Hindawi
Mobile Information Systems
Volume 2022, Article ID 9005552, 14 pages
https://doi.org/10.1155/2022/9005552

Research Article

Application of an Improved DCGAN for Image Generation

Bingqi Liu,[1,2] **Jiwei Lv**,[2] **Xinyue Fan**,[2] **Jie Luo**,[2] **and Tianyi Zou**[2]

[1]*School of Mechanical Engineering, Chengdu University, Chengdu 610106, China*
[2]*Geomathematics Key Laboratory of Sichuan Province, Chengdu University of Technology, Chengdu 610059, China*

Correspondence should be addressed to Jiwei Lv; lvjiwei_cd@163.com

Received 3 March 2022; Revised 16 June 2022; Accepted 21 June 2022; Published 18 July 2022

Academic Editor: Manoj Kumar

With the rapid development of deep learning, image generation technology has become one of the current hot research areas. A deep convolutional generative adversarial network (DCGAN) can better adapt to complex image distributions than other methods. In this paper, based on a traditional generative adversarial networks (GANs) image generation model, first, the fully connected layer of the DCGAN is further improved. To solve the problem of gradient disappearance in GANs, the activation functions of all layers of the discriminator are LeakyReLU functions, the output layer of the generator uses the Tanh activation function, and the other layers use ReLU. Second, the improved DCGAN model is verified on the MNIST dataset, and simple initial fraction (ISs) and complex initial fraction (ISc) indexes are established from the two aspects of image quality and image generation diversity, respectively. Finally, through a comparison of the two groups of experiments, it is found that the quality of images generated by the DCGAN model constructed in this paper is 2.02 times higher than that of the GANs model, and the diversity of the images generated by the DCGAN is 1.55 times higher than that of GANs. The results show that the improved DCGAN model can solve the problem of low-quality images being generated by the GANs and achieve good results.

1. Introduction

With the introduction of the concepts of cloud computing and big data and the rapid development of computer hardware facilities, deep learning has undergone rapid development and has been used in many applications in recent years [1]. However, the development of image generation technology is slow in several branches of deep learning. Before GANs were proposed, the main image generators were automatic regression models [2] and variational autoencoders [3]. At the same time, based on the improvements in GANs theory, GANs have been applied in image conversion, image feature extraction, and other fields.

There are many models used in image generation and modeling research, including BPT-CNN [4], the BERT-based deep spatial-temporal network [5], GeNet of deep convolutional neural network [6], and RNN-LSTM [4]. However, as a new type of image generation model, GANs have attracted the attention of many researchers, who have gradually improved and provided a large number of mature image generation frameworks (such as DCGAN, CGAN, Pix2Pix, etc.). In terms of theoretical research on GANs, in 2014, Goodfellow et al. [7] first described a new image generation model, the GANs, which is composed of a generator and a discriminator [7]. In the same year, Mirza and Osindero [8] were inspired by the introduction of convolutional neural networks on the basis of GANs and proposed the CGAN, which solved the unstable training behavior problem of GANs by adding category labels [8]. To solve the instability of GANs, in 2016, Goodfellow et al. [7] and Salimans et al. [9] proposed stabilizing the training process of DCGAN with feature matching, small batch recognition, and historical averaging, and this work provided a basis for follow-up research [9]. With regard to the applications of GANs, Isola et al. [10] implemented image conversion using Pix2Pix and paired training data [10]; Zhang et al. [11] proposed StackGAN, which first generates basic images and text descriptions based on the original image information and then improved the process to generate high-resolution images [11]. In 2017, Zhu et al. [12] proposed CycleGAN, which solved the problem of Pix2Pix needing paired data and proposed the cycle-consistency loss function to realize image conversion from a horse to a zebra [12]. Karras et al. [13] proposed StyleGAN in 2018 to accelerate and stabilize the training speed of the network by gradually increasing the numbers of generators and discriminators [13]. This method uses natural style-conversion technology, such as adaptive instance normalization (AdaIN), for reference purposes and realizes the real-time transmission of any style [14]. BigGAN was proposed by Andrew Brock in 2019. BigGAN adopts a self-attention mechanism and spectral normalization, and it is a good model for image generation on ImageNet at present [15]. In summary, there have been some comprehensive methods proposed to solve the problems of GANs generation and resolution in recent years.

Alec Radford et al. [16] used the CNN structure [17, 18] to implement the GANs model for the first time and proposed the DCGAN [16]. Liu et al. compared the unconstrained DCGAN and the constrained DCGAN, and the results showed that after adding constraints during the training phase, the DCGAN model significantly improved upon the results of the virtual face generation model, thus demonstrating the enhanced ability of the generator and discriminator [19]. Mahmoud and Guo [20] used the DCGAN to extract depth features for strongly representing TSR images [20]. Fang et al. [21] proposed a new gesture recognition algorithm based on a convolutional neural network and the DCGAN and applied this method to expression recognition, calculation, and text output, achieving good results in all cases [21]. The actual image and noise vector of the DCGAN were trained, and smoke image training was used to generate a discriminator, thereby showing that the DCGAN can effectively monitor smoke images [22]. The biggest differences between the DCGAN and original GANs are that the DCGAN uses a convolutional neural network (CNN) to replace the multilayer perceptron in the original GANs, removes the pooling layer, and uses a convolution with a defined step size to replace the upper sampling layer for improving the stability of the training process.

To solve the problem of the easily disappearing gradient in GANs, this paper further improves the DCGAN fully connected layer. For the activation functions of all layers of the discriminator, generator output, and other layers, LeakyReLU, Tanh, and ReLU functions are used, respectively, and the open source MNIST dataset is used. For verification, we use the improved DCGAN and GANs for comparison and establish ISs and ISc from the two aspects of image quality and image generation, respectively. According to the numerical value, it can be concluded that the improved DCGAN has better image processing ability. The remainder of the paper is organized as follows: Section 1 summarizes the progress of research with regard to GANs and the DCGAN; Section 2 mainly introduces the principles of the improved DCGAN algorithm and designs the network structure; Section 3 constructs the image generation models, with one based on GANs and the other based on the DCGAN; Section 4 introduces two image generation quality evaluation methods and analyzes the effects of the two models on image generation quality and image diversity. The study's discussion and conclusions are presented in Section 5.

2. Improved Design of the Structure of the DCGAN

2.1. Principles of the DCGAN Algorithm. The adversarial training process of the DCGAN model established in this paper was calculated by using

$$\min_{G}\max_{D}{}^{V}(D,G) = \mathrm{E}_{x\sim P_{\text{data}(x)}}\log[D(x)] + \mathrm{E}_{Z\sim P_{Z(Z)}}[\log(1-D(G(z)))], \tag{1}$$

where x is the distribution of the real data, x is the sample image data, P_Z represents arbitrarily distributed noise, Z expresses the number of random vectors in P_Z, and E- expresses expectations.

The first step is to find the minimum cross-entropy of the discriminator D under the condition where a generator G is given. The objective function was calculated by using

$$\mathrm{Obj}(D) = -\mathrm{E}_{x\sim P_{\text{data}}}\log[D(x)] - \mathrm{E}_{Z\sim P_Z}\log[1-D(G(Z))], \tag{2}$$

where $\log[D(x)]$ is used to judge the sample data, $\log[1-D(G(Z))]$ represents the judgment of the generated sample data, that is, the closeness of the distribution of the sample data output by the discriminator P_{data} and the data distribution generated by $GP_{G(x)}$, and x is a sample from the real data, according to

$$\begin{aligned}\mathrm{Obj}(\theta_D,\theta_G) &= -\int_x P_{\text{data}}(x)\log(D(x))dx \\ &\quad - \int_Z P_Z(Z)\log(1-D(G(Z)))dz \\ &= \int_x [P_{\text{data}}\log(D(x)) + P_G(x)\log(1-D(x))]dx.\end{aligned} \tag{3}$$

At this time, because the data and generator have been given, they can be regarded as constants. Assuming that the data and generator are replaced, then

$$f(D) = c_1\log D + c_2\log(1-D). \tag{4}$$

Let $f(D)=0$ in (1); then it is the maximum point:

$$D*(x) = \frac{P_{\text{data}}(x)}{P_{\text{data}}(x)+P_G(x)}. \tag{5}$$

The second step is to fix the discriminator D. At this time, the optimization function for the generator G was calculated by using

$$V(G,D) = \mathrm{E}_{x\sim P_{\text{data}}}\log D(x) + \mathrm{E}_{x\sim P_G}\log[1-D(x)]. \tag{6}$$

Furthermore, using (2), $D*$ brings $V(G,D)$ as the optimal solution of the generator, which can be calculated by using

$$\min_G V(G,D) = V(G,D*)$$

$$= \mathrm{E}_{x\sim P_{\mathrm{data}}}(x)\left[\log\frac{P_{\mathrm{data}}(x)}{P_{\mathrm{data}}(x)+P_G(x)}\right] \tag{7}$$

$$+ \mathrm{E}_{x\sim P_G}(x)\left[\log\frac{P_{\mathrm{data}}(x)}{P_{\mathrm{data}}(x)+P_G(x)}\right].$$

In practical training, the discriminator D is usually trained first. Then, the discriminator D is fixed and the generator G is trained. Next, we continue to fix G and train the discriminator D, performing iterative optimization training until $P_{\mathrm{data}} = P_G$, at which point global optimization is achieved.

2.2. Design of the Structure of the DCGAN. Compared with traditional GANs, the salient feature of the DCGAN is that a CNN is used to replace the multilayer perceptron. The pooling layer and sampling layer are removed in the CNN model. The convolution layer is used to discriminate the image in the discriminator, and the deconvolution layer is used to generate the image in the generator. The specific structure of the DCGAN generator is as follows: the input layer is followed by a batch normalization layer (which can hasten the convergence of the model), and the reshaping layer is used to normalize the preliminary data; then, an upsampling layer, a Conv2DTranspose layer, and a batch normalization layer are used to sample, deconvolute, and normalize the data, respectively. In this paper, the DCGAN adds three groups of the above structures to increase the depth of the network. The main framework of the network architecture of the generator is shown in Figure 1.

In this paper, to solve the problem in which gradients disappear easily, the LeakyReLU activation function is used in all layers of the discriminator, and the Tan*h* activation function is used in the output layer of the generator, where definition of this function is

$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}. \tag{8}$$

For the generator, except for the activation function of the last layer, the ReLU activation function is used, and its definition is

$$f(x) = \begin{cases} x, & x \geq 0, \\ 0, & x < 0. \end{cases} \tag{9}$$

In addition, the concrete structure of the DCGAN discriminator is a Conv2D layer (2D convolution layer), a batch normalization layer, and a dropout layer (after the image is convoluted, normalization process is continued, and the dropout layer is added to increase the generalization ability of the model). These three layers form a group, and four groups are added. Finally, a flattening layer and a fully connected layer are used to flatten the data and output the probability of whether it is sample data or generated data. Except for that of the last layer, the activation function of the other layers is the LeakyReLU function. The definition of this function is

$$f(x) = \begin{cases} x, & x \geq 0, \\ ax, & x < 0, \end{cases} \tag{10}$$

where x is a sample from the real data and a is the relevant parameter.

The activation function of the last layer adopts the sigmoid function. The definition of this function is

$$б(x) = \frac{1}{1+e^{-x}}. \tag{11}$$

3. Construction of the Image Generation Models Based on the DCGAN and GANS

3.1. Data Sources. This paper uses the MNIST dataset, which is free of charge, is open source, contains small pixels, and includes a large number of points [24]. It is composed of 250 handwritten digits (0–9, a total of 10 digits); it is relatively mature in image processing and image quality processing as shown in Figure 2.

To universalize the dataset, 50% of the data in the dataset are from high school students, and 50% are from Census Bureau staff. At the same time, this paper uses the Keras framework with Tensorflow as the back end. The Keras framework is an open source artificial neural network library written in Python. The code structure is written with an object-oriented method, which is completely modular and extensible. It is suitable for the implementation framework of the code in this experiment.

3.2. Determination of the DCGAN Model Parameters. The collected basic data is used in the DCGAN model for experimental research, and the network parameters are determined through continuous testing. The situation is as follows: the model generator accepts 1×100 random, normally distributed noise data. Considering that the layout of the pixels of the image generated by deconvolution is 28×28 (i.e., MNIST dataset image pixels), the input layer is set as a fully connected layer, and the number of neurons is $7\times7\times256$. After that, a batch normalization layer and reshaping layer are added, and then the size of the data pixel is expanded through the upsampling layer. Using a 5×5 convolution kernel, the conv2dspread layer uses the "same" border model to preserve the convolution results at the boundary, so that the input and output dimensions are the same. The upsampling layer is added after the first module to make the pixel size of the output image 28×28. The activation function in this layer uses the ReLU function, and the last layer uses the Tanh function.

In the discriminator, a Conv2D layer (2D convolution layer), a batch normalization layer, and a dropout layer are added as a module, and four modules are added. Finally, a flattening layer and a fully connected layer are used as the back end. The Conv2D layer uses a 5×5 convolution kernel, the boundary mode of the convolution layer is the same as that in the generator, and the activation function is the

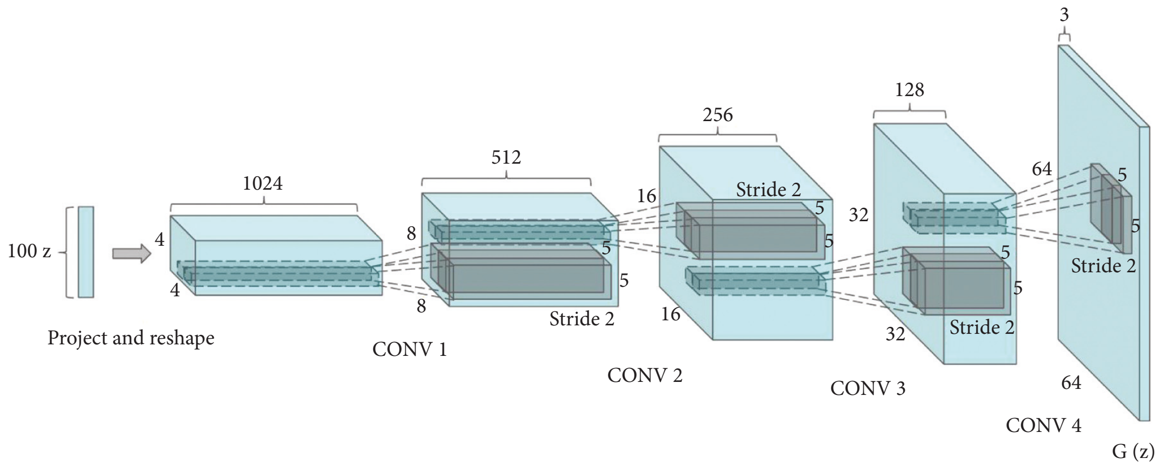


Figure 1: Network structure of the DCGAN generator [23].

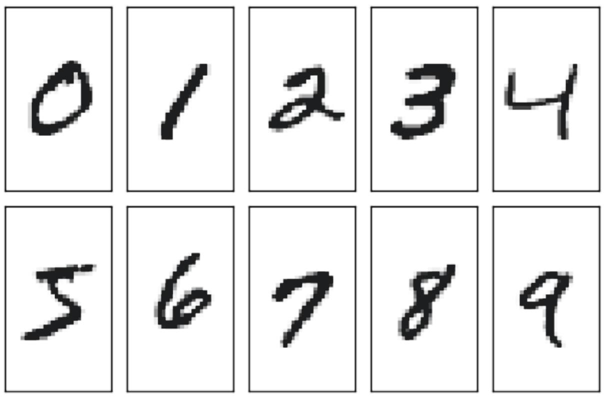


Figure 2: Example from the MNIST dataset.

Table 1: Main parameters of the DCGAN.

Parameter	Value
BATCHSIZE	32
EPOCH	100,000
LEARNINGRATE (GANs)	0.0001
DECAY (GANs)	0.00000003
LEARNINGRATE (Discriminator)	0.0002
DECAY (Discriminator)	0.00000006
ALPHA (LeakyReLU)	0.2
MOMENTUM (Batch normalization)	0.9
DROPOUT	0.3
STRIDES (Conv2D)	2
OUTPUTWIDTH	28
OUTPUTHIGHT	28
OUTPUTCHANNEL	1
LATENTSIZE	**100**

Table 2: Main parameters of the GANs.

Parameter	Value
BATCHSIZE	32
EPOCH	100,000
LEARNINGRATE (GANs)	0.0002
LEARNINGRATE (Discriminator)	0.0002
DECAY (Discriminator)	0.000000009
ALPHA (LeakyReLU)	0.2
MOMENTUM (batch normalization)	0.8
OUTPUTWIDTH	28
OUTPUTHIGHT	28
OUTPUTCHANNEL	1
LATENTSIZE	100

LeakyReLU function. The dropout layer in the module makes the activation value of a certain neuron have a certain probability p when it propagates forward. This can make the model independent of some local features and enhance the generalization ability of the model. The last activation function uses the sigmoid function, which can output the probability of discrimination, that is, the probability that the discriminator thinks the image belongs is the real image and not a generated image.

The discriminator uses the Adam optimizer with a learning rate of 0.0002, and the GANs use the RMSprop optimizer with a learning rate of 0.0001. The RMSprop optimizer combines the exponential moving average of the square of the gradient to adjust the learning rate. It can converge effectively under an unstable objective function and yields good results with the DCGAN model. The batch size is 32, and the training time of each DCGAN iteration is 10000. The binary cross-entropy function is selected as the loss function.

The training process of the DCGAN is slightly different from that of the GANs because it takes more time to train the discriminator of the DCGAN model. First, when training the discriminator, the input data size is $2 \times$ batch, where the input data contains the real data and generated data from one batch; second, the combined data of size $2 \times$ batch are used as the input data to train the discriminator 5 times; finally, the whole GAN model is trained once with the random noise data from one batch as the input data, and only the generator is updated. A complete training cycle is a batch (epoch) in which the ratio of discriminator training iterations to generator training iterations is 1 : 5 to realize the alternating iterative training process. The main parameter configuration of the DCGAN is shown in Table 1.

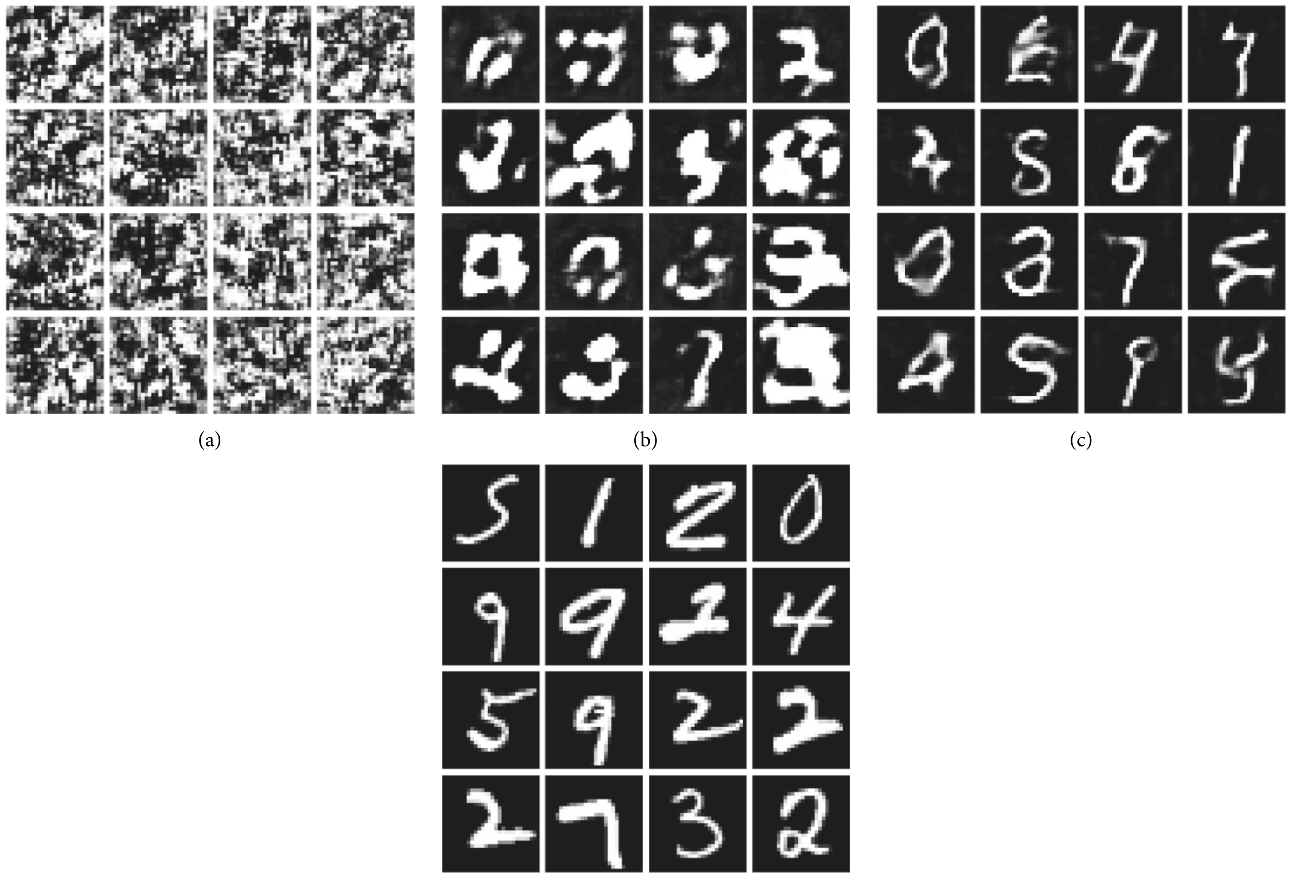


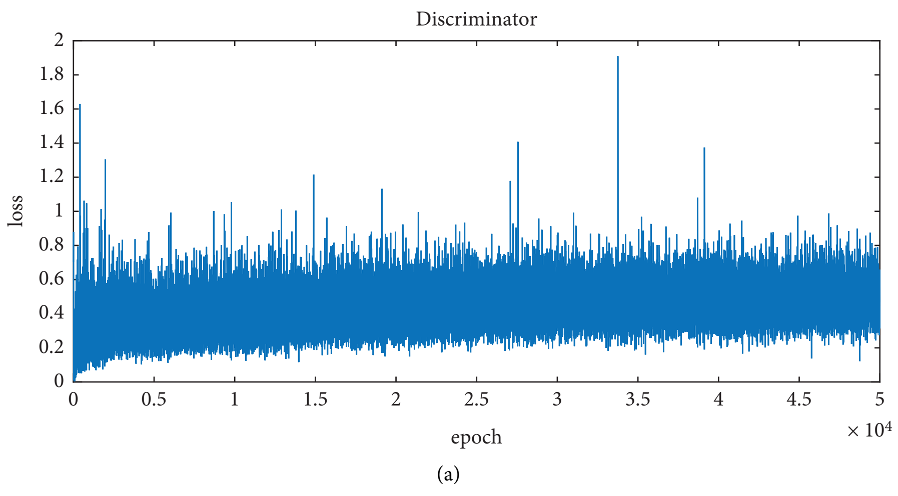
Figure 3: DCGAN partial results (Experiment 1). (a) Training 0 times. (b) Training 1000 times. (c) Training 5000 times. (d) Training 10000 times.

3.3. Determination of the Parameters of the GAN Model. The noise data received by the generator are the same as above, and a basic multilayer perceptron is used to add these three modules as a fully connected layer, a LeakyReLU layer, and a batch normalization layer; the model ends with a fully connected layer and a reshaping layer. The activation function of the last layer uses the Tan*h* function.

The discriminator uses a flattening layer to flatten the data and then adds two fully connected layers. The activation function also uses the LeakyReLU function, and the last activation function uses the sigmoid function to output the discrimination probability. Both the GANs and the discriminator use Adam as their optimizer, and the learning rate is 0.002. To calculate the update step size, the Adam optimizer comprehensively considers the first-order moment estimation (average value of the gradient) and second-order moment estimation (noncentral variance of the gradient). The batch size is 32. Due to the slow convergence speeds of GANs, the model can avoid falling into local optimal solutions and undergo training 100K times iteratively. The binary cross-entropy function is selected as the loss function. The main parameter configuration of the GANs is shown in Table 2.

During training, the discriminator is trained once, and the input data are half true and half false; i.e., the input dataset is composed of half real data and half batch-generated data. Such a complete combined dataset is used as input data to train the discriminator once. Then, the whole GAN model is trained once with a batch of random noise data as input, and only the generator is updated. Such a training cycle is a batch (epoch), which is performed to realize the alternating iterative training process.

3.4. Experimental Results of the Image Generation Models Based on the GANs and DCGAN. There are experimental groups in the training experiments of the image generation models based on the GANs and DCGAN. Each experimental group is divided into Experiment 1 (composed of 250 handwritten digits (0–9, a total of 10 digits)) and Experiment 2 (only using the number "6" in the dataset). The experiment under the unified experimental group is conducted to observe the learning effects of different networks with different image distribution complexity and image generation quality; the experiment with different experimental groups using the same dataset is done to compare the advantages and disadvantages of the GANs and DCGAN with regard to image generation. The experiments covered in this article require the use of the following: CPU frequency: 2.5 GHz, memory capacity: 16 GB, graphics chip: NVIDIA GeForce RTX 3070, and hard disk capacity: 512 GB.



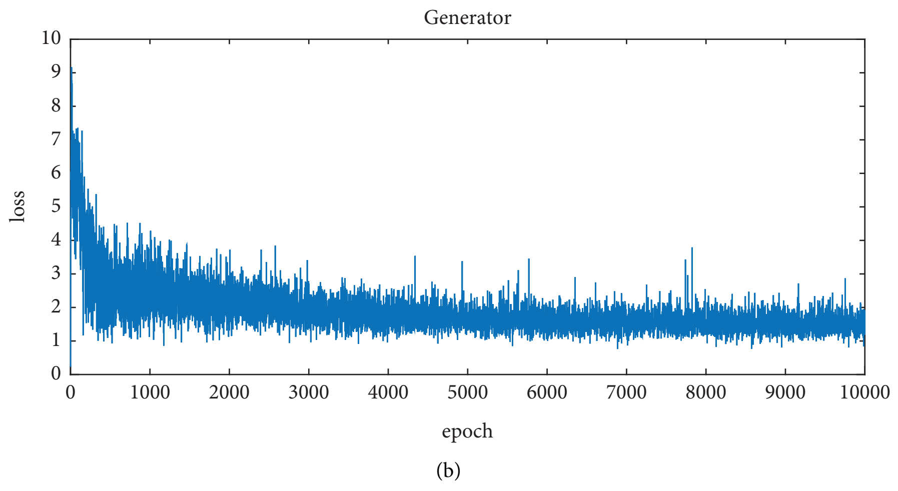


Figure 4: Loss curves of the DCGAN discriminator (a). Generator (b) (Experiment 1).

3.4.1. Training Results of the Image Generation Model Based on the DCGAN

(1) Experiment 1. Set checkpoints in the training process, run the training process for 10 h in the local environment, and observe the output results once every 50 training iterations. The results are shown in Figure 3, in which Figures 3(a)–3(d) are the results of the original noise image, the results after 1K training iterations, the results after 5K training iterations, and the results after 10K training iterations, respectively. By observing the output at each checkpoint, we can find that the original noise data are disordered. After 1K iterations, the image is gradually regionalized, and the only content is in the central area. However, most of the digital contours cannot be clearly recognized, and obvious characteristics of the deconvolution layer can be found. The image learning process includes regionalization and characterization rather than pixel learning, similar to the process of the GANs. Training 5K times can yield a gradually clearer line for which the numbers can be identified but not recognized clearly. After training 10K times, each number can be clearly identified, clear images are generated, and the complex image distribution is successfully fitted.

From the loss images of the discriminator (Figure 4(a)) and the generator (Figure 4(b)), it can be seen that the discriminator stabilizes at approximately 0.5 for a very short batch. For the 1K batch, the loss of the generator decreases to 2, then approaches 1 slowly, and finally stabilizes at approximately 1.5, but there is a downward trend. Because of this configuration, the experiment cannot continue; even if the Nash equilibrium point is not reached, the effect of the output image is still very good, and this shows that the DCGAN can obtain better experimental results than those of other methods under the premise of satisfying the computational power requirements.

(2) Experiment 2. Run the same training process as in Experiment 1 in the local environment for 10 h; the results are shown in Figure 5, in which Figures 5(a)–5(d) are the results of original noise image, the results after training 1K times, the results after training 5K times, and the results after training 10K times, respectively. By observing the output of each checkpoint, it can be found that the original noise is the same as in Experiment 1. When iterating 1K times, the number “6” can be identified, such as the first number from

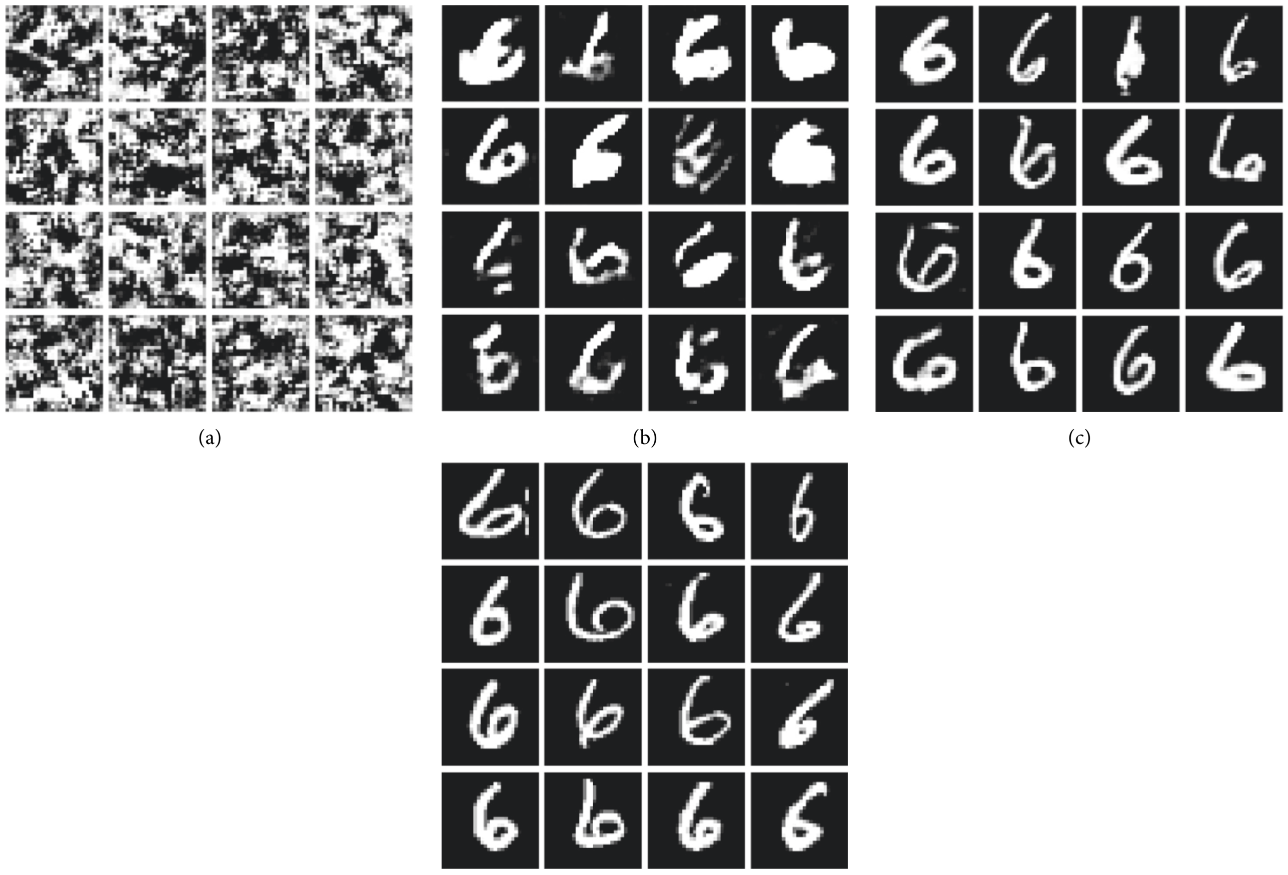

(a)
(b)
(c)
(d)

Figure 5: Partial results of the DCGAN (Experiment 2). (a) Training 0 times. (b) Training 1000 times. (c) Training 5000 times. (d) Training 10000 times.

the left in the second row and the fourth number from the left in the third row; most of the number "6" can be recognized after iterating 5K times. All the numbers can be recognized after 10K iterations, but the numbers are not standardized, mainly because the MNIST dataset contains handwritten numbers from adults and children.

According to the loss images of the discriminator (Figure 6(a)) and generator (Figure 6(b)), the discriminator *D* starts to fluctuate at approximately 0.5 at 1000 epochs (the ratio of the training times of the discriminator and generator is 5 : 1); the generator also starts to stabilize at approximately 2 at 1K epochs, gradually converges to 1, and finally fluctuates at approximately 1 with a small fluctuation range.

Through the comparative training processes of the two experiments, it is found that the DCGAN model can adapt to both complex image distributions and simple image distributions, and it can converge earlier than other methods. Its learning law is characterized and regionalized. At the beginning, it concentrates on the central area, then learns the line features, and finally learns the location characteristics of the lines.

3.4.2. Training Results of the Image Generation Model Based on GANs

(1) Experiment 1. Set checkpoints in the training process, run the training process for 5 h in the experimental environment, and observe the output results once every 500 training iterations. The results are shown in Figure 7, in which Figures 7(a)–7(d) are the results of the original noise image, the results after training 10K times, the results after training 50K times, and the results after training 100K times, respectively. By observing the output at each checkpoint, we can find that the original noise data are disordered. After 10K iterations, the image is gradually focused in the central area rather than at scattered points in each position. After iterating 50K times, some figures have a preliminary outline, indicating that the generator is gradually learning the image distribution of the original dataset. Continuing to train for a total of 100K iterations, one can find that most of the figures are clear and distinguishable. This situation lasts nearly 20K batches during the training process, indicating that the GANs maintain stability for a long time but only reach the

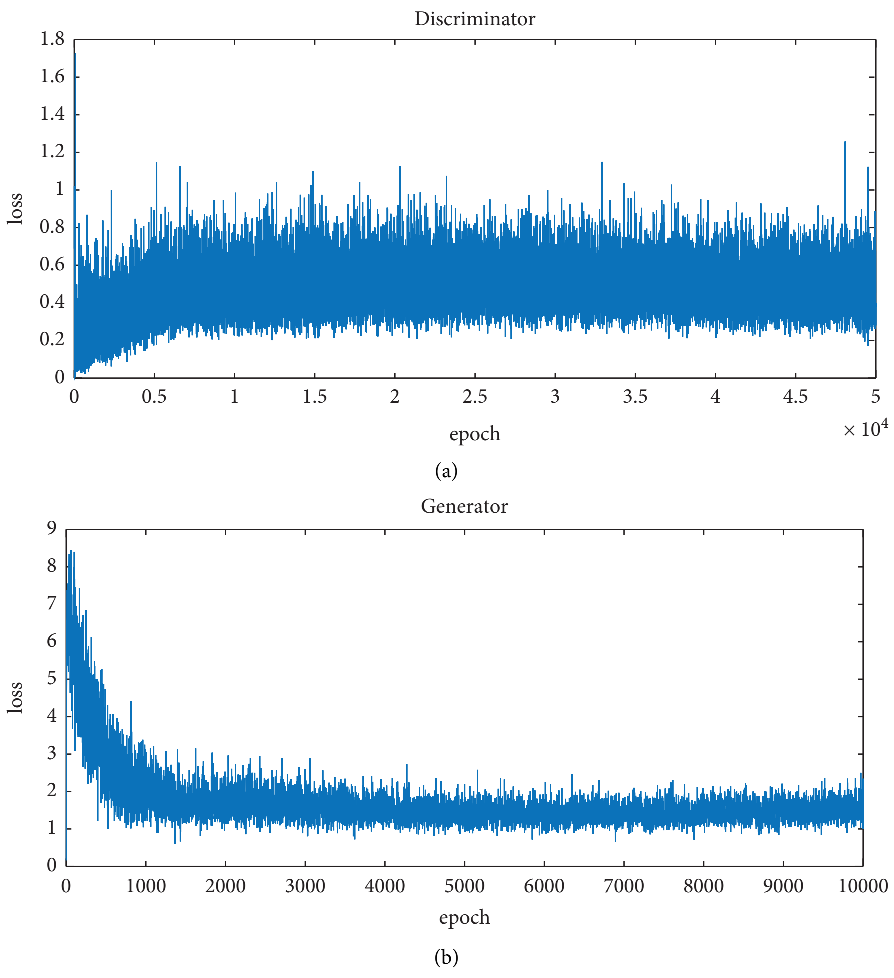


Figure 6: Loss curves of the DCGAN discriminator (a). Generator (b) (Experiment 2).

pseudo-Nash equilibrium point. It is found that adding a hidden layer does not affect the experimental results but rather increases the experimental time.

By observing the loss function images of the discriminator (Figure 8(a)) and generator (Figure 8(b)), it is not difficult to see that the loss of the generator decreases rapidly in each of the first 10K batches, then gradually approaches 1, fluctuates greatly between 30K and 70K iterations, and finally stabilizes near 1. This shows that the image generated by the generator can make the discriminator think that it is true. However, the discriminator gradually becomes stable at approximately 0.5 starting with the 15000th batch, and then it fluctuates up and down. As the number of batches increases, the fluctuation range does not decrease, and the overall loss is slightly less than 0.5, indicating that the discriminator has a high probability of correct discrimination (recognition of the real image). After 100 K iterations, the Nash equilibrium point cannot be reached, but the model cannot be further converged.

(2) Experiment 2. The running time of the training process is 5 h in the local environment, and the results are shown in Figure 9, where Figures 9(a)–9(d) are the results of the original noise image, the results after training 10K times, the results after training 50K times, and the results after training 100K times, respectively. The original noise is the same as in Experiment 1, and the learning process of the GANs can still be seen in (b) and (c), but the speed of the pixel setting is much higher than that in Experiment 1. Compared with Figures 9(d) and 7(d) of Experiment 1, it can be found that when iterating 100K times, GANs can effectively fit the simple image distribution. The second digit on the left in the second row of Figure 7(d) generates "6", but this is not as effective as simply learning "6". In Figure 9(d) of Experiment 2, all the numbers "6" can be clearly identified.

Comparing the loss function images of the discriminator (Figure 10(a)) and generator (Figure 10(b)), it is not difficult to find that the generator loss rapidly decreases in each of the first 20K batches, gradually approaches 1, and stabilizes near 1. This shows that the image gradually generated by the generator can make the discriminator think it is true. Since the 20000th batch, the discriminator gradually stabilizes at approximately 0.5 and gradually reduces its fluctuation range, but the overall loss is slightly higher than 0.5, which

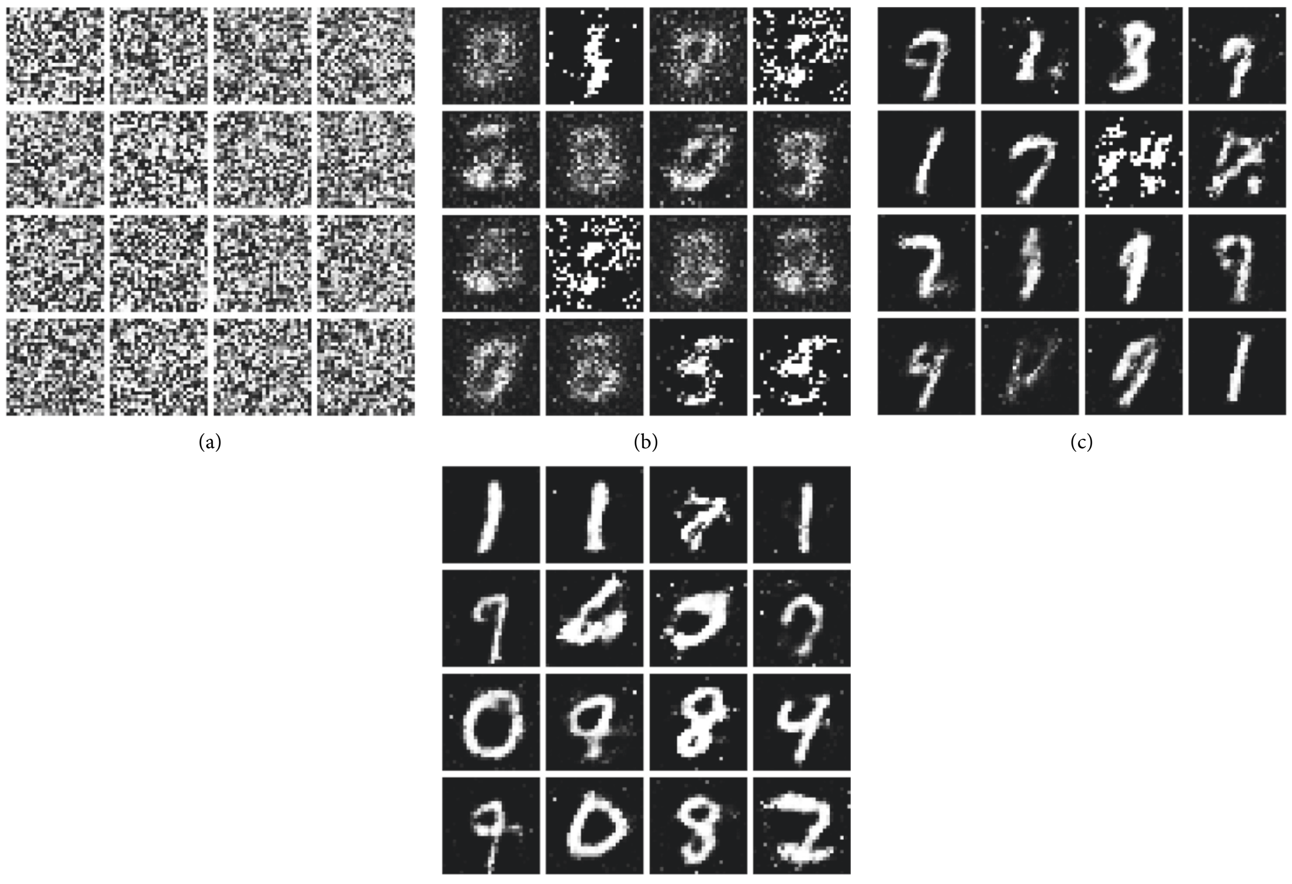

(a)
(b)
(c)
(d)

Figure 7: Partial results of the GANs (Experiment 1). (a) Training 0 times. (b) Training 10000 times. (c) Training 50000 times. (d) Training 100000 times.

indicates that the discriminator cannot correctly judge that the generated image is real, and the loss curve still cannot converge at 0.5; that is, it cannot reach the real Nash equilibrium point.

4. Comparative Analysis and Assessment of Image Generation Quality

4.1. Construction of the Evaluation Index for Image Generation. When comparing and analyzing the quality of images generated by models, two factors are generally considered: one is the textures of the images, and the other is the diversity of image generation. At present, the popular quantitative evaluation method for image texture is the inception score (IS). In this paper, the simple initial fraction (ISs) is used to evaluate image quality, and the complex initial fraction (ISc) is used to evaluate image diversity.

4.2. Image Quality Assessment. The calculation formula of the ISs evaluation index using the simple initial score method is

$$IS_s(G) = \exp\left(\mathrm{E}_{x\sim p_G} D_{KL}\left(p(y|x) \| p(y)\right)\right), \tag{12}$$

where $x \sim p_G$ represents the image generated by the generator to be evaluated, $p(y|x)$ represents the probability that the picture belongs to each category, and $p(y)$ is the probability distribution of the image to be evaluated.

The following formula can be obtained by further derivation:

$$\mathrm{In}\left(IS_s(G)\right) = H(y) - H(y|x). \tag{13}$$

It can be seen from (13) that the larger the ISs evaluation index, the greater the differences between the $p(y|x)$ and $p(y)$ distributions.

If only the ISs comprehensive analysis method is used, there may be a large error. In this paper, a concept classification network is proposed to eliminate the error as much as possible. In the MNIST dataset, there are 10 numbers from 0 to 9, and each image is a black and white image (the value is composed of 0s and 1s). For any two random variables X, Y, the correlation coefficient is calculated as follows:

$$\tau(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}[X]\mathrm{Var}[Y]}}, \tag{14}$$

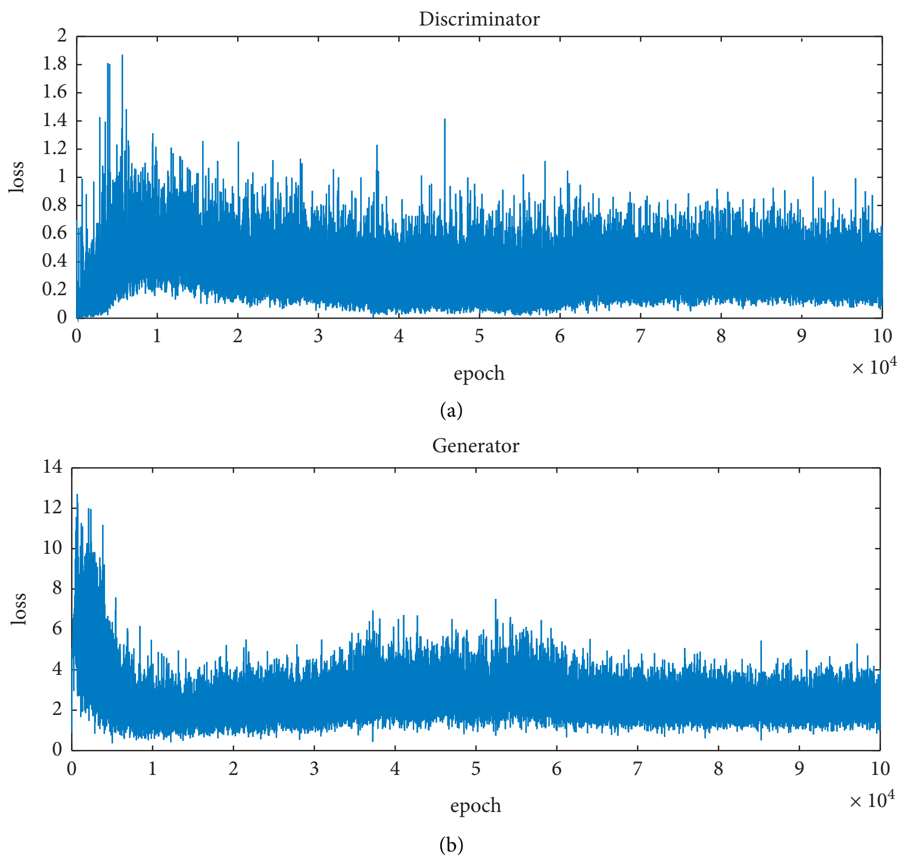


Figure 8: Loss curves of the GANs discriminator (a). Generator (b) (Experiment 1).

where $\mathrm{Cov}(X,Y)$ in (14) is the covariance of X and Y, $\mathrm{Var}[X]$ is the variance of X, and $\mathrm{Var}[Y]$ is the variance of Y.

According to the principle of the concept classification network, it is not difficult to know that if the image correlation is weak, the classifier easily divides it into two categories; otherwise, it is easier to divide it into one category. When considering the diversity of images, we should consider the distribution of the labels. If the distribution of complex images is complex, we naturally want the labels to be evenly distributed instead of generating a certain kind of image. For example, when all the numbers in the MNIST dataset are used, a good situation is that the generated 0 s–9 s are evenly distributed, rather than the distribution containing more of certain numbers than others. In this case, we need to consider the edge probability $p(y)$. In the ideal state, the expansion can yield $p(y_1) = p(y_2) = \cdots = p(y_n) = 1/n$, where n is the number of classes in the original training data; the greater the entropy of $p(y)$, the better the situation.

If the same kind of data is used, that is, a single number is used, although these data would be highly correlated and belong to the same class, because of the characteristics of the concept classification network, the set would still be divided into several categories, but $\lim p(y_1) = 1$ and $\lim p(y_i) = 0$ where $i = 2, 3, \ldots, n$. At this time, n is the number of classification categories for the concept classification network.

4.3. Image Diversity Assessment. The complex initial fraction (ISc) is based on the simple initial fraction method. According to the image classification value of the simple distribution, the entropy of the simple image distribution under discrete conditions is calculated, where H in (15) represents entropy, and the calculation formula is as follows:

$$H(p(y_i)) = \sum_{i=1}^{n} p(y_i) \times \log(p(y_i)). \tag{15}$$

The edge probability value of the simple image distribution is included when $i = 1$; using (16), we calculated the potential:

$$H(p(y_i)) = H(p(y_1)) = 0. \tag{16}$$

When $i = 2, 3, \ldots, n$, $\lim p(y_i) = 0^+$:

$$H(p(y_i)) = \sum_{i=1}^{n} p(y_i) \times \log(p(y_i)) = 0. \tag{17}$$

Thus, the entropy limit of the concept classification network is obtained when the image tends to be simple distribution, i.e., when $\lim H(p(y)) = 0$ in (17). Then, the formula of the ISc evaluation index becomes

$$\mathrm{In}(IS_C(G)) = H(y|x), \tag{18}$$

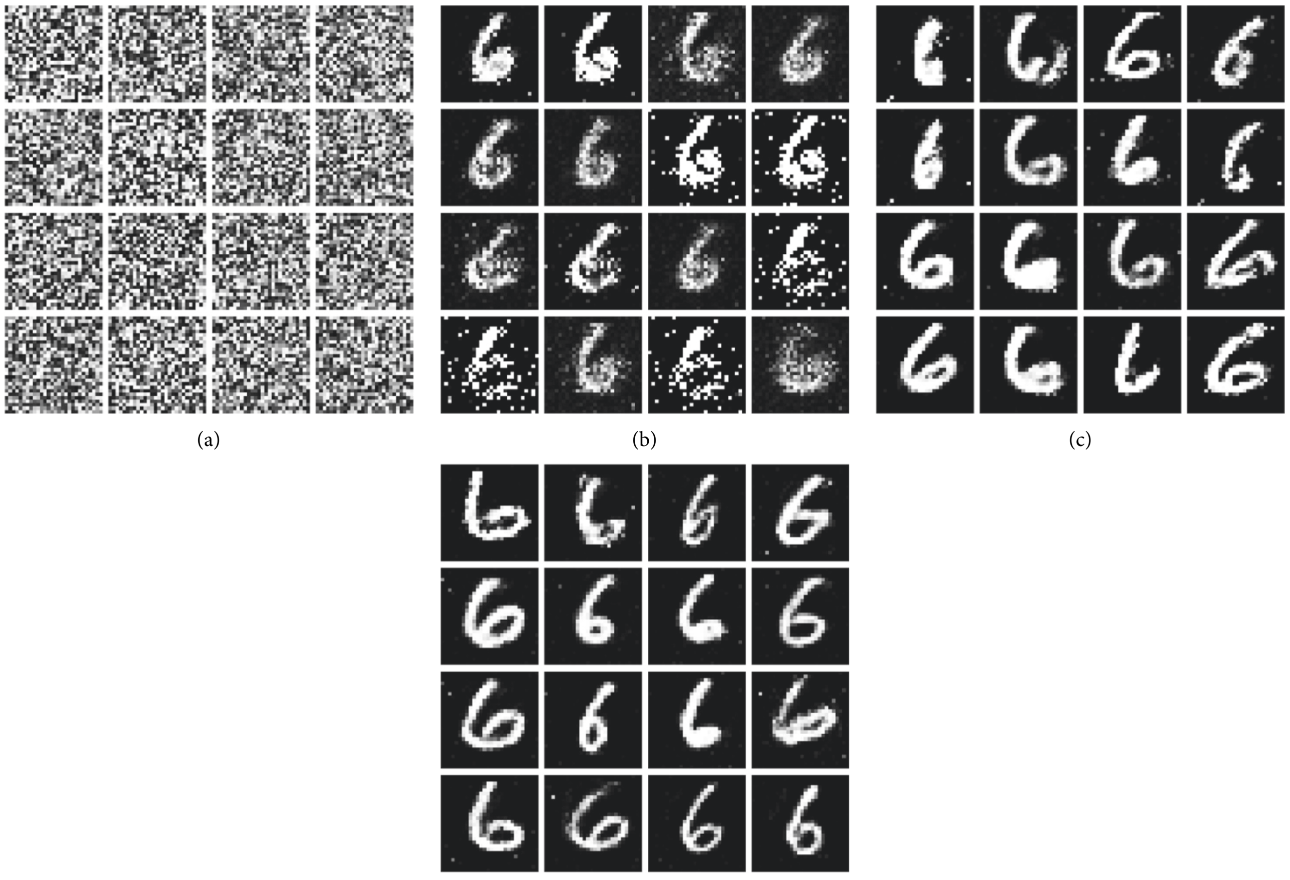
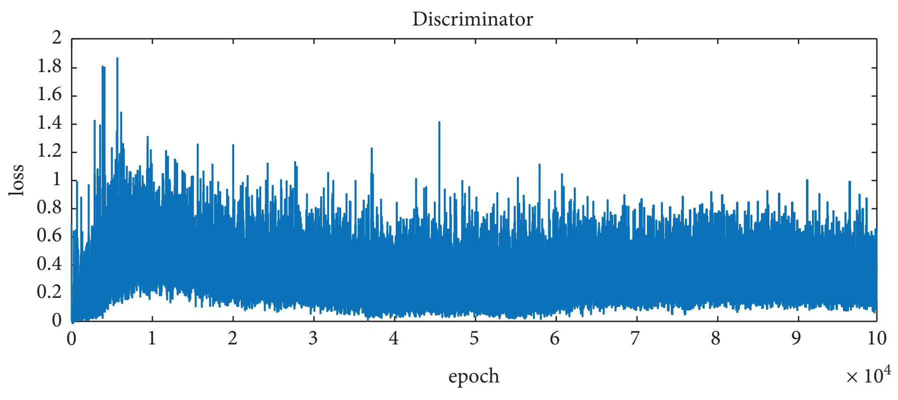


Figure 9: Partial results of the GANs (Experiment 2). (a) Training 0 times. (b) Training 10000 times. (c) Training 50000 times. (d) Training 100000 times.

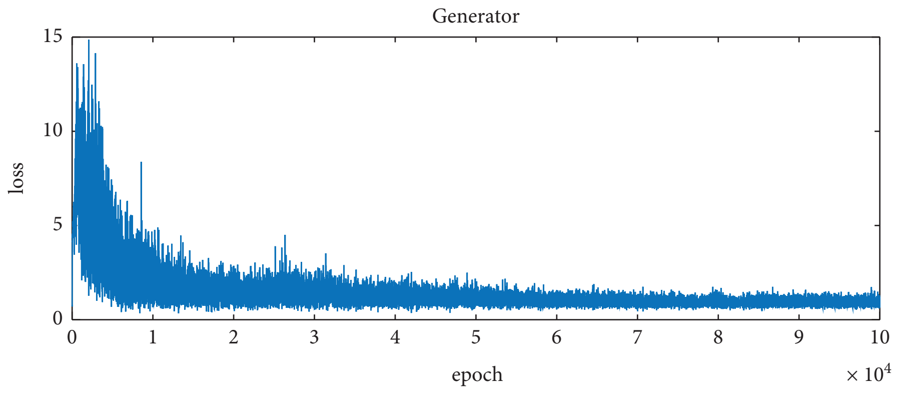
where $H(y|x)$ in (18) represents the probability that an image belongs to a certain category. The higher the value, the higher the image quality. Therefore, when the image distribution is simple, the ISc under a complex distribution can be used to comprehensively evaluate the diversity of the images. The diversity of image generation can be combined with two experiments and a comprehensive ISc analysis. The two groups of experiments discuss image generation under different complexities of the image distribution. If the ISc value can still reach a high value under the condition of a complex image distribution, this shows that the diversity of image generation is very high.

4.4. Comparative Analysis of Experimental Results. According to the image quality evaluation method proposed in Section 4.1, two groups of experiments are compared and analyzed. For the MNIST dataset, first, 10K groups of images are generated by the generator, all data are divided into 10 pieces, and each piece is calculated and averaged. Second, the concept classification network is built to calculate the ISc and ISs of the experimental GANs and the DCGAN, respectively. The experimental results are shown in Table 3. The first experiment is conducted to evaluate the image diversity, and the second is performed to evaluate the image quality.

According to the experimental results in Table 3, in terms of image quality, the ISs value of the DCGAN model is 2.02 higher than that based on the GANs model; in terms of image generation diversity, the ISc value based on the DCGAN model reaches 6.10, which is 1.55 higher than that based on the GANs model. The results show that the improved DCGAN model has more advantages than the GANs model, can effectively solve the problem of low-quality images being output by the GANs model, and achieves good results.



(a)



(b)

Figure 10: Loss curves of the GANs discriminator (a). Generator (b) (Experiment 2).

Table 3: Experimental results of the image quality comparison.

Experimental group	Model	IS (28×28)
Group 1	GANs	ISc: 4.55
	DCGAN	ISc: 6.10
Group 2	GANs	ISs: 4.80
	DCGAN	ISs: 6.82

5. Discussion and Conclusions

Based on the GANs model and the improved DCGAN model, this paper uses the MNIST dataset as experimental data for experiments on the algorithms and evaluates the quality and diversity of the generated images based on the ISs and ISc metrics:

(1) This paper compares and analyzes the different performances of traditional GANs and the DCGAN in two groups of experiments.

The DCGAN is a model based on a combination of GANs and a convolutional neural network. The fully connected layer is replaced by a convolution layer and deconvolution layer. The structure of the DCGAN layers is redesigned: An upsampling layer is used in the output layer of the generator to expand the data, and a dropout layer is added in each layer of the discriminator. To solve the problem that the gradient easily disappears in GANs, the generator output process uses a beneficial Tan*h* function; the ReLU function is used in the other layers of the generator, and the LeakyReLU function is used in all layers of the discriminator. In the two groups of comparative experiments, it can be concluded that the images of the GANs model are generated in a column because of the flattening layers and reshaping layers. The image generation method using the DCGAN constructed in this paper involves generating regional and characteristic images by using a conv2dspread layer (two-dimensional anti-convolution layer), which fundamentally solves the problem of low-quality images being generated by the GANs.

(2) In this paper, through the optimized DCGAN model, it is proven that the variables of a simple image distribution tend to be independent of each other, thereby overcoming the error caused by the independence of traditional indexes.

The ISs value under the simple image distribution is taken as the image quality evaluation result, and the ISc value of the complex image distribution is combined with it to comprehensively evaluate the diversity and quality of the generated image. Through two groups of experiments, it can be concluded that the image quality evaluation index ISs of the DCGAN model is 6.82, which is 2.02 higher than that of the GANs model with the same image distribution. The image diversity index ISc of the DCGAN is 6.10, while that of the GANs is only 4.55. The reason for this is that the image quality evaluation index ISs of the DCGAN is 6.82, which is 2.02 higher than that of the GANs model with the same image distribution. In addition, the DCGAN has more advantages than the GANs in terms of its model framework and model detail parameters.

Abbreviations

ISs:	Simple initial fraction
ISc:	Complex initial fraction
GANs:	Generative adversarial nets
DCGAN:	Deep convolutional generative adversarial network
CNN:	Convolutional neural network.

Data Availability

Some or all data, models, or codes that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No:4210040255), the Sichuan Science and Technology Program (2021JDRC0108), Opening Fund of Geomathematics Key Laboratory of Sichuan Province (scsxdz2020yb05) and Chengdu University of Technology Development Funding Program for Young and Middleaged Key Teachers (10912-JXGG2020-06251)".

References

[1] J. Wang, Y. Yang, T. Wang, R. Sherratt, and J. Zhang, "Big data service architecture: a survey," *Journal of Internet Technology*, vol. 21, no. 2, pp. 393–405, 2020.

[2] A. V. D. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of the ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning*, vol. 48, pp. 1747–1756, 2016.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the ICLR 2014 International Conference on Learning Representations (ICLR)*, 2014.

[4] J. Chen, K. Li, K. Bilal, Xu Zhou, K. Li, and P. S. Yu, "A Bi-layered parallel training architecture for large-scale convolutional neural networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 965–976, 2019.

[5] D. Cao, K. Zeng, J. Wang et al., "BERT-Based deep spatial-temporal network for taxi demand prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 202113 pages, 2021.

[6] N. Shobha Rani and B. J Nair, "A deep convolutional architectural framework for radiograph image processing at bit plane level for gender & age assessment," *Computers, Materials & Continua*, vol. 62, no. 2, pp. 679–694, 2020.

[7] I. J. Goodfellow, J. P. Abadie, M. Mirza, B. Xu, D. W. Farley, and S. Ozair, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.

[8] M. Mirza and S. Osindero, *Conditional Generative Adversarial Nets*, vol. 1411, 2014 https://arxiv.org/abs/1411.1784?context=cs.

[9] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proceedings of the NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2234–2242, 2016.

[10] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image translation with conditional adversarial networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, IEEE, Honolulu, HI, USA, July 2017.

[11] H. Zhang, T. Xu, and H. Li, "StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5908–5916, IEEE, Venice, Italy, October 2017.

[12] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, IEEE, Venice, Italy, October 2017.

[13] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proceedings of the ICLR 2018 International Conference on Learning Representations*, 2018, https://arxiv.org/abs/1812.04948.

[14] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1510–1519, IEEE, Venice, Italy, 22-29 October 2017.

[15] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proceedings of the ICLR 2019 7th International Conference on Learning Representations*, 2019, https://arxiv.org/abs/1809.11096.

[16] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proceedings of the ICLR 2016 International Conference on Learning Representations*, 2016.

[17] B. Pu, K. Li, S. Li, and N. Zhu, "Automatic fetal ultrasound standard plane recognition based on deep learning and IIoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7771–7780, 2021.

[18] C. Chen, K. Li, S. G. Teo, X. Zou, K. Li, and Z. Zeng, "Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 14, no. 4, pp. 42–51, 2020.

[19] S. Liu, M. Yu, M. Li, and Q. Xu, "The research of virtual face based on deep convolutional generative adversarial networks using TensorFlow," *Physica A: Statistical Mechanics and Its Applications*, vol. 521, pp. 667–680, 2019.

[20] M. A. B. Mahmoud and P. Guo, "A novel method for traffic sign recognition based on DCGAN and MLP with PILAE algorithm," *IEEE Access*, vol. 7, Article ID 74602, 2019.

[21] W. Fang, Y. Ding, F. Zhang, and J. Sheng, “Gesture recognition based on CNN and DCGAN for calculation and text output,” *IEEE Access*, vol. 7, Article ID 28230, 2019.

[22] S. Aslan, U. Gudukbay, B. U. Toreyin, and A. E. Cetin, “Early wildfire smoke detection based on motion-based geometric image transformation and deep convolutional generative adversarial networks,” in *Proceedings of the ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8315–8319, IEEE, Brighton, UK, May 2019.

[23] J. Viola, Y. Q. Chen, and J. Wang, “Faultface: deep convolutional generative adversarial network (DCGAN) based ball-bearing failure detection method,” *Information Sciences*, vol. 542, no. 4, pp. 195–211, 2021.

[24] K. Y. Cheng, R. Tahir, L. K. Eric, and M. Z. Li, “An analysis of generative adversarial networks and variants for image synthesis on MNIST dataset,” *Multimedia Tools and Applications*, vol. 79, no. 19-20, Article ID 13725, 2020.