electronics

MDPI

Article

Data Augmentation Based on Generative Adversarial Network with Mixed Attention Mechanism

Yu Yang *, Lei Sun, Xiuqing Mao and Min Zhao

School of Cryptography Engineering, Information Engineering University, Zhengzhou 450001, China; sl20210221@163.com (L.S.); mxq_zf@163.com (X.M.); zm717207@163.com (M.Z.)
* Correspondence: yuyoung0107@163.com

Abstract: Some downstream tasks often require enough data for training in deep learning, but it is formidable to acquire data in some particular fields. Generative Adversarial Network has been extensively used in data augmentation. However, it still has problems of unstable training and low quality of generated images. This paper proposed Data Augmentation Based on Generative Adversarial Network with Mixed Attention Mechanism (MA-GAN) to solve those problems. This method can generate consistent objects or scenes by correlating the remote features in the image, thus improving the ability to create details. Firstly, the channel-attention and the self-attention mechanism are added into the generator and discriminator. Then, the spectral normalization is introduced into the generator and discriminator so that the parameter matrix satisfies the Lipschitz constraint, thus improving the stability of the model training process. By qualitative and quantitative evaluations on small-scale benchmarks (CelebA, MNIST, and CIFAR-10), the experimental results show that the proposed method performs better than other methods. Compared with WGAN-GP (Improved Training of Wasserstein GANs) and SAGAN (Self-Attention Generative Adversarial Networks), the proposed method contributes to higher classification accuracy, indicating that this method can effectively augment the data of small samples.

Keywords: generative adversarial network; mixed attention mechanism; small samples; data augmentation

Citation: Yang, Y.; Sun, L.; Mao, X.; Zhao, M. Data Augmentation Based on Generative Adversarial Network with Mixed Attention Mechanism. *Electronics* **2022**, *11*, 1718. https://doi.org/10.3390/electronics11111718

Academic Editor: Manohar Das

Received: 21 February 2022
Accepted: 27 April 2022
Published: 27 May 2022

1. Introduction

The three significant elements of artificial intelligence (AI) are data, computing power, and algorithm. AI is developing more and more rapidly. The authors of [1] attempted to make these models intelligent by introducing more explicable and physically meaningful knowledge in a non-convex modeling way that meets the actual needs. The authors of [2] proposed a new small-batch GCN that allows training of large-scale GCN in a small-batch manner. More importantly, their mini-GCN can infer out of sample data and improve classification performance without retraining the network. Deep learning, a branch of AI, relies heavily on a good deal of data, and the size of the dataset has a specific impact on the model's performance. The data used for training is usually a large number of indeed collected images, text, or voice data. Meanwhile, the training data need to be artificially marked with some information, which consumes a lot of time and cost. However, in some particular fields, such as astronomy, medical treatment, security, aviation, power systems, and so on, it is burdensome to acquire enormous data. So, efficiently acquiring high-quality data has become a research hotspot, and Data Augmentation (DA) technology has emerged.

Traditional image DA mainly involves geometric transformation and color transformation. Geometric transformation performs a simple transformation on the image, such as flipping, rotating, and cutting. The most common color transformation is based on noise. The image DA technique proposed in [3] combines sharpening and noise reduction. Although these methods can realize DA, they are limited to changing the original images, which results in repetitive and single data distribution. Generative Adversarial

Network (GAN) [4] provides a feature learning framework for solving the problems in DA. The framework includes two modules: Generator (G) and Discriminator (D), and both the modules learn to produce output by adversarial pattern. GAN has been universally used in diverse image generation tasks. However, GAN still has some problems in that it convergence is difficult, and it is easy to produce mode collapse. Based on the original GAN, a series of improved models have been proposed. Wasserstein GAN (WGAN) [5] was proposed by Martin et al. It has basically solved model collapse and ensured the generated samples diversity. Improved Training of Wasserstein GANs (WGAN-GP) [6] was proposed by Ahmed et al. Gradient penalty was used to achieve an approximate 1-Lipschitz [7] constraint on the discriminator, making the GAN training more stable and converging faster. Based on this, it can generate higher-quality samples. However, WGAN-GP has no fundamental implementation restriction on 1-Lipschitz conditions. In February 2018, Spectral Normalization GAN (SNGAN) [8] was proposed, and the main idea is to restrict the modulus of a gradient to a range. The parameter matrix W of the neural network is normalized by spectral norm. Although some achievements have been made in improving the speed of convergence, further improvement is still needed. Convolutional Neural Network (CNN) [9] was used for Deep Convolution GAN (DCGAN) [10]. Based on CNN's strong fitting and expressive ability, the quality of generating images is greatly improved, but model collapse can easily occur during training. The authors of [11–13] generated high-resolution face images, but these models are all complicated and time-consuming. The authors of [14,15] proposed that a single natural image can train the GAN model. The authors of [16] proposed a method in which the low-resolution images can be transformed into the high-resolution images.

AugGAN [17] is an improved GAN model that emphasizes the cross-domain adaptation of DA. For vehicle detection, if we use the data collected in the daytime to train the model, the model will achieve a poor performance when it is used at night. Thus, people are required to spend time collecting the data at night. To solve this problem, AugGAN takes advantage of the high redundancy of daytime and night, and it transfers one domain to another. In this way, it realizes data transformation to complete DA. The work in [18] proposed DAGAN, which obtains source domain data and then learns to generate other data items by taking any data item. DAGAN could improve the performance of a classifier after DA. For medical image generation using GAN, a DA method was proposed in [19]. A training scheme based on classical DA was exploited to enlarge the training dataset, and GAN was then used to expand data diversity further. In addition, DCGAN was proposed for DA to improve the performance of the Person re-identification model [20]. The authors of [21] proposed an image-to-image translation network to generate high-quality images for data augmentation, and this method can be used to improve the performance of different computer vision tasks. The proposed network consists of a detail branch, a transfer branch, a filter module, and a reconstruction module. It shows significant effectiveness in preserving details and structural information. The authors of [22] proposed a deep translation (GAN)-based change detection network called DTCDN for optical and SAR remote sensing images. The basic idea is to first utilize image translation to reduce the difference of heterogeneous remote sensing images and then adopt an improved deep network to detect changes between two periods. The experimental change detection results of the optical and SAR remote sensing images in four different datasets demonstrate the validity of the proposed method. However, the detection capability of the method for complex multi-source remote sensing images need to be further improved. The authors of [23] introduced basic data augmentation operations to address the fundamental data limitation problem in applying deep learning for remote sensing image processing. It uses data augmentation to improve the remote sensing scene classification performance of CNNs.

Attention mechanism was first developed as part of the encoder–decoder framework in RNN (Recurrent Neural Network) to encode lengthy input statements, and it has been widely used in RNN subsequently [24]. However, no matter how long the antecedent is

and how much information it includes, it will finally be compressed into a vector of several hundred dimensions. This means that the larger the context, the more information will be lost from the final state vector. In CNN, the feature map was acted on the attention mechanism, trying to acquire available information in the feature map and achieve a better task effect [25]. "Attention is all you need" introduced the transformer model with self-attention as the basic unit [26], which makes the attention mechanism genuinely successful. The key of Squeeze-and-Excitation Networks (SENet) [27] is SE block, in which the interdependence between feature maps was explicitly modeled by an attention mechanism. The importance of each feature map is acquired adaptively through learning, and then, the original data are updated according to the matter. Han Zhang et al. proposed Self-Attention GAN (SAGAN) [28]. It adopted a non-local so that the network can effectively construct relationships between different regions. SAGAN considers the global information in each layer to solve unclear local details in tiny. In addition, it finds a good balance between increasing the receptive field and reducing the number of parameters. However, it ignores the relationship between the channels of the feature map, and many abnormal structures are presented in the generated images.

It is different from the present work [29]: in this research, not only CelebA but also MNIST and CIFAR-10 are used; the method of regular training was introduced; the classification method was used to indicate the validity of DA further.

A summary of the prime contributions in this paper follows:

(1) Through research and experiments, a mixed attention mechanism was introduced into the generator and discriminator so that the GAN can capture structure, texture, and other details of images. Meanwhile, the geometry and distribution of the image can be captured effectively, and more detailed and realistic images can be depicted.
(2) Spectral Normalization (SN) was applied to the generator and discriminator, respectively, to make the training process more stable.
(3) Compared with the baseline method, data generated by the proposed method can efficaciously improve classification accuracy and convergence, indicating that the DA of small sample data is reliable and effective.

This paper is organized as follows. Section 2 introduces the related work, including GAN and DCGAN. Section 3 introduces the proposed method in detail, including the overall framework, the theory of self-attention and channel-attention, and spectral normalization. Section 4 presents experimental process and analysis, the datasets and parameter settings of the experiment, and the quantitative evaluation method and experimental results. Section 5 analyzes time efficiency and complexity. The final section concludes this paper.

2. Related Work

2.1. Generative Adversarial Network

GAN consists of the generator and the discriminator. The generator generates images from a given noise z; at the same time, the discriminator determines whether the sample is real or fake by measuring the divergence between the distribution of generated samples and real samples.

For real data x, random noise z, generator G, and discriminator D, the objective function of an original GAN is:

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

$p_{data}(x)$ represents the distribution of real samples, and $p_z(z)$ represents the distribution of generated samples. $D(x)$ is a probability, and it indicates the possibility that the input of D are real samples. The training of GAN is a process of the max–min game. In the training process, D is optimized after G is fixed, and G is optimized after D is fixed.

2.2. Deep Convolution Generative Adversarial Network

DCGAN is mainly composed of a generator (G) and discriminator (D). The generator and discriminator compete with each other to reach the equilibrium.

Figure 1 shows the structure of DCGAN. The hidden variable z is generally a random noise subject to Gaussian distribution, putting z into the generator and obtaining $G(z)$. After the discriminator obtains $G(z)$, it compares it with the real data, making a true or false judgment and feeding it back to the generator. The discriminator optimized process is similar to the two-classification problem.

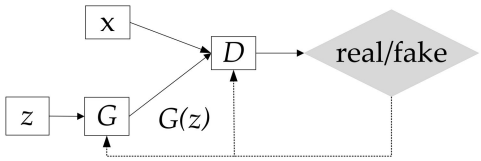


Figure 1. The structure of DCGAN.

The first step is to fix the generator and train the discriminator, sampling x from the real data distribution and sampling z from the prior distribution, and then generating data $G(z)$ through the generator. x and $G(z)$ input a discriminator for training to maximize the objective function.

3. Proposed Method

This paper proposed a GAN model based on the Mixed Attention Mechanism (MAM) called MA-GAN to solve the problem that the original GAN cannot effectively capture the geometry and shape of images. By capturing the importance of features at different positions, this model can obtain the weight parameters of corresponding areas, accelerating the network's convergence. Meanwhile, our method can acquire the utmost out of the features of each layer for image generation. In this way, the model can flexibly capture the local and global connections, achieving a high expressive ability and a low model complexity. Figure 2 displays the network structure of the MA-GAN.

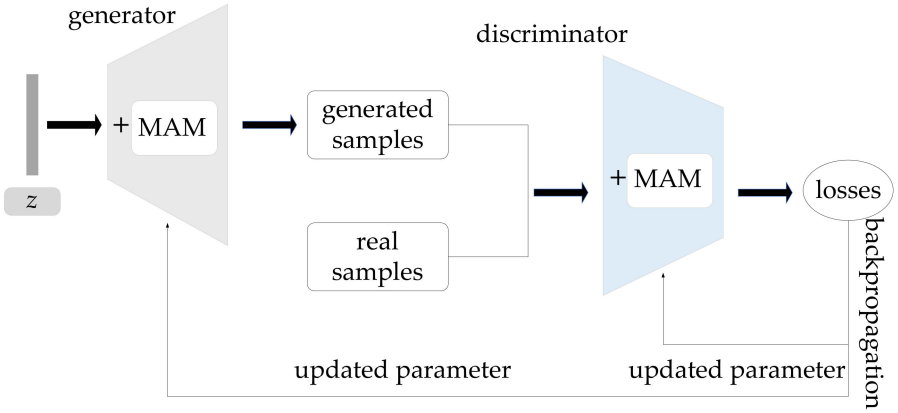


Figure 2. Network structure of the proposed MA-GAN.

3.1. Self-Attention Mechanism

Traditional convolutional GAN can easily generate images from simple geometric figures such as ocean and sky, but it fails on images with specific geometric shapes (such as dogs, horses, etc.). For example, it can produce the skin textures of a cat but cannot generate the legs. In addition, it pays great attention to details when generating face images. Taking a left and right eye as an example, as long as there is a slight asymmetry, the generated faces will be unreal. It is difficult for convolutional kernels to cover a large area. When a convolutional operation is performed on the left eye, it does not know the impact of the right eye on the left eye. In this case, the generated images will lack the integrity of

facial structure features. Since the convolutional operation has a receptive field, resulting in low model learning efficiency and loss of details. Some traditional methods use a deeper CNN or directly use a fully connected layer to obtain global information, generating many parameters and computations. Therefore, in order to establish a balance between the efficiency and an enormous receptive field and increase the size of the receptive field, the self-attention mechanism is added.

The self-attention provides self-attention-based remote dependency. Figure 3 displays the structure of the self-attention mechanism.

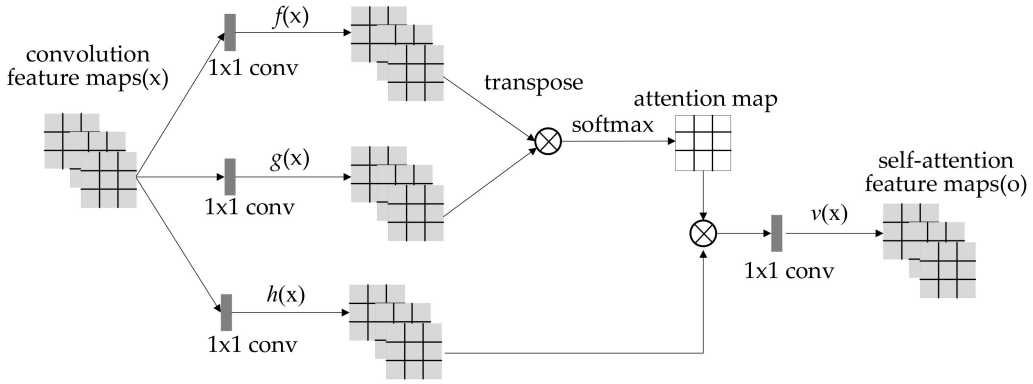


Figure 3. Schematic diagram of the self-attention structure.

In the self-attention module, the features of the previously hidden layer $x \in \mathrm{R}^{C \times N}$ are converted into two feature space functions f, g to calculate the attention, where C is the number of channels and N is the product of width times height. $f(x)$, $g(x)$ and $h(x)$ are all 1×1 convolutions, and they are respectively called the query, key, and value. The output has a different channel. The role of 1×1 convolution is to reduce the number of channels, and the size of the generated images is affected by the size and stride of the convolutional kernel. For a 1×1 convolutional kernel with a stride of 1, then the size of the generated image will not be changed. Since the convolutional process usually involves an activation function, much non-linearity is introduced without changing the size of the input, which enhances the expressive ability of the neural network. $f(x) = \mathrm{W}_f x$, $g(x) = \mathrm{W}_g x$. $\mathrm{W}_g \in \mathrm{R}^{\overline{C} \times C}$ and $\mathrm{W}_h \in \mathrm{R}^{\overline{C} \times C}$ are the learning weight matrices.

$$\overline{C} = C/8 \tag{2}$$

As shown in Formula (3), the output of $f(x_i)$ is transposed and multiplied with $g(x_j)$. Then, the result is normalized by softmax to obtain an attention map.

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})} \tag{3}$$

where $\beta_{j,i}$ represents the influence degree of the model at the i-th position when the j-th area is synthesized. These learning weight matrices are called "attention maps" that essentially quantify the importance of pixel j relative to image i.

The output of the attention layer is:

$$o = (o_{1,} o_{2,} \ldots, o_{j,} \ldots, o_N) \in \mathrm{R}^{C \times N} \tag{4}$$

$h(x_i)$ is multiplied with the obtained attention map pixel-by-pixel to obtain the feature map of the adaptive attention, where x_i is the i-th extracted feature map.

$$h(x_i) = \mathrm{W}_h x_i \tag{5}$$

$$v(x_i) = W_v x_i \tag{6}$$

$$o_j = v\left(\sum_{i=1}^{N} \beta_{j,i} h(x_i)\right) \tag{7}$$

In addition, the output of o is further multiplied by the scale parameter γ. The result is added to the input feature map x to obtain the final output. The formula is as follows:

$$y_i = \gamma o_i + x_i \tag{8}$$

To take into account the relevance of domain information and long-distance features, a transition parameter γ with an initial value of 0 is introduced. It makes the model gradually assign weights to other long-distance feature details starting from the domain information.

3.2. Channel-Attention Mechanism

As shown in Figure 4, the channel-attention module automatically obtains the importance of each feature channel by learning. Then, according to this importance, it enhances useful features and suppresses less valuable features for the current task. Assuming that the neural network uses global information to enhance the helpful information while suppressing the useless information, for channel-dimensional characteristic fusion, the convolutional operation fuses all the channels of the input feature map by default. Figure 4 displays the structure of the channel-attention mechanism.

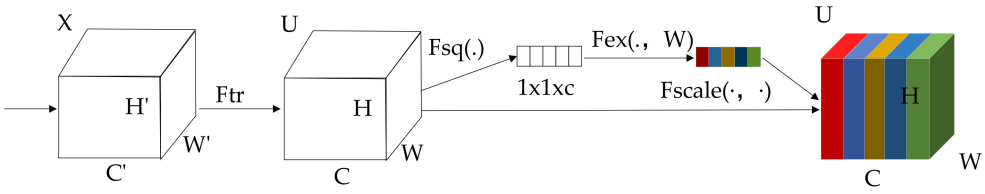


Figure 4. Structure of the channel-attention mechanism.

$X \in R^{H' \times W' \times C'}$, $U \in R^{H \times W \times C}$, $V = (v_1, v_2, \ldots v_c)$ represents the learning set of convolutional kernels, and v_c represents the parameters of the c-th convolutional kernel. The output is $U = (u_1, u_2, \ldots u_c)$.

$$u_c = v_c * X = \sum_{i=1}^{C'} v_c^s * x^s \tag{9}$$

$*$ defines a convolution operation. $v_c = (v_c^1, v_c^2, \ldots, v_c^{c'})$, $X = (x^1, x^2, \ldots, x^{c'})$, and $u_c \in R^{H \times W}$. v_c^s is a two-dimensional convolutional kernel with s channels, and x^s represents the s-th input. The spatial feature on a channel is input to learn the spatial feature relationship. However, after the convolutional results of each channel are added, the channel feature relationship is mixed with the spatial relationship learned by the convolutional kernel. To this end, the channel attention module is designed to abstract this mix so that the model directly learns the channel feature relationship. After F_{tr}, the extracted features are obtained, and the size of convolution at this time is $H \times W \times C$.

Since convolution only operates in a local space, it is difficult for U to obtain enough information to extract the relationship between channels. Therefore, the squeeze is proposed. It first encodes the entire spatial feature on a channel into a global feature. Then, by using Global Average Pooling (GAP), it compresses the spatial information into a channel descriptor.

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j) \tag{10}$$

Feature compression is performed in the spatial dimension, and each two-dimensional feature channel becomes a real number that has a global receptive field to a certain extent.

The output dimension matches the number of input feature channels, representing the feature channel that is the global distribution of responding. The dimension of the original feature map is $H \times W \times C$, where H, W, and C are respectively the height, width, and the number of channels. After the feature map is compressed from $H \times W \times C$ to $1 \times 1 \times C$, it has only one dimension. In this way, the parameter in the dimension can be used to obtain the previous $H \times W$ global field of vision, and the perception area is more comprehensive. Then, the excitation operation is performed to use the information gathered in the squeezed operation and completely capture the dependence of the channel. After the $1 \times 1 \times C$ representation of the squeeze is obtained, a fully connected layer is added to predict the importance of each channel. Finally, the importance of different channels is obtained and applied to the corresponding channel of the previous feature map.

$$s = F_{ex}(z, \mathrm{W}) = \sigma(g(z, \mathrm{W})) = \sigma(\mathrm{W}_2 \mathrm{ReLU}(\mathrm{W}_1, z)) \tag{11}$$

σ represents a sigmoid activation function; $\mathrm{W}_1 \in \mathrm{R}^{\frac{c}{r} \times c}$, $\mathrm{W}_2 \in \mathrm{R}^{c \times \frac{c}{r}}$, W_1 and W_2 are weight matrices of the two fully connected layers; r equals 16. The use of this parameter is to reduce the number of channels and the amount of computation. The output weight of the excitation is regarded as the importance of each channel of the feature.

In Formula (11), the result of the squeeze is z, and $\mathrm{W}_1 \times Z$ is a fully connected operation. The dimension of W_1 is $\frac{c}{r} \times c$, and the dimension of z is $1 \times 1 \times c$, so the dimension of $\mathrm{W}_1 \times Z$ is $1 \times 1 \times \frac{c}{r}$. After the multiplication by W_2, the dimension of the output is $1 \times 1 \times c$. Finally, s is obtained through sigmoid function. c represents the number of channels that are used to describe the weight of each feature map, and the weight is learned through previous fully connected layers and nonlinear layers. The function of the two fully connected layers is to fuse the feature map information of each channel. After s is obtained, U is processed, where u_c is a matrix. s_c is the activation value of each channel learned, and it is multiplied with u_c to complete the F_{scale} operation. After this operation, the useless information tends to 0 while the useful information is still useful. Corresponding to:

$$X_c' = F_{scale}(u_c, s_c) = s_c u_c \tag{12}$$

To reduce the complexity of the model and improve its generalization ability, two fully connected (FC) layers are involved in the model. The first FC layer performs dimensionality reduction before the ReLU activation function is used, while the second FC layer restores the original dimension. Figure 5 displays the network of the channel-attention mechanism.

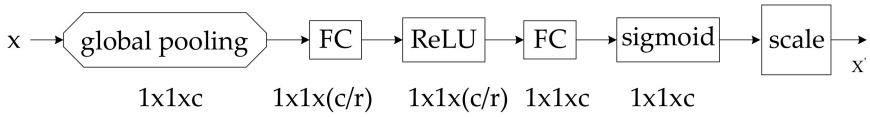


Figure 5. Structure of channel-attention network.

3.3. Weight Standard Technique of Spectral Normalization

To handle the problem that the GAN model is easy to collapse and difficult to converge, the spectral norm normalized parameter matrix is introduced to limit the gradient to a fixed range and slow down the convergence of the discriminator, thereby improving the training stability of GAN.

A feedforward neural network can be expressed as a cascading operation:

$$x^l = D^l_{\theta,x} \mathrm{W}^l x^{l-1} \tag{13}$$

where l represents the number of layers, $l \in \{1, \ldots, L\}$; x^{l-1} is the input of the l-th layer, and x^l is the output of the l-th layer; W^l, respectively, represents the weight matrix and bias vector of a layer. To facilitate derivation, bias is omitted in this paper.

$D^l_{\theta,x}$ represents the diagonal matrix that is used in the activation function. The diagonal value of the matrix is 0 when the input is negative; otherwise, it is 1. Let θ denote the set of all parameters that need to be learned, $\theta = \left\{ \mathrm{W}^l, b^l \right\}_{l=1}^{L}$. The function formed by the entire layer is denoted as f_θ. Given K sets of training data, i.e., $(x_i, y_i)_{i=1}^{K}$, the loss function is defined as $\frac{1}{K}\sum_{i=1}^{K} L(f_\theta(x_i), y_i)$. As for the parameters in θ, $\sigma(W_{\theta,x})$ denotes the calculational of the spectral norm. It is mathematically equivalent to the calculation of the maximum singular value of matrix $W_{\theta,x}$, which is shown as follows:

$$\sigma(A) := \max_{h:h\neq 0} \frac{||Ah||_2}{||h||_2} = \max_{||h||_2 \leq 1} ||Ah||_2 \tag{14}$$

For Lipschitz continuity, an additional restriction is imposed on a consecutive function f, which requires a constant $0 \leq K$ to make any two elements in the domain to meet the following:

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2| \tag{15}$$

The constant K is called the Lipschitz constant of function f.

To satisfy the Lipschitz limit, the value of θ can be obtained by minimizing $f(x+\xi) - f(x)$, where ξ is a perturbed vector with a small modulus. Considering the small area of x, f_θ can be regarded as a linear function with affine mapping $x \to \mathrm{W}_{\theta,x}x + b_{\theta,x}$, where $\mathrm{W}_{\theta,x}$ is a matrix and $b_{\theta,x}$ is a vector. Both $\mathrm{W}_{\theta,x}$ and $b_{\theta,x}$ depend on the values of θ and x. Then, for a minor disturbance ξ, thus,

$$\frac{||f_\theta(x+\xi) - f(x)||_2}{||\xi||_2} = \frac{||\mathrm{W}_{\theta,x}(x+\xi) + b_{\theta,x} - (\mathrm{W}_{\theta,x}x + b_{\theta,x})||_2}{||\xi||_2} = \frac{||\mathrm{W}_{\theta,x}\xi||_2}{||\xi||_2} \leq \sigma(\mathrm{W}_{\theta,x}) \tag{16}$$

$\mathrm{W}_{\theta,x}$ can be rewritten as:

$$\mathrm{W}_{\theta,x} = D^L_{\theta,x}\mathrm{W}^L D^{L-1}_{\theta,x}\mathrm{W}^{L-1} \ldots D^1_{\theta,x}\mathrm{W}^1 \tag{17}$$

Since LeakyReLU and ReLU satisfy the 1-Lipschitz constraint, then for each $l \in \{1, \ldots L\}$, there is $\sigma(D^l_{\theta,x}) \leq 1$ so that:

$$\sigma(\mathrm{W}_{\theta,x}) \leq \sigma(D^L_{\theta,x})\sigma(\mathrm{W}^L)\sigma(D^{L-1}_{\theta,x})\sigma(\mathrm{W}^{L-1}) \ldots \sigma(D^1_{\theta,x})\sigma(\mathrm{W}^1) \leq \sigma(\mathrm{W}^L)\sigma(\mathrm{W}^{L-1}) \ldots \sigma(\mathrm{W}^1) \leq \prod_{l=1}^{L}\sigma(\mathrm{W}^l) \tag{18}$$

To limit the spectral norm of $\mathrm{W}_{\theta,x}$, only the spectral norm of each $l \in \{1, \ldots L\}$ limiting W^l is sufficient. For a linear layer function $g(h) = \mathrm{W}h$, the Lipschitz paradigm can be calculated as follows:

$$\left|\left|g\right|\right|_{Lip} = \sup_h \sigma(\nabla g(h)) = \sup_h \sigma(\mathrm{W}) = \sigma(\mathrm{W}) \tag{19}$$

If the Lipschitz paradigm of the activation function $\left|\left|a_l\right|\right|_{Lip} = 1$, the following inequality constraint holds:

$$\left|\left|g_1 \circ g_2\right|\right|_{Lip} \leq \left|\left|g_1\right|\right|_{Lip} \cdot \left|\left|g_2\right|\right|_{Lip} \tag{20}$$

where $\circ$ represents a composite function. Based on the inequality constraint, the restriction of f_θ Lipschitz paradigm can be expressed as:

$$||\mathrm{f}||_{Lip} \leq ||\mathrm{h}_L \to \mathrm{W}^{L+1}h_L||_{Lip} \cdot ||a_L||_{Lip} \cdot ||h_{L-1} \to \mathrm{W}^L h_{L-1}||_{Lip} \ldots ||a_1||_{Lip} \cdot ||\mathrm{h}_0 \to \mathrm{W}^1 h_0||_{Lip} = \prod_{l=1}^{L+1}\left|\left|(h_{l-1} \to \mathrm{W}^l h_{l-1})\right|\right|_{Lip} = \prod_{l=1}^{L+1}\sigma(\mathrm{W}^l) \tag{21}$$

Therefore, by ensuring that $\sigma(W^l)$ is always equal to 1, this function can meet the 1-Lipschitz limit. Let the parameters of each layer of the network be divided by the spectral norm of the layer number matrix. The division is represented as follows:

$$\overline{W}(W) := \frac{W}{\sigma(W)} \tag{22}$$

In the training process, different learning rates are given to balance speed of the generator and discriminator. To investigate the effects of spectrum normalization on the model performance, ablation experiments were performed.

Figure 6a illustrates the result of the generator without spectral normalization after 600 iterations. It can be seen that generated images contain much noise, and the training is very unstable. Figure 6b illustrates the result of the generator with spectral normalization after 600 iterations. We can see that the generated images have better quality than those shown in Figure 6a, and the training is more stable.

(a) (b)

Figure 6. The samples generated by different models. (**a**) 600 iterations without SN, (**b**) 600 iterations with SN.

As for CelebA, the generator and discriminator network structures each have five layers. As for CIFAR-10 and MNIST, the network structure of the generator and discriminator each have four layers. The experimental result shows that after the channel-attention mechanism is added to the first two layers of the generator and discriminator, the quality of the generated images is improved. However, adding this mechanism does not result in excessive parameters and time overhead, because the low-level layers extract detailed information. Meanwhile, the self-attention mechanism is added to the last two layers of the generator and discriminator. The reason is that for larger feature maps, the self-attention mechanism is a supplement to convolution but works similar to local convolution for modeling the dependencies on smaller feature maps. The network structures of the generator and discriminator in CelebA, CIFAR-10, and MNIST are shown in the following tables.

The network structure of the generator in CelebA is shown in Table 1. The generator receives a 100-dimensional noise vector, and then, the vector goes through the first layer of transposed convolution. The size of the input channel is 100, while that of the output channel is 512. A 4×4 kernel with a stride of 1 is used here. Next, the data are processed by SN, BN, and channel-attention, respectively. Finally, the ReLU activation function is used, and the size of the output feature map is $4 \times 4 \times 512$. After the second layer of transposed convolution, the size of the input channel is 512, and that of the output channel is 256. A 4×4 kernel with a stride of 2 is used here. Then, the data pass through SN, BN, and channel-attention, respectively. Finally, the ReLU activation function is used, and the output feature map is $8 \times 8 \times 256$. After the third layer of transposed convolution, the size of the input channel is 256, while that of the output channel is 128. Then, the data pass through SN, BN, and self-attention, respectively. Finally, the ReLU activation function is used, and the size of the output feature map is $16 \times 16 \times 128$. After the fourth layer of

the transposed convolution, the size of the input channel is 128, while that of the output channel is 64. Then, the data are processed by going through SN, BN, and self-attention, respectively. Finally, the ReLU activation function is used, and the size of the output feature map is $32 \times 32 \times 64$. After the fifth layer of transposed convolution, the size of the input channel is 64, while that of the output channel is 3. Finally, the Tanh activation function is used, and the size of the output is $64 \times 64 \times 3$.

Table 1. The network structure of the generator in CelebA.

Input	Operation	Output
—	ConvTranspose2d (100, 512, k = 4 × 4, s = 1) SN + BN + channel-attention + ReLU	4 × 4 × 512
4 × 4 × 512	ConvTranspose2d (512, 256, k = 4 × 4, s = 2) SN + BN + channel-attention + ReLU	8 × 8 × 256
8 × 8 × 256	ConvTranspose2d (256, 128, k = 4 × 4, s = 2) SN + BN + ReLU	16 × 16 × 128
16 × 16 × 128	+self-attention	16 × 16 × 128
16 × 16 × 128	ConvTranspose2d (128, 64, k = 4 × 4, s = 2) SN + BN + ReLU	32 × 32 × 64
32 × 32 × 64	+self-attention	32 × 32 × 64
32 × 32 × 64	ConvTranspose2d (64, 3, k = 4 × 4, s = 2) Tanh	64 × 64 × 3

The network structure of the discriminator in CelebA is shown in Table 2. The size of the input image to the discriminator is $64 \times 64 \times 3$. After the first layer of convolution, the size of the input channel is 3, while that of the output channel is 64. A 4×4 kernel with a stride of 2 is used here. Then, the data are processed by SN and channel-attention, respectively. Finally, the LeakyReLU activation function is used, and the size of the output feature map is $32 \times 32 \times 64$. After the second layer of convolution, the size of the input channel is 64, while that of the output channel is 128. Finally, the LeakyReLU activation function is used, and the size of the output feature map is $32 \times 32 \times 64$. After the fifth layer of convolution, the size of the input channel is 512, while that of the output channel is 1. A 4×4 kernel with a stride of 1 is used.

Table 2. The network structure of the discriminator in CelebA.

Input	Operation	Output
64 × 64 × 3	Conv2d (3, 64, k = 4 × 4, s = 2) SN + channel-attention + LeakyReLU	32 × 32 × 64
32 × 32 × 64	Conv2d (64, 128, k = 4 × 4, s = 2) SN + channel-attention + LeakyReLU	16 × 16 × 128
16 × 16 × 128	Conv2d (128, 256, k = 4 × 4, s = 2) SN + LeakyReLU	8 × 8 × 256
8 × 8 × 256	+self-attention	8 × 8 × 256
8 × 8 × 256	Conv2d (256, 512, k = 4 × 4, s = 2) SN + LeakyReLU	4 × 4 × 512
4 × 4 × 512	+self-attention	4 × 4 × 512
4 × 4 × 512	Conv2d (512, 1, k = 4 × 4, s = 1)	—

The network structure of the generator in CIFAR-10 is shown in Table 3. The generator receives a 100-dimensional noise vector, and then, the vector goes through the first layer of transposed convolution. The size of the input channel is 100, while that of the output channel is 256. A 4×4 kernel with a stride of 1 is used. Then, the data are processed by SN, BN, and channel-attention, respectively. Finally, the ReLU activation function is used, and the size of the output feature map is $4 \times 4 \times 256$. After the second layer of transposed convolution, the size of the input channel is 256, while that of the output channel is 128. The size of the output feature map is $8 \times 8 \times 128$. After the third layer of transposed

convolution, the size of the input channel is 128, while that of the output channel is 64. The size of the output feature map is 16 × 16 × 64. After the fourth layer of transposed convolution, the size of the input channel is 64, while that of the output channel is 3. The size of the output is 32 × 32 × 3.

Table 3. The network structure of the generator in CIFAR-10.

Input	Operation	Output
—	ConvTranspose2d (100, 256, k = 4 × 4, s = 1) SN + BN + channel-attention + ReLU	4 × 4 × 256
4 × 4 × 256	ConvTranspose2d (256, 128, k = 4 × 4, s = 2, p = 1) SN + BN + channel-attention + ReLU	8 × 8 × 128
8 × 8 × 128	+self-attention	8 × 8 × 128
8 × 8 × 128	ConvTranspose2d (128, 64, k = 4 × 4, s = 2, p = 1) SN + BN + ReLU	16 × 16 × 64
16 × 16 × 64	+self-attention	16 × 16 × 64
16 × 16 × 64	ConvTranspose2d (64, 3, k = 4 × 4, s = 2, p = 1) Tanh	32 × 32 × 3

The network structure of the discriminator in CIFAR-10 is shown in Table 4. The size of the input image to the discriminator is 32 × 32 × 3. After the first layer of convolution, the size of the input channel is 3, while that of the output channel is 64. A 4×4 kernel with a stride of 2 and a padding of 1 is used. Then, the data pass through SN and channel-attention, respectively. Finally, the LeakyReLU activation function is used, and the size of the output feature map is 16 × 16 × 64. After the second layer of convolution, the size of the input channel is 64, while that of the output channel is 128. Finally, the LeakyReLU activation function is used, and the size of the output feature map is 8 × 8 × 128. After the fourth layer of convolution, the size of the input channel is 256, while that of the output channel is 1. A 4 × 4 kernel is used.

Table 4. The network structure of the discriminator in CIFAR-10.

Input	Operation	Output
32 × 32 × 3	Conv2d (3, 64, k = 4 × 4, s = 2, p = 1) SN + channel-attention + LeakyReLU	16 × 16 × 64
16 × 16 × 64	Conv2d (64, 128, k = 4 × 4, s = 2, p = 1) SN + channel-attention + LeakyReLU	8 × 8 × 128
8 × 8 × 128	+Self-Attention	8 × 8 × 128
8 × 8 × 128	Conv2d (128, 256, k = 4 × 4, s = 2, p = 1) SN + LeakyReLU	4 × 4 × 256
4 × 4 × 256	+self-attention	4 × 4 × 256
4 × 4 × 256	Conv2d (256, 1, k = 4 × 4)	—

The network structure of the generator in MNIST is shown in Table 5. The generator receives a 100-dimensional noise vector, and then, the vector goes through the first layer of transposed convolution. The size of the input channel is 100, while that of the output channel is 512. A 3 × 3 kernel with a stride of 1 is used. Then, the data go through SN, BN, and channel-attention, respectively. Finally, the ReLU activation function is used, and the size of the output feature map is 3 × 3 × 512. After the second layer of transposed convolution, the size of the input channel is 512, while that of the output channel is 256. The size of the output feature map is 7 × 7 × 256. After the third layer of transposed convolution, the size of the input channel is 256, while that of the output channel is 128. The size of the output feature map is 14 × 14 × 128. After the fourth layer of transposed convolution, the size of the input channel is 128, while that of the output channel is 1. The size of the output is 28 × 28 × 1.

Table 5. The network structure of the generator in MNIST.

Input	Operation	Output
—	ConvTranspose2d (100, 512, k = 3 × 3, s = 1, p = 0) SN + BN + Channel-Attention + ReLU	3 × 3 × 512
3 × 3 × 512	ConvTranspose2d (512, 256, k = 5 × 5, s = 2, p = 1) SN + BN + Channel-Attention + ReLU	7 × 7 × 256
7 × 7 × 256	+Self-Attention	7 × 7 × 256
7 × 7 × 256	ConvTranspose2d (256, 128, k = 5 × 5, s = 2, outpadding = 1) SN + BN + ReLU	14 × 14 × 128
14 × 14 × 128	+Self-Attention	14 × 14 × 128
14 × 14 × 128	ConvTranspose2d (128, 1, k = 5 × 5, s = 2, p = 2, outpadding = 1) Tanh	28 × 28 × 1

The network structure of the discriminator in MNIST is shown in Table 6. The size of the input image to the discriminator is 28 × 28 × 1. After the first layer of convolution, the size of the input channel is 1, while that of the output channel is 128. A 5 × 5 kernel with a stride of 2 and a padding of 2 is used. Then, the data pass through SN and channel-attention, respectively. Finally, the LeakyReLU activation function is used, and the size of the output feature map is 14 × 14 × 128. After the second layer of convolution, the size of the input channel is 128, while that of the output channel is 56. Finally, the LeakyReLU activation function is used, and the size of the output feature map is 7 × 7 × 256. After the fourth layer of convolution, the size of the input channel is 512, while that of the output channel is 1, using a 3 × 3 kernel.

Table 6. The network structure of the discriminator in MNIST.

Input	Operation	Output
28 × 28 × 1	Conv2d (1, 128, k = 5 × 5, s = 2, p = 2) SN + channel-attention + LeakyReLU	14 × 14 × 128
14 × 14 × 128	Conv2d (128, 256, k = 5 × 5, s = 2, p = 2) SN + channel-attention + LeakyReLU	7 × 7 × 256
7 × 7 × 256	+self-attention	7 × 7 × 256
7 × 7 × 256	Conv2d (256, 512, k = 5 × 5, s = 2, p = 1) SN + LeakyReLU	3 × 3 × 512
3 × 3 × 512	+self-attention	3 × 3 × 512
3 × 3 × 512	Conv2d (512, 1, k = 3 × 3, s = 1, p = 0)	—

4. Experimental Process and Analysis

The experimental environment is shown in Table 7.

Table 7. The experimental environment.

Item	Configuration
CPU	Intel (R) Core (TM) i7-10700K CPU@3.80 GHz
GPU	NVIDIA GeForce RTX 2080 SUPER
GPU memory	8 G
OS	Windows 10 64 bits
Language	Python 3.6
Framework	PyTorch
IDE	PyCharm

4.1. Experimental Datasets

The datasets used in the experiment include CelebFaces Attributes (CelebA) [30], CIFAR-10 [31], and MNIST.

CelebA: CelebA includes 202,599 face images. Each image has a size of 178 × 218 and contains multiple perspectives and backgrounds. It is generally used for face-related learning.

CIFAR-10: CIFAR-10 was collected and organized by Hinton et al. It contains chromatic images with a length of 32 × 32 of 10 categories such as cars, frogs, horses, and boats. There are 1000 images in each class. In the training dataset, each category consists of 5000 images, a total of 50,000 images.

MNIST: MNIST contains handwritten digits of 0–9 from 250 different people. The training and testing datasets respectively contain 60,000 and 10,000 grayscale images with a size of 28 × 28.

4.2. Experimental Design

For CelebA, the size of the generated images is 64 × 64, and the total steps of the generators of MA-GAN, SAGAN, and WGAN-GP are set to 200,000. The generated images and the pre-trained weights during the training process are saved every 100 iterations.

For MNIST, the size of the generated is 28 × 28, and the total steps of the generators of MA-GAN, SAGAN, and WGAN-GP are set to 2000. The generated images and the pre-trained weights during the training process are saved every 100 iterations. A total of 500 images are randomly extracted from each category of the original dataset, and a total of 5000 images are put into the three GAN models for image generation.

For CIFAR-10, the size of the generated images is 32 × 32, and the total steps of the generators of MA-GAN, SAGAN, and WGAN-GP are set to 200,000. The generated images and the pre-trained weights during the training process are saved every 100 iterations. A total of 500 images are randomly extracted from each category of the original dataset, and a total of 5000 images are put into the three GAN models for image generation.

Gradient penalty is used as the loss function to strengthen the Lipschitz constraint on the training target: if and only if the regular term of a differentiable function gradient is less than or equal to 1, it satisfies the 1-Lipschitz constraint. Gradient penalty can observably improve speed and astringency during training. Its selection is not performed in the whole network but sampled between the true and false distributions. The model applies gradient penalties to each sample independently. The gradient penalty coefficient λ is set to 10, and the batch size is set to 64. Using the Adam optimizer [32] with $\beta_1 = 0.0$ and $\beta_2 = 0.9$, the learning rate attenuation is set to 0.95, and the learning rates of the generator and discriminator are set to 0.0001 and 0.0004, respectively.

On the classification experiment on a small-scale dataset, to focus on the impact of the DA method rather than the classifier performance, a classification network is designed based on LeNet5. For MNIST, 500 images are randomly selected from each category, creating a total of 5000 images. Before DA, the training dataset contains 3500 images while the test dataset contains 1500 images. Through DA, the training dataset is enhanced by five times to contain 17,500 images in total. CIFAR-10 contains colorful images, and the structure of these images is more complex than that of MNIST. A total of 500 images are randomly selected from each category, and a total of 5000 images are chosen from Cifar10. Before DA, the training dataset includes 3500 images while the test dataset contains 1500 images. Through DA, the training dataset is enhanced by 10 times to contain 35,000 images in total. To verify the effectiveness of DA, the classification accuracy on the test set before and after using real images and different GAN methods is compared. The higher the accuracy, the better the quality of the generated images.

4.3. Sample Quantitative Indicators

Inception score (IS) [33] is an initial scoring algorithm proposed by Salimans et al. to evaluate the semantics of generated images. Originated from Google's Inception Nets, this image evaluation index can measure the clarity of a single generated image and a variety of generated images. For each generated image, its category can be expressed by conditional probability $p(y|x)$. The greater the probability, the better. Meanwhile, the

entropy of $p(y|x)$ should be as small as possible, where x represents the given images and y represents the main objects contained in the image. For a diversity of images, it is hoped that the label distribution is uniform. Since the model is not expected to generate a specific type of image, the edge probabilities $p(y)$ need to be considered. It contains $p(y_1)$, $p(y_2)$, ..., and $p(y_n)$. As shown in the following formula:

$$p(y_1) = p(y_2) = \ldots = p(y_n) = \frac{1}{n} \tag{23}$$

The larger the entropy, the better. Denoting the entropy as E, as shown in the following formula:

$$E(p(y|x)) = -\sum_{1}^{m} p(y_i|x_i) * \log(p(y_i|x_i)) \tag{24}$$

m is the number of generated images.

KL-divergence is also called relative entropy, which measures the degree of difference between two probabilistic distributions and integrates the quality and diversity of images. Based on KL-divergence, the value of IS can be calculated as follows.

$$\text{IS(G)} = \exp\left[\frac{1}{N}\sum_{i=1}^{N} D_{\text{KL}}(p(y\left|x^{(i)}\right)\left|\right|p(y))\right] \tag{25}$$

The calculation of KL-divergence D_{KL} is shown as follows.

$$D_{\text{KL}}(Q||P) = \sum_{i=1} Q(i) Ln(\frac{Q(i)}{P(i)}) \tag{26}$$

Next, the relationship between the above two entropies and KL divergence is found:

$$\text{KL}(p(y|x) * p(y)) = \sum_{i=1}^{m} p(y_i|x_i) * \log(y_i|x_i) - \sum_{i=1}^{m} p(y_i|x_i) * \log(p(y_i)) \tag{27}$$

According to the conditional probability and the joint probability of mutually independent variables, then,

$$\text{KL}(p(y|x) * p(y)) = -E(p(y|x)) + E(p(y)) \tag{28}$$

Based on this,

$$\text{IS(G)} = \exp\left[\frac{1}{N}\sum_{i=1}^{N} (-E(p(y|x)) + E(p(y)))\right] \tag{29}$$

The smaller the conditional entropy, the larger the entropy.

The training set with different DA methods is input into the classification network for training, and then, the average accuracy of classification is evaluated on the test set. The higher the accuracy, the closer the generated images are to real images, and the better the DA effect.

4.4. Experimental Results

The baseline model in this paper is DCGAN. To verify the effectiveness of MA-GAN, the results are qualitatively and quantitatively compared with those of WGAN-GP and SAGAN that also use attention mechanisms. The common feature of these models is that DCGAN is their baseline method. These models have a simple structure and few parameters and only occupy a small GPU memory, so they can better verify the availability of the MA-GAN. For CelebA, it can be seen from Figure 7a, b that when the generator executes 100 times iteratively, the images generated by WGAN-GP and SAGAN still contain much noise. From Figure 7c, it can be seen that when the generator of MA-GAN executes

100 times iteratively, the outline of the human face has already been presented in the generated images. For the number 8 in MNIST, WGAN-GP, SAGAN, and MA-GAN all outline the number after the generators of these models execute 100 times. However, there are still irregular shapes in WGAN-GP and SAGAN. By contrast, the MA-GAN has a relatively stable performance, and there are not many abnormal structures in the generated images. For CIFAR-10, the images generated by the three models after 1000 iterations are challenging to distinguish with eyes, and there will be a quantitative evaluation later.

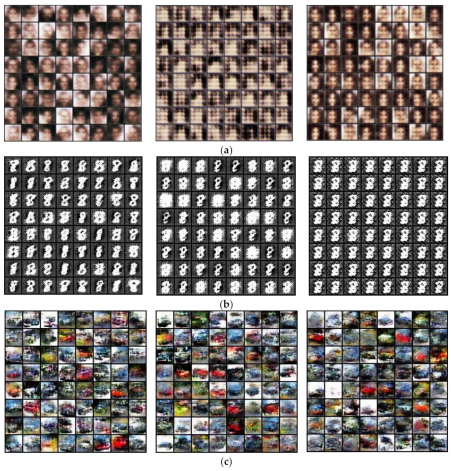


Figure 7. The samples generated by different models. (**a**) WGAN-GP, (**b**) SAGAN, (**c**) MA-GAN.

Figure 8 shows that compared with SAGAN, the images generated by MA-GAN are more slippery. Meanwhile, the edges of the images generated by WGAN-GP have a sharp sawtooth pattern. SAGAN has more abnormal images. The reason is that SAGAN cannot capture the connections between channels. It cannot integrate the dependencies between various categories or capture all the geometric features and structures of images.

It can be seen from Figure 9 that the MA-GAN can generate relatively smooth images for complex handwritten numbers such as 2, 3, 5, and 8, and there is no noticeable noise in the generated images. However, abnormal structures can be observed in the images generated by WGAN-GP and SAGAN for the handwritten numbers 4, 5, and 8.

It can be seen from Figure 10 that there are apparent content and background segmentation in the images randomly generated by the MA-GAN. Meanwhile, it is difficult to distinguish the subjects in the images generated by WGAN-GP and SAGAN. By contrast, the images generated by MA-GAN can distinguish “aircraft”, “deer”, “trucks”, and other objects. In addition, the bright saturation of the colorful images is better than those generated by SAGAN. The images generated by WGAN-GP have much noise and different degrees of structural abnormalities on the objects such as “cats”, “frogs”, “ships”, and “trucks”, while these problems are rare in the MA-GAN.

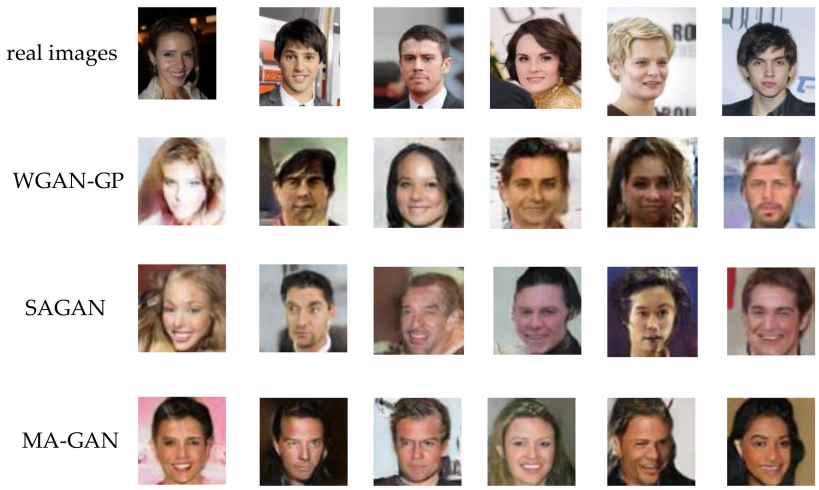


Figure 8. CelebA generated by different models.

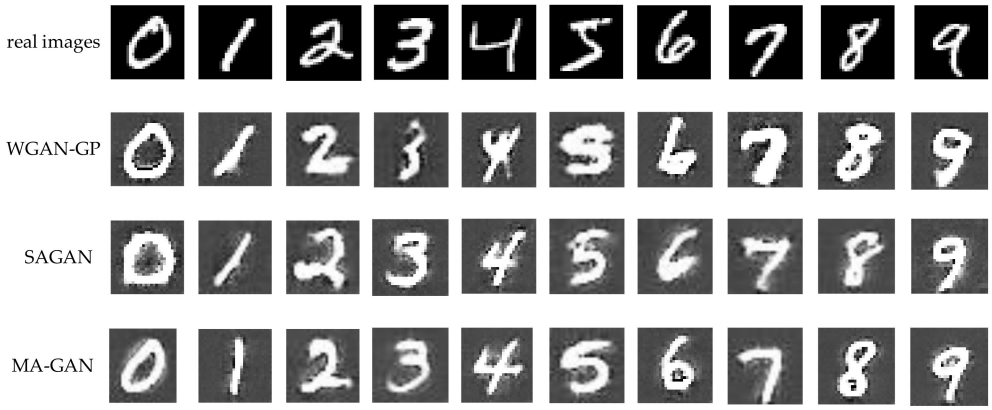


Figure 9. Comparison of the MNIST generated by different models.

Figure 10. Comparison of CIFAR-10 generated by different models.

Since MNIST contains single-channel images and IS quantitatively evaluates three-channel images, this paper directly uses classified accuracy instead of IS for quantitative evaluation of the generated images.

After 2000 images are randomly generated by the pre-trained model, ten IS average evaluations are performed on these images. The larger the value is, the better the quality and diversity of the generated images are, as shown in Table 8.

Table 8. The IS of different models on CelebA.

Model	IS
WGAN-GP	2.189
SAGAN	2.238
MA-GAN	2.315

It can be seen from Table 8 that for CelebA, compared with WGAN-GP, the IS of MA-GAN is increased by 5.76% and that of SAGAN with the same attention mechanism is increased by 3.44%.

As shown in Table 9, for CIFAR-10, the IS of MA-GAN is the highest for all classes. For the class of truck, compared to WGAN-GP, the IS of MA-GAN is increased by 12.16%, and that of SAGAN with attention mechanism is increased by 3.26%.

Table 9. The IS of different models on CIFAR-10.

Class	WGAN-GP	SAGAN	MA-GAN
air-plane	3.738	3.756	3.958
automobile	3.156	3.273	3.552
bird	3.018	3.042	3.374
cat	2.990	2.971	3.066
deer	2.491	2.627	2.920
dog	3.354	3.523	3.635
frog	2.496	2.506	2.594
horse	3.426	3.619	3.755
ship	3.206	3.073	3.273
truck	2.853	3.099	3.200

4.5. Classification Performance Analysis

It can be seen from Figure 11 that under the same iteration steps, the training loss of the MA-GAN with DA decreases the fastest, and loss is close to 0 after 50 iterations, indicating that the model converges easily.

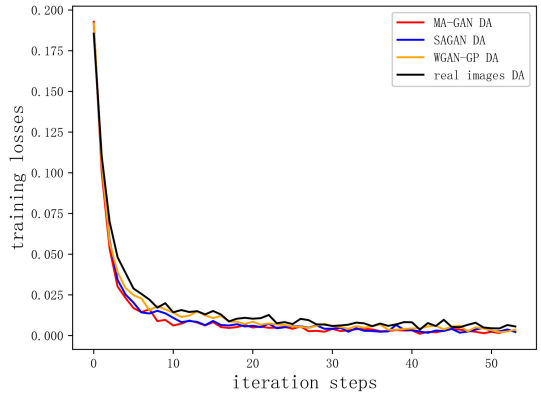
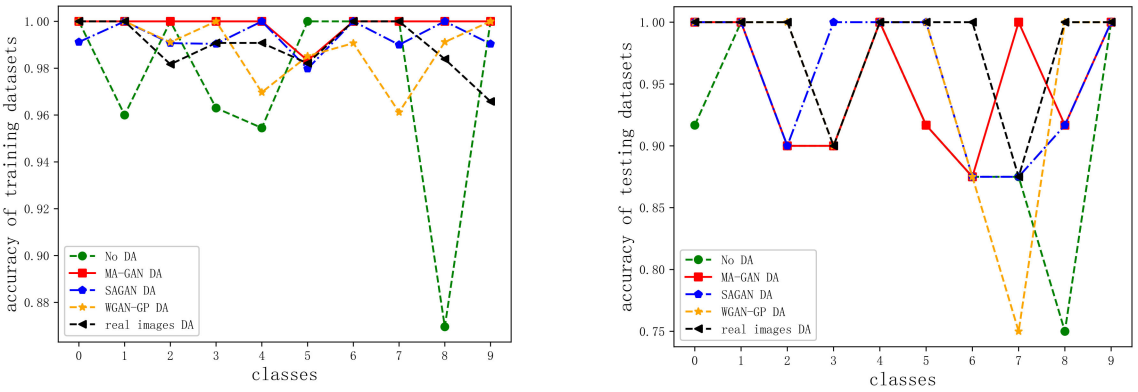


Figure 11. The change of training loss after MNIST augmentation.

Figure 12 shows that without augmentation, the classification accuracy of each class in the training set and test set fluctuates the most. However, the WGAN-GP augmentation presents greater fluctuations than other methods. The classification accuracies of the classes with SAGAN augmentation and real images augmentation are equivalent, while that with MA-GAN augmentation shows the slightest fluctuation.



(**a**) (**b**)

Figure 12. Classified accuracy of the training set and testing set (**a**) before and (**b**) after MNIST augmentation.

It can be seen from Figure 13 that under the same iteration steps, the training loss of the MA-GAN DA decreases the fastest, and the loss becomes stable after 400 iterations, indicating that the model converges easily. After real images augmentation, the training loss does not reduce as fast as that of the GAN DA.

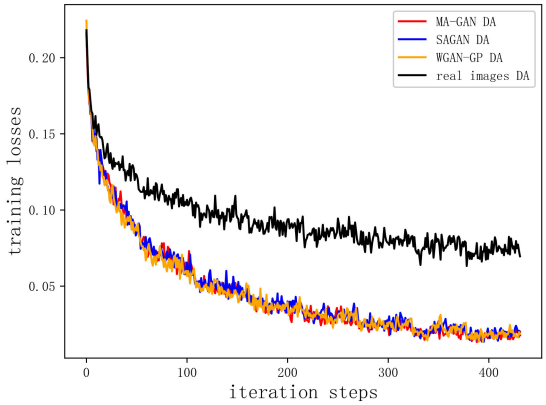
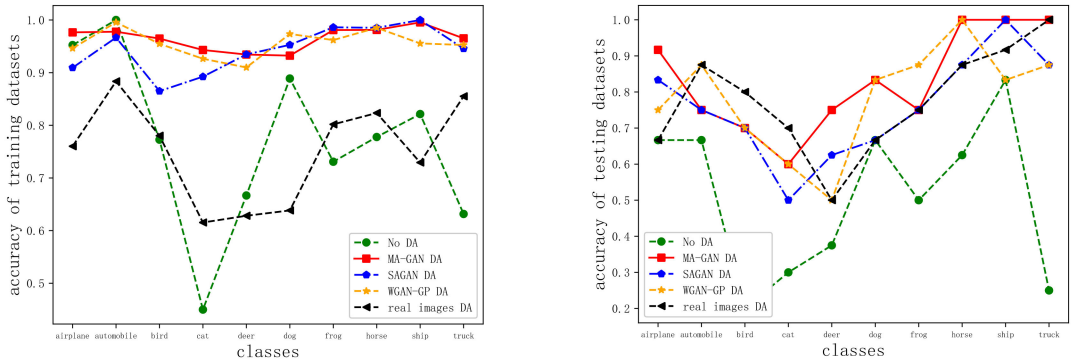


Figure 13. The change of the training loss after CIFAR-10 augmentation.

Figure 14 shows that without augmentation, the classification accuracy of each class fluctuates the most. However, real images augmentation leads to greater fluctuations than other methods. The classification accuracy of each class with SAGAN augmentation and real images augmentation is equivalent, while the flux of classification accuracy is the smallest after the MA-GAN augmentation is used. Moreover, the classification accuracy in the test set reaches 100% for the three classes of "truck", "boat", and "horse" after the MA-GAN is used.

As shown in Table 10, after the MA-GAN is used for augmentation, the accuracy of the remaining numbers is 100% except for the complex number "5" in the training set. Compared with real images, SAGAN, and WGAN-GP augmentation, the MA-GAN achieves the highest average accuracy.



(**a**) (**b**)

Figure 14. Classified accuracy of the training set and testing set (**a**) before and (**b**) after CIFAR-10 augmentation.

Table 10. Classified accuracy of MNIST training set after augmentation by different methods (%).

Class	No DA	Real Images DA	WGAN-GP DA	SAGAN DA	MA-GAN DA
0	100.00	100.00	100.00	99.12	100.00
1	96.00	100.00	100.00	100.00	100.00
2	100.00	98.17	99.11	99.07	100.00
3	96.30	99.08	100.00	99.04	100.00
4	95.45	99.08	96.97	100.00	100.00
5	100.00	98.21	98.52	97.98	98.33
6	100.00	100.00	99.07	100.00	100.00
7	100.00	100.00	96.12	99.00	100.00
8	86.96	98.39	99.12	100.00	100.00
9	100.00	96.58	100.00	99.04	100.00
average accuracy	97.06	98.51	99.02	99.64	99.71

It can be seen from Table 11 that the augmentation of MA-GAN achieves the highest average accuracy.

Table 11. Classified accuracy of MNIST training set after augmentation by different methods (%).

Class	No DA	Real Images DA	WGAN-GP DA	SAGAN DA	MA-GAN DA
0	91.67	100.00	100.00	100.00	100.00
1	100.00	100.00	100.00	100.00	100.00
2	90.00	100.00	100.00	90.00	90.00
3	90.00	90.00	90.00	100.00	90.00
4	100.00	100.00	100.00	100.00	100.00
5	91.67	100.00	100.00	100.00	91.67
6	87.50	100.00	87.50	87.50	87.50
7	87.50	87.50	75.00	87.50	100.00
8	75.00	100.00	100.00	91.67	91.67
9	100.00	100.00	100.00	100.00	100.00
average accuracy	94.07	97.87	96.27	96.27	96.73

Table 12 shows that compared with SAGAN and WGAN-GP, MA-GAN achieves the highest average accuracy.

Table 12. Classified accuracy of CIFAR-10 training set after augmentation by different methods (%).

Class	No DA	Real Images DA	WGAN-GP DA	SAGAN DA	MA-GAN DA
airplane	95.24	76.72	94.63	90.95	97.65
automobile	100.00	88.35	99.56	96.67	97.78
bird	77.27	78.02	95.48	86.50	96.48
cat	45.00	61.54	92.65	89.24	94.31
deer	66.67	62.82	90.99	93.47	93.44
dog	88.89	63.84	97.35	95.28	93.24
frog	73.08	80.18	96.19	98.61	98.10
horse	77.78	82.33	98.52	98.52	98.12
ship	82.14	72.94	95.55	100.00	99.55
truck	63.16	85.51	95.26	94.57	96.52
average accuracy	72.31	76.34	95.27	94.80	96.15

Table 13 shows that the MA-GAN achieves the highest average accuracy compared with real images, SAGAN, and WGAN-GP augmentation.

Table 13. Classified accuracy of CIFAR-10 training set after augmentation by different methods (%).

Class	No DA	Real Images DA	WGAN-GP DA	SAGAN DA	MA-GAN DA
airplane	66.67	66.67	75.00	83.33	91.67
automobile	66.67	87.50	87.50	75.00	75.00
bird	20.00	80.00	70.00	70.00	70.00
cat	30.00	70.00	60.00	50.00	60.00
deer	37.50	50.00	50.00	62.50	75.00
dog	66.67	66.67	83.33	66.67	83.33
frog	50.00	75.00	87.50	75.00	75.00
horse	62.50	87.50	100.00	87.50	100.00
ship	83.33	91.67	83.33	100.00	100.00
truck	25.00	100.00	87.50	87.50	100.00
average accuracy	51.20	75.93	74.33	76.20	79.47

5. Time Efficiency Discussion and Complexity Analysis

Table 14 presents the computational cost comparison of the models on a Nvidia's RTX 2080 Super GPU. For CelebA, the training time of MA-GAN increased by 6.8 h compared to WGAN-GP, while it increased by 4 h compared to SAGAN. For CIFAR-10, the training time of MA-GAN increased by 3.2 h compared to WGAN-GP, while it increased by 0.9 h compared to SAGAN. For MNIST, there is no significant difference in the training time between the three models.

Table 14. Computational cost comparison of the models.

Dataset		WGAN-GP	SAGAN	MA-GAN
CelebA	epoch	200,000	200,000	200,000
	data size	100,000	100,000	100,000
	training time (hour)	5.2	8	12
CIFAR-10	epoch	200,000	200,000	200,000
	data size (each class)	500	500	500
	training time (hour)	11.9	15.1	16
MNIST	epoch	2000	2000	2000
	data size (each class)	500	500	500
	training time (hour)	0.13	0.17	0.2

The proposed MA-GAN is unconditional GAN. When the dataset includes many classes, different classes need to be input into the model to generate images so as to meet the needs of DA. This may take a lot of time, and it is a limitation of the proposed method. However, the experimental results of and model comparison show that the proposed method can improve the quality of generated images and then meet the needs of DA.

6. Conclusions

This paper proposes MA-GAN to augment small samples. The proposed method consists of two modules. These two modules respectively deal with local features and global dependencies. Meanwhile, the relationship between each channel and the spectral normalization technique is introduced to the proposed method. Through experiments on CelebA, CIFAR-10, and MNIST, it is verified that the MA-GAN performs better than other methods in the standard IS metric and classification applications. The proposed method improves the quality of the generated images and speeds up network convergence. In addition, the classification accuracy of the classifier with MA-GAN is better than other comparison methods, indicating the effectiveness of the proposed method for DA. However, this method still has some shortcomings: the highest resolution of generated samples by this method is 64×64; we will focus on generating higher resolution images for DA tasks in the future.

Author Contributions: Conceptualization, L.S. and X.M.; methodology, Y.Y.; software, Y.Y.; validation, Y.Y.; formal analysis, Y.Y.; investigation, Y.Y.; resources, L.S. and X.M.; data curation, Y.Y.; writing—original draft preparation, Y.Y.; writing—review and editing, L.S., X.M. and M.Z.; visualization, Y.Y.; supervision, L.S.; project administration, L.S. and X.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: This study did not involve humans or animals.

Informed Consent Statement: This study did not involve humans.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Hong, D.; He, W.; Yokoya, N.; Yao, J.; Gao, L.; Zhang, L.; Chanussot, J.; Zhu, X. Interpretable hyperspectral artificial intelligence: When nonconvex modeling meets hyperspectral remote sensing. *IEEE Geosci. Remote Sens. Mag.* **2021**, *9*, 52–87. [CrossRef]
2. Hong, D.; Gao, L.; Yao, J.; Zhang, B.; Plaza, A.; Chanussot, J. Graph convolutional networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 5966–5978. [CrossRef]
3. Russo, F. An image enhancement technique combining sharpening and noise reduction. *IEEE Trans. Instrum. Meas.* **2002**, *51*, 824–828. [CrossRef]
4. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–14 December 2014; pp. 2672–2680.
5. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 214–223.
6. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved training of wasserstein gans. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–6 December 2017.
7. Weaver, N. *Lipschitz Algebras*; World Scientific: Singapore, 2018.
8. Miyato, T.; Kataoka, T.; Koyama, M.; Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–30 May 2018.
9. Fukushima, K. A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* **1980**, *36*, 193–202. [CrossRef] [PubMed]
10. Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
11. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.

12. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 4401–4410.
13. Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; Aila, T. Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 8110–8119.
14. Shaham, T.R.; Dekel, T.; Michaeli, T. Singan: Learning a generative model from a single natural image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, CA, USA, 16–20 June 2019; pp. 4570–4580.
15. Hinz, T.; Fisher, M.; Wang, O.; Wermter, S. Improved techniques for training single-image gans. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Virtual, 5–9 January 2021; pp. 1300–1309.
16. Menon, S.; Damian, A.; Hu, S.; Ravi, N.; Rudin, C. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 2437–2445.
17. Huang, S.W.; Lin, C.T.; Chen, S.P.; Wu, Y.; Hsu, P.; Lai, S. Auggan: Cross domain adaptation with gan-based data augmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 718–731.
18. Antreas, A.; Amos, S.; Edwards, H. Data Augmentation generative adversarial networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 16 February 2018.
19. Frid-Adar, M.; Klang, E.; Amitai, M.; Jacob Goldberger, H.G. Synthetic data augmentation using GAN for improved liver lesion classification. In Proceedings of the 2018 IEEE 15th International Symposium on Biomedical Imaging, Washington, DC, USA, 8 January 2018; pp. 289–293.
20. Zheng, Z.; Yang, X.; Yu, Z.; Zheng, L.; Yang, Y.; Kaut, j. Joint discriminative and generative learning for person re-identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 2138–2147.
21. Li, L.; Li, Y.; Wu, C.; Dong, H.; Jiang, P.; Wang, F. Detail Fusion GAN: High-Quality Translation for Unpaired Images with GAN-based Data Augmentation. In Proceedings of the 25th International Conference on Pattern Recognition, IEEE, Milan, Italy, 10–15 January 2021; pp. 1731–1736.
22. Li, X.; Du, Z.; Huang, Y.; Tan, Z. A deep translation (GAN) based change detection network for optical and SAR remote sensing images. *ISPRS J. Photogramm. Remote Sens.* **2021**, *179*, 14–34. [CrossRef]
23. Yu, X.; Wu, X.; Luo, C.; Ren, P. Deep learning in remote sensing scene classification: A data augmentation enhanced convolutional neural network framework. *GIScience Remote Sens.* **2017**, *54*, 741–758. [CrossRef]
24. Liang, B.; Liu, Q.; Xu, J. Target specific emotion analysis based on multi attention convolutional neural network. *Comput. Res. Dev.* **2017**, *54*, 1724–1735.
25. Zhu, Z.; Rao, Y.; Wu, Y. Research progress of attention mechanism in deep learning. *Chin. J. Inf.* **2019**, *33*, 1–11.
26. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
27. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
28. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–14 June 2019; pp. 7354–7363.
29. Yang, Y.; Sun, L.; Mao, X.; Dai, L.; Guo, S.; Liu, P. Using Generative Adversarial Networks Based on Dual Attention Mechanism to Generate Face Images. In Proceedings of the 2021 International Conference on Computer Technology and Media Convergence Design IEEE, Sanya, China, 23–25 April 2021; pp. 14–19.
30. Liu, Z.; Luo, P.; Wang, X.; Tang, X. Large-Scale Celebfaces Attributes (Celeba) Dataset. Available online: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html (accessed on 21 February 2022).
31. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Master's Thesis, The Pennsylvania State University, State College, PA, USA, 2009.
32. Kingma, P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
33. Barratt, S.; Sharma, R. A note on the inception score. *arXiv* **2018**, arXiv:1801.01973.