	4.57	13.73	18.06
CVAEGAN	Train(s)	4394.08	8514.97	6988.92
	Test(s)	3.70	16.47	19.72
Proposed	Train(s)	14,101.28	25,665.42	17,524.97
	Test(s)	10.26	43.10	102.79

5. Discussion

To analyze the influence of the self-attention mechanism, WGAN-GP loss, and additional generated samples on the classification accuracy of the proposed method, we conducted several ablation experiments to analyze the spatial features, spectral features, and overall accuracy under different conditions.

5.1. Spatial Feature Analysis

To illustrate the advantages of the spatial features of the generated samples, we draw the corresponding virtual samples under different conditions. As shown in Figure 10, it can be seen that the samples generated by our method are closer to the real sample distribution. Thus, the detailed features captured by the virtual sample can improve the classification performance of the model in return. To analyze the difference between the spatial features of the generated virtual sample and the real sample, we calculated the mean square error (MSE) between the generated sample and the real sample. The mean square error can be calculated using Equation (24). As shown in Table 13, the sample generated by our proposed method is closer to the real sample compared with other methods.



(**a**) Ground Truth (**b**) Proposed Method (**c**) without self-attention (**d**) without wgan-gp (**e**) without more virtual samples

(**f**) Ground Truth (**g**) Proposed Method (**h**) without self-attention (**i**) without wgan-gp (**j**) without more virtual samples

(**k**) Ground Truth (**l**) Proposed Method (**m**) without self-attention (**n**) without wgan-gp (**o**) without more virtual samples

Figure 10. Illustration of the spatial features on the Indian Pines (**a**–**e**), PaviaU (**f**–**j**), and Salinas (**k**–**o**) data sets.

Table 13. MSE when self-attention, WGAN-GP, and additional virtual samples are removed.

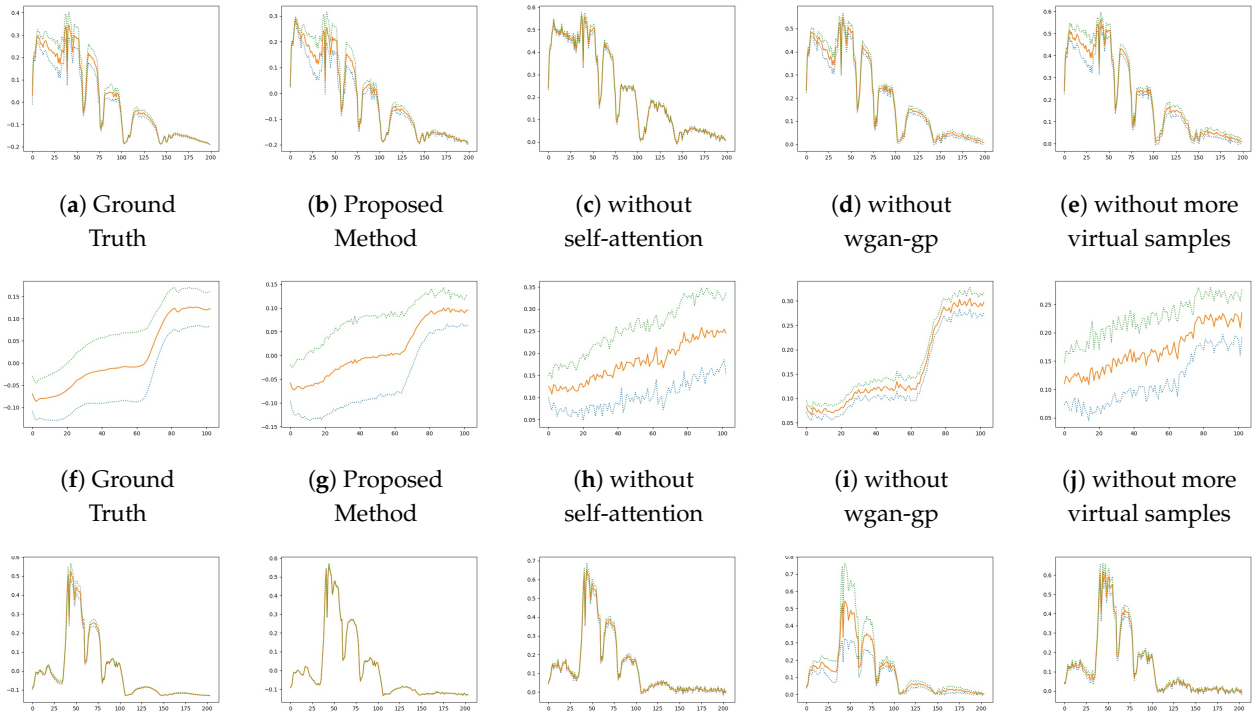
Algorithms	Indian Pines	PaviaU	Salinas
SACVAEGAN without self-attention	0.021	0.048	0.013
SACVAEGAN without wgan-gp	0.022	0.052	0.023
SACVAEGAN without additional virtual samples	0.020	0.046	0.021
SACVAEGAN	0.017	0.043	0.011

$$MSE(x,y) = \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n} \tag{24}$$

5.2. Spectral Feature Analysis

To illustrate the advantages of spectral feature, we plot the spectral feature map corresponding to the generated virtual sample under different conditions in Figure 11. It can be seen that the spectral feature distribution of the virtual sample generated by our method is more consistent with the spectral feature distribution of real samples. In order to analyze the difference in spectral features between the virtual sample and the real sample, we calculated the spectral information divergence (SID) between the virtual sample and the real sample. SID is based on the theory of information theory to measure the difference between two spectral calculated by Equation (25). The smaller the SID value, the more similar the spectral. As shown in Table 14, the proposed method obtains the optimal SID on the three data sets.

$$SID(x,y) = \sum_{i=1}^{l} p_i log(\frac{p_i}{q_i}) + \sum_{i=1}^{l} q_i log(\frac{q_i}{p_i}) \tag{25}$$



(**a**) Ground Truth (**b**) Proposed Method (**c**) without self-attention (**d**) without wgan-gp (**e**) without more virtual samples

(**f**) Ground Truth (**g**) Proposed Method (**h**) without self-attention (**i**) without wgan-gp (**j**) without more virtual samples

(**k**) Ground Truth (**l**) Proposed Method (**m**) without self-attention (**n**) without wgan-gp (**o**) without more virtual samples

Figure 11. Illustration of the spectral features on the Indian Pines (**a**–**e**), PaviaU (**f**–**j**), Salinas (**k**–**o**) data sets. The orange line represents the mean, the green line represents the mean add the variance, and the blue line represents the mean minus the variance.

Table 14. SID when self-attention, WGAN-GP, and additional virtual samples are removed.

Algorithms	Indian Pines	PaviaU	Salinas
SACVAEGAN without self-attention	0.019	0.041	0.003
SACVAEGAN without wgan-gp	0.012	0.021	0.028
SACVAEGAN without additional virtual samples	0.011	0.035	0.011
SACVAEGAN	0.011	0.014	0.002

5.3. Overall Accuracy Analysis

Self-attention enables the model to better extract global features. For the WGAN-GP, it can make the model more stable and can improve performance. Additional virtual samples can improve the generalization performance and classification accuracy. To analyze the advantage of each module, we run the experiments with or without each module for 10 times. As shown in Table 15, each module contributes to the improvement of classification accuracy. When the proposed method uses all three strategies, it achieves the best performance. Therefore, we apply all three strategies in our proposed method.

To analyze the impact of WGAN-GP loss on model performance, we calculate the Frechet Inception Distance (FID) of the proposed method with and without WGAN-GP. FID is a criterion to evaluate the performance of GAN. The basic idea is to input the training samples and generated samples into the classifier. Then, it extracts the features of the middle layer of the classifier. Assuming that the data conforms to the multivariate Gaussian distribution, it estimates the mean values of μ_{train} and μ_{gen}, and the variance of σ_{train} and σ_{gen} of the Gaussian distribution of the generated sample and the training sample. Then, it calculates the Freche distance of two Gaussian distributions using Equation (26). The smaller the value of FID, the better the performance of GAN. We run experiments 10 times on each data set and obtain the average values. It can be seen from Table 16 that, when the WGAN-GP loss is added, the FID of our model is lower. This proves that adding WGAN-GP can improve the performance of the SACVAEGAN model to a certain extent.

$$FID = \|\mu_{train} - \mu_{gen}\| + tr(\sigma_{train} + \sigma_{gen} - 2(\sigma_{train}\sigma_{gen})^{\frac{1}{2}}) \quad (26)$$

Table 15. OA when self-attention, WGAN-GP, and additional virtual samples are removed.

Algorithms	Indian Pines	PaviaU	Salinas
SACVAEGAN without self-attention	95.37	98.52	98.38
SACVAEGAN without wgan-gp	95.61	99.01	98.67
SACVAEGAN without additional virtual samples	95.43	98.99	98.64
SACVAEGAN	95.94	99.32	98.82

Table 16. FID value of the model with or without WGAN-GP.

Algorithms	Indian Pines	PaviaU	Salinas
SACVAEGAN	10.21	1.78	2.37
SACVAEGAN without wgan-gp	12.35	12.62	11.83

6. Conclusions

In this paper, a self-attention-based conditional variational autoencoder generative adversarial network (SACVAEGAN) is proposed for hyperspectral classification. We combine the Conditional GAN with CVAEGAN, which can generate more high-quality training data to enhance the performance. Moreover, the self-attention mechanism is also applied to our proposed SACVAEGAN to extract better features. A novel loss function is used to make the whole training process more stable. Compared with the GAN-based methods, SACVAEGAN achieved a better classification performance than the state-of-the-art methods on three commonly used hyperspectral image data sets by incorporating an extra self-attention mechanism and the WGAN-GP loss. In the future, we will explore more GAN-based models for HSI classification.

Author Contributions: All authors contributed to this manuscript: Conceptualization, L.T. and Z.C.; Methodology, L.T. and Z.C.; Supervision, L.T. and C.X.; Validation, Z.C and B.Q.; Resources, Z.C. and J.Y.; Writing original draft, Z.C.; Writing review and editing, Z.C., L.T. and B.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the National Natural Science Foundation of China under Grant 61701009, 61801479 and Beijing Municipal Education Commission Science and Technology Program under Grant KM202010005016.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hong, D.; Yokoya, N.; Chanussot, J.; Zhu, X.X. An Augmented Linear Mixing Model to Address Spectral Variability for Hyperspectral Unmixing. *IEEE Trans. Image Process.* **2019**, *28*, 1923–1938. [CrossRef]
2. Gevaert, C.M.; Suomalainen, J.; Tang, J.; Kooistra, L. Generation of Spectral–Temporal Response Surfaces by Combining Multispectral Satellite and Hyperspectral UAV Imagery for Precision Agriculture Applications. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 3140–3146. [CrossRef]
3. Hunt, E.R.; Daughtry, C.S.T.; Mirsky, S.B.; Hively, W.D. Remote Sensing With Simulated Unmanned Aircraft Imagery for Precision Agriculture Applications. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 4566–4571. [CrossRef]
4. Tao, C.; Wang, Y.J.; Zou, B.; Tu, Y.L.; Jiang, X.L. Assessment and Analysis of Migrations of Heavy Metal Lead and Zinc in Soil with Hyperspectral Inversion Model. *Spectrosc. Spectr. Anal.* **2018**, *38*, 1850.
5. Wang, Q.; Yuan, Z.; Du, Q.; Li, X. GETNET: A General End-to-End 2-D CNN Framework for Hyperspectral Image Change Detection. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 3–13. [CrossRef]
6. Salah Bourennane, C.F. Robust Denoising Method based on Tensor Models Decomposition for Hyperspectral Imagery. *WSEAS Trans. Signal Process.* **2019**, *15*, 92–98.
7. Sawssen, B.; Okba, T.; Noureeddine, L. A Mammographic Images Classification Technique via the Gaussian Radial Basis Kernel ELM and KPCA. *Int. J. Appl. Math. Comput. Sci. Syst. Eng.* **2020**, *2*, 92–98.
8. Vetova, S. A Comparative Study of Image Classification Models using NN and Similarity Distance. *Int. J. Electr. Eng. Comput. Sci. (EEACS)* **2021**, *3*, 109–113.
9. Vetova, S. Covid Image Classification using Wavelet Feature Vectors and NN. *Eng. World* **2021**, *3*, 38–42.
10. Luqman Hakim, M.I.Z. Implementation of Discrete Wavelet Transform on Movement Images and Recognition by Artificial Neural Network Algorithm. *WSEAS Trans. Signal Process.* **2019**, *15*, 149–154.
11. Zhong, S.; Chang, C.I.; Zhang, Y. Iterative Support Vector Machine for Hyperspectral Image Classification. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; pp. 3309–3312.
12. Sun, S.; Zhong, P.; Xiao, H.; Liu, F.; Wang, R. An active learning method based on SVM classifier for hyperspectral images classification. In Proceedings of the 2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Tokyo, Japan, 2–5 June 2015; pp. 1–4.
13. Melgani, F.; Bruzzone, L. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Trans. Geoence Remote Sens.* **2004**, *42*, 1778–1790. [CrossRef]
14. Song, W.; Li, S.; Kang, X.; Huang, K. Hyperspectral image classification based on KNN sparse representation. In Proceedings of the 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, China, 10–15 July 2016; pp. 2411–2414.
15. Guo, Y.; Cao, H.; Han, S.; Sun, Y.; Bai, Y. Spectral–Spatial HyperspectralImage Classification with K-Nearest Neighbor and Guided Filter. *IEEE Access* **2018**, *6*, 18582–18591. [CrossRef]
16. Jia, X. Block-based maximum likelihood classification for hyperspectral remote sensing data. In Proceedings of the IGARSS'97. 1997 IEEE International Geoscience and Remote Sensing Symposium Proceedings: Remote Sensing—A Scientific Vision for Sustainable Development, Singapore, 3–8 August 1997; Volume 2, pp. 778–780
17. Ratle, F.; Camps-Valls, G.; Weston, J. Semisupervised Neural Networks for Efficient Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2010**, *48*, 2271–2282. [CrossRef]
18. Li, J.; Bioucas-Dias, J.M.; Plaza, A. Semisupervised Hyperspectral Image Classification Using Soft Sparse Multinomial Logistic Regression. *IEEE Geosci. Remote Sens. Lett.* **2013**, *10*, 318–322.
19. Khodadadzadeh, M.; Li, J.; Plaza, A.; Bioucas-Dias, J.M. A Subspace-Based Multinomial Logistic Regression for Hyperspectral Image Classification. *IEEE Geosci. Remote Sens. Lett.* **2014**, *11*, 2105–2109. [CrossRef]
20. Hughes, G. On the mean accuracy of statistical pattern recognizers. *IEEE Trans. Inf. Theory* **1968**, *14*, 55–63. [CrossRef]
21. Chen, Y.; Jiang, H.; Li, C.; Jia, X.; Ghamisi, P. Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 6232–6251. [CrossRef]
22. Hamida, A.B.; Benoit, A.; Lambert, P.; Amar, C.B. 3-D Deep Learning Approach for Remote Sensing Image Classification. *IEEE Trans. Geoence Remote Sens.* **2018**, *56*, 4420–4434. [CrossRef]
23. Chen, Y.; Wang, Y.; Gu, Y.; He, X.; Ghamisi, P.; Jia, X. Deep Learning Ensemble for Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *12*, 1882–1897. [CrossRef]
24. Chen, Y.; Lin, Z.; Zhao, X.; Wang, G.; Gu, Y. Deep Learning-Based Classification of Hyperspectral Data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *7*, 2094–2107. [CrossRef]
25. Zhang, M.; Li, W.; Du, Q. Diverse Region-Based CNN for Hyperspectral Image Classification. *IEEE Trans. Image Process.* **2018**, *27*, 2623–2634. [CrossRef]
26. Li, W.; Wu, G.; Zhang, F.; Du, Q. Hyperspectral Image Classification Using Deep Pixel-Pair Features. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 844–853. [CrossRef]

27. Xu, X.; Li, W.; Ran, Q.; Du, Q.; Gao, L.; Zhang, B. Multisource Remote Sensing Data Classification Based on Convolutional Neural Network. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 937–949. [CrossRef]
28. Xie, J.; He, N.; Fang, L.; Ghamisi, P. Multiscale Densely-Connected Fusion Networks for Hyperspectral Images Classification. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 246–259. [CrossRef]
29. Ying, L.; Haokui, Z.; Qiang, S. Spectral-Spatial Classification of Hyperspectral Imagery with 3D Convolutional Neural Network. *Remote Sens.* **2017**, *9*, 67.
30. Zhang, X.; Sun, Y.; Jiang, K.; Li, C.; Jiao, L.; Zhou, H. Spatial Sequential Recurrent Neural Network for Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 4141–4155. [CrossRef]
31. Yang, J.; Zhao, Y.; Chan, J.C.W.; Yi, C. Hyperspectral image classification using two-channel deep convolutional neural network. In Proceedings of the 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, China, 10–15 July 2016; pp. 5079–5082.
32. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
33. Wang, X.; Tan, K.; Du, Q.; Chen, Y.; Du, P. Caps-TripleGAN: GAN-Assisted CapsNet for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 7232–7245. [CrossRef]
34. Feng, J.; Yu, H.; Wang, L.; Cao, X.; Zhang, X.; Jiao, L. Classification of Hyperspectral Images Based on Multiclass Spatial–Spectral Generative Adversarial Networks. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 5329–5343. [CrossRef]
35. Zhong, Z.; Li, J.; Clausi, D.A.; Wong, A. Generative Adversarial Networks and Conditional Random Fields for Hyperspectral Image Classification. *IEEE Trans. Cybern.* **2020**, *50*, 3318–3329.
36. Zhu, L.; Chen, Y.; Ghamisi, P.; Benediktsson, J.A. Generative Adversarial Networks for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 5046–5063. [CrossRef]
37. Wang, H.; Tao, C.; Qi, J.; Li, H.; Tang, Y. Semi-Supervised Variational Generative Adversarial Networks for Hyperspectral Image Classification. In Proceedings of the IGARSS 2019—2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 28 July–2 August 2019; pp. 9792–9794.
38. Zhang, F.; Bai, J.; Zhang, J.; Xiao, Z.; Pei, C. An Optimized Training Method for GAN-Based Hyperspectral Image Classification. *IEEE Geosci. Remote Sens. Lett.* **2020**, 1–5. [CrossRef]
39. Wang, W.Y.; Li, H.C.; Deng, Y.J.; Shao, L.Y.; Lu, X.Q.; Du, Q. Generative Adversarial Capsule Network With ConvLSTM for Hyperspectral Image Classification. *IEEE Geosci. Remote Sens. Lett.* **2021**, *18*, 523–527. [CrossRef]
40. Yin, J.; Li, W.; Han, B. Hyperspectral Image Classification Based on Generative Adversarial Network with Dropblock. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 405–409.
41. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
42. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv* **2017**, arXiv:1710.10196.
43. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, C.A. Improved Training of Wasserstein GANs. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 5767–5777.
44. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic Routing Between Capsules . *arXiv* **2017**, arXiv:1710.09829.
45. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
46. Vaswani, A.; Shazeer, N.M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. *arXiv* **2017**, arXiv:1706.03762.
47. Bao, J.; Chen, D.; Wen, F.; Li, H.; Hua, G. CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2764–2773.
48. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. In Proceedings of the International Conference on Machine Learning (PMLR 2019), Beach, CA, USA, 9–15 June 2019; pp. 7354–7363.
49. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. *arXiv* **2014**, arXiv:1312.6114.
50. Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets. *arXiv* **2014**, arXiv:1411.1784.