IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 September 2022, accepted 5 October 2022, date of publication 10 October 2022, date of current version 17 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3213079

RESEARCH ARTICLE

Image Creation Based on Transformer and Generative Adversarial Networks

HANGYU LIU AND QICHENG LIU
School of Computer and Control Engineering, Yantai University, Yantai, Shandong 264005, China
Corresponding author: Qicheng Liu (ytliuqc@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 62172351.

ABSTRACT To address the problem of low authenticity of generated images in existing generative models, the transformer super-resolution generative adversarial network(TransSRGAN) model based on the generative adversarial network is proposed. The generator of the model uses the transformer encoder sub-module as the basic module. The features of the input vector are extracted. low-definition images are generated through the transformer encoder submodule, and the low-definition image is up-sampled by the convolutional neural network to complete the image generation. The discriminator of this model uses the convolutional neural network as the basic module. To discriminate the real samples from the generated fake samples, the discriminator extracts the image features by the convolutional neural network. The experimental results show that the TransSRGAN model brings the distribution of the generated samples closer to the training samples, effectively raises the quality of the generated samples, improves the authenticity of the generated samples, and enriches the diversity of the generated samples. During the training process, there was no mode collapse or instability.

INDEX TERMS Image generation, generative adversarial network, transformer, self-attention.

I. INTRODUCTION

The generative model is a sort of algorithm that can learn reusable features in large unlabeled datasets and generate data that do not exist in the dataset. Generative models have been focus of research in recent years. The earliest generative model is the variational autoencoder [1] based on variational inference and Bayesian theory. Variational autoencoder can generate not only pictures, but also text [2] and audio [3]. Although the variational autoencoder is simple and effective, it tends to generate noisy data irrelevant to the trainset owing to the assumption of a simple normal distribution as the original sample distribution.

With the development of deep learning, its application to generative models has begun. Deep learning not only has a better learning ability for the characteristics of the data set but also has a better ability to fit the real data distribution of the samples. The generative adversarial network (GAN) [4] based on deep learning has an excellent generation effect in the generative model. GANs can utilize a large number of unlabeled samples to learn good intermediate feature representations of samples. Because GAN has excellent adaptability to samples, it can be suitable for various generative tasks, such as the generation of videos [5], images [6], and audio [7] with different styles. In recent years, generative adversarial networks have been widely used in the energy field, such as energy scheduling [8], [9].

With the development of computer vision, increasing number of generative models have become accustomed to generating images. Because the network structure of GAN determines its ability to extract the features of the sample, a reasonable network structure can make GAN generate more real and high-definition pictures. The exploration of a reasonable GAN structure has become a research hotspot today [10].

II. RELATED WORK

The variational autoencoder [1] proposed by King Ma et al. first learns reusable features in large unlabeled images. It generates images that do not exist in the dataset, but the generated images are relatively blurry. Owing to the development of deep learning, GANs can generate clearer images than

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Li.

variational autoencoders. The GAN [4] was proposed by Ian Goodfellow of the University of Montreal in 2014, which is a machine learning architecture. It is a neural network based on the game theory minimax algorithm. The GAN consists of a discriminator and a generator. The purpose of the generator is to maximize the realism of fake samples. The purpose of the discriminator is to attempt to discriminate between real and the fake samples. During the training process, the discriminator and generator continue to converge to the optimal state.

Because GAN is an unsupervised learning model, it cannot extract the features of labeled data. For labeled data, researchers have proposed many GANs with labeled classifications, such as CGAN [11] and ACGAN [12]. ACGAN is a generative model proposed by Odena et al. in 2016 which can generate images based on tags. ACGAN performs supervised learning on labeled data by fusing the classification tags into a loss function.

Although a GAN can effectively generate images, it is difficult to train [13]. When the generating adversarial network is trained, two situations occur. First, if the distributions of the real and generated samples do not overlap, the gradient of the generator is always 0. This causes the generator not to update. Second, the generator tends to generating repeated and safe samples, leadings to mode collapse. One solution is to use methods based on integral probability metrics, such as the Wasserstein distance (WGAN) [14], kernel MMD [15], and Cramer distance [16]. Another methold is to add a gradient penalty term to maintain stabilization during the training of GANs [16], [17], [18]. Among them, DRAGAN [18] is a method that adds a gradient penalty term to maintain stabilization in the training of GANs. DRAGAN was proposed by Kodali et al.The DRAGAN stabilizes the GAN training by adding a gradient penalty to the discriminator. It can be observed that restricting the discriminator D(x) to a Lipschitz continuity for K. Compared with WGAN, the training of DRAGAN is more stable, and the update direction is the same as the gradient direction when the momentum-based optimization algorithm is used for training.

The generator and discriminator of GAN can be constructed using by various networks. The DCGAN [19], proposed by Alec Radford, uses a multilayer convolutional neural network (CNN) to build a GAN. The DCGAN dramatically improves the quality and style richness of the generated adversarial network results compared to the GAN containing the multilayer perceptron. However, there are still some problems with DCGAN. First, the generated samples deviated from the real sample distribution. Second, the generator is prone to collapse during training. To solve the problem of generated samples deviating from the distribution of real samples, StyleGan [20] based on style is proposed by T Karras et al. StyleGan introduced a style network to generate images through feature fusion after learning the features of style images. StyleGan makes the distribution of generated samples and real samples closer, but there are still some defects, such as a single generation style and complex network structure. Because the input of StyleGan is not a random vector, but a styled vector, StyleGan does not generate images from scratch. It is closer to a style transfer. However, This method is not suitable for images with changing styles.

SRResNet [21], based on DCGAN, was proposed by Y Jin et al. In contrast of the vanilla DCGAN, the generator and discriminator of this network introduced the residual neural network ResNet [22] proposed by K He. Moreover, the CNNs of the network generator and discriminator wear deepened. The SRResNet generator can generate images with a higher resolution, and the discriminator has a stronger discriminative ability. Compared with the vanilla DCGAN, the model can generate more realistic images. However, the model still has the problem of the low authenticity of the generated samples.

At present, generative models based on GAN have the problem that the generated samples deviate from the original samples. In order to make the generated samples closer to the original samples, a GAN consisting of Transformer Encoder and CNN is proposed. This network can effectively minimize the distance between the generated sample distribution and the original sample distribution without using a style transfer. This method can generate more realistic images. Compared with the existing models, this model adopts super-resolution technology after the transformer generator generates the images. Compared with DCGAN, the model in this study has more realistic generated samples. Compared to a GAN composed of a full transformer, the proposed model requires less computation.

III. RELATED CONCEPTS

A. CONVOLUTIONAL NEURAL NETWORK

The convolutional neural network (CNN) refers to a neural network that uses convolution operations for feature extraction. A general CNN consists of a pooling layer, a normalization layer, a convolutional layer, and a fully connected layer. Each layer is usually followed by an activation function.

The convolutional layer refers to a neural network composed of convolution operations. The convolution operations are obtained by sliding the convolution kernel and multiplying and adding the corresponding image pixels. The convolution operation can be explained as follows:

$$Conv(X) = w \cdot X[i : i + h - 1] + b \tag{1}$$

where b is the bias parameter, w is the weight matrix of $h \times k$ dimension, X[i:i+h-1] represents the i-th row to the i+h-1th row of the matrix X, and the size is $h \times k$ the convolution kernel.

Generally, a CNN consists of multiple pooling layers and convolutional layers. Which can be explained as:

$$\begin{cases} Conv_{n+1}(x) = Conv(f_n(Conv_n(x))) & n \in \{1, 2, \ldots, m\} \\ MultiConv_n(X) = Conv(f_n(Conv_n(x))) & n = m - 1 \end{cases} \tag{2}$$

where $f(x)$ is the normalization function or activation function or pooling function, n is the number of iterations.

Normalization can reduce the influence of distribution changes caused by parameter updates in CNN, and map the data distribution to a specific interval. Normalization can be explained as:

$$BatchNorm(x) = \frac{x - E_b[x]}{\sqrt{Var_b[x] + \varepsilon}} * \gamma + \beta \tag{3}$$

$$LayerNorm(x) = \frac{x - E_l[x]}{\sqrt{Var_l[x] + \varepsilon}} * \gamma + \beta \tag{4}$$

where Var is the standard deviation, E is the expectation, ε is the value prevent the denominator from being 0, and β and γ are learnable parameters.

BatchNorm calculates the expectation and standard deviation on the input batch dimension. LayerNorm calculates the expectation and standard deviation on the input channel dimension.

The activation function can de-linearize the linearly transformed feature matrix, and the commonly used activation functions are ReLU, tanh, sigmoid, etc. These activation functions can be explained as:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{6}$$

$$ReLu(x) = \max(0, x) \tag{7}$$

where max is the maximum value.

B. GENERATIVE ADVERSARIAL NETWORK

Generative adversarial network (GAN) can be explained as:

$$\begin{aligned} &\min_G \max_D L(D, G) \\ &= E_{z\sim P_z(z)}[\log(1 - D(G(z))) + E_{x\sim P_{data}(x)}[\log D(x)] \end{aligned} \tag{8}$$

where G refers to the generator, D refers to the discriminator, z refers to the noise vector, x refers to the real data, E refers to the expectation, and P refers to the distribution function. Y Jin proposed a new network SRResNet based on DCGAN. The network can be represented as:

$$\begin{cases} H(x) = F(x) + x \\ F(x) = MultiConv_n(x) \\ D(x) = sigmoid(MultiConv_n(H(x))) \\ G(x) = tanh(MultiConv_n(H(x))) \end{cases} \tag{9}$$

where H(x) is the residual neural network. ACGAN is a generative model that can generate images based on tags. For the sample set, using this model needs not only the sample image but also the class tag corresponding to each sample, and the loss function can be explained as:

$$L_s = E_{s\sim real}[\log P(S|X_{real})] + E_{s\sim fake}[\log P(S|X_{fake})] \tag{10}$$

$$L_{cls} = E_{c\sim class}[\log P(C|X_{real})] + E_{c\sim class}[\log P(C|X_{fake})] \tag{11}$$

where L_s is the true and false discrimination loss, L_{cls} is the classification loss, P is the distribution function, and E is the expectation. DRAGAN is a stable training method. The loss function of DRAGAN can be explained as:

$$\begin{aligned} L_D = &-E_{x\sim P_{data}}[\log D(x)] \\ &+ \lambda E_{x\sim P_{data}, \sigma\sim N(0,cl)}[\|\nabla_x D(x + \sigma)\| - K]^2 \\ &- E_{z\sim P_z}[\log(1 - D(G(z)))] \end{aligned} \tag{12}$$

$$L_G = E_{z\sim P_z}[\log(D(G(z)))] \tag{13}$$

where ∇ represents the gradient, N is the normal distribution,cl and λ is a parameter, and K is the gradient penalty parameter.

C. TRANSFORMER

Transformer [23] neural network was proposed by Ashish Vaswani et al. and originally was used for natural language processing. The Transformer is composed of Encoder and Decoder. The Encoder is responsible for encoding the data into hidden vectors.The Decoder is responsible for decoding the hidden vector into data. The Transformer Encoder consists of two blocks. The first block is a multi-head self-attention block. The second block is a feed-forward fully connected network with Relu activation function. Normalization is used before the two blocks. Both blocks have residual network links. The Transformer Decoder has three blocks. The first block is a multi-head self-attention block. The second block is a multi-head attention block that extracts the relationship between latent vectors and input attention. The third block is a feed-forward fully connected network with Relu activation function. Normalization was used before the two blocks. All three blocks have residual network links.

D. SELF-ATTENTION

The self-attention was first proposed by Ashish Vaswani et al. and applied in Transformer. The difference between the self-attention and the convolution operation is a range of receptive field. The receptive field of the convolution operation is the local receptive field. In contrast, the receptive field of the self-attention is the global receptive field. The self-attention describes the dependency between any two data. It can be regarded as a particular case of embedding Gaussian [24]. The input data is linearly transformed into three matrices Q, K, and V. After that, Q and K are dot-multiplied and then divided by the square root of the scaling factor to calculate Softmax, and finally dot-multiplied with the matrix V. The self-attention can be explained as:

$$Q = DW_q, K = DW_k, V = DW_v \tag{14}$$

$$softmax(X)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{15}$$

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{16}$$

where W_q, W_k, W_v are trainable parameters, d_k is the scaling parameter, and D is the input data. Using a single self-attention can only make the model extract the feature information of one space. Using multi-head attention can

make the model extract the feature information of multiple subspaces. The multi-head self-attention can be explained as:

$$\begin{cases} head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \\ MultiHead(Q, K, V)_h = Concat(head_1, \ldots, head_h)W^0 \end{cases} \tag{17}$$

where W^0, W^Q, W^K, W^V are trainable parameters, and Concat is matrix splicing. The multi-head attention is the result of linear transformation after splicing multiple attention.

IV. TRANSFORMER-BASED GENERATIVE MODEL TransSRGAN

In order to minimize the distance between the generated sample distribution and the original sample distribution, the TransSRGAN model based on GAN is proposed. Different from vanilla GAN, the generator of TransSRGAN uses Transformer Encoder to build sub-modules and uses CNN sub-modules for upsampling operations. The discriminator of TransSRGAN is constructed using a CNN as a sub-module. The CNN submodule used by TransSRGAN uses only normalization and convolutional layers and uses an activation function for delinearization.

TransSRGAN can be explained as:

$$\begin{cases} D(x) = MultiConv_n(MultiConv_n(x) + x) \\ G(x) = tanh(MultiConv_n(Transformer_n(x) + x)) \end{cases} \tag{18}$$

where Transformer refers to Transformer Encoder (Figure (2)), MultiConv refers to the calculation of multiple convolutions by formula (2), and tanh refers to the calculation of hyperbolic tangent by formula (6).

The network structure of the generator is shown in Figure (1), and the structure of the Transformer Encoder is shown in Figure (2). In Figure (1), s is the convolution stride, k is the convolution kernel dimension, and n is the number of output channels. The generator contains 16 layers of Transformer Encoder. Each Transformer Encoder includes a multi-head self-attention layer composed of four self-attentions. Constructing generative adversarial networks entirely with Transformer Encode will result in extremely high memory consumption. Therefore, three-layer CNNs are used to upsample the 16*16 image generated by the Transformer Encoder. First, the generator inputs a 162-dimensional noise vector consisting of 128-dimensional random floating-point numbers and 34-dimensional classification tags. The random floating-point numbers are randomly generated from a normal distribution with an expectation of 0 and a standard deviation of 1. 34-dimensional one-hot encoding was used for the classification tags. Then, the noise vector passes through a fully connected network to output a feature vector with a dimensions of 64*16*16. After the normalized feature vector, a 16*16*64 image is generated through a 16-layer Transformer Encoder and normalization, then upsampled through a three-layer CNNs, and the final output is 128*128*3 image. A residual neural network was used to abstract the shallow features. It can not only solve the problem of vanishing gradients or exploding gradients but also improve the extraction of shallow network features.

Since the calculation of the large-dimensional attention matrix will consume a lot of computing resources, the network will divide the data into 64 sequences for calculation, and the length of each sequence is 256. In this manner, only a 64*64 attention matrix will be generated. The sequence can output a 16*16*64 image after going through multiple layers of the Transformer Encoder. Since Transformer Encoder outputs larger images, the memory footprint increases, so CNN is used for upsampling. Under normal circumstances, the convolution operation, which is a downsampling operation, reduces the height and width of the feature matrix. However, the Pixel Shuffle algorithm [25] can obtain the feature matrix of r2 channels through the convolution operation and then perform upsampling through the method of periodic screening. It can increase the resolution of the output image by a factor of r compared to the input image. After using the above methods, the memory occupation and calculation amount of the algorithm can be effectively reduced, and the calculation speed can be accelerated.

The generator is represented as Algorithm (1). Where W^1, W^2, W^3, W^4, W^q,W^k, W^v are the parameters of the generator. Before training, the initial values of the parameters of the generator are randomly selected from a normal distribution with the standard deviation of 0.02 and the expectation of 0. After training, the parameters of the generator will be fixed to the local optimal solution of the generator. The parameters were used after training when using the generator to generate images. The feature vector z is a 162-dimensional noise vector, of which the first 128 bits are randomly generated by a normal distribution with the expectation of 0 and the standard deviation of 1, and the last 34 bits are randomly generated by One-Hot encoding.

The discriminator is shown in Figure (3). Since the addition of the sigmoid activation function will cause the mode to collapse, the final sigmoid activation function in the discriminator based on ACGAN is canceled. The main structure of the discriminator is composed of a multi-layer CNNs. The input of the network is 3*128*128 images, and the output is a 35-dimensional vector composed of 34-dimensional tags and 1-dimensional true and false samples. If the output of the true and false sample flag is 0, it is discriminated as the generated sample; if the output is 1, it is discriminated as the original sample. The initial values of all parameters of the discriminator network obey the normal distribution with the standard deviation of 0.02 and the expectation of 0.

Since the vanilla generative adversarial network cannot perform feature extraction on the data with classification tags, the auxiliary classifier is included in the vanilla GAN loss function in the form of a multiplier. To solve the problem of mode collapse in the GAN, the gradient penalty term is included in the vanilla GAN loss function in the form of a multiplier. Through these two schemes, the GAN can extract the features of image classification tags while stably training.

Algorithm 1 Generator

Input: feature vector z, parameters W^1, W^2, W^3, W^4, W^q,W^k, W^v

Output: image im

$O_1 \leftarrow W_{1z}$
{Get the feature matrix O_1 through the fully connected network}
$O_2 \leftarrow BatchNorm(Relu(O_1))$
{Using formula (3) and formula (7) to normalize O_1 to obtain characteristic matrix O_2}
$O_{temp} \leftarrow W_2$
for i from 1 to 16 **do** {Use 16 layer Transformer Encoder to generate small image}
  $O_3 \leftarrow LayerNorm(O_2)$
  {Use formula (4) for normalization to get the feature matrix O_3}
  $Q_i, K_i, V_i \leftarrow O_3W_{qi}, O_3W_{ki}, O_3W_{vi}$
  {Calculate Q, K, V using formula (14)}
  $O_4 \leftarrow W_2MultiHead(Q_i, K_i, V_i)^4 + O_2$
  {Use formula (17) to calculate the attention, and calculate the feature matrix O_4 through a fully connected layer}
  $O_5 \leftarrow W_4Relu(W_3LayerNorm(O_4)) + O_4$
  {Use formula (4) for normalization and then use formula (7) to calculate the feature matrix O_5}
  $O_2 \leftarrow O_5$
  {Store the result of one iteration to O_2 and go to the next iteration}
end for
$O_6 \leftarrow BatchNorm(Relu(O_5)) + O_{temp}$
{Normalize the output small image O_5 using formula (3) and formula (7) to obtain the feature matrix O_6}
for i from 1 to 3 **do** {Upsampling using a 3-layer deep CNNs }
  $O_7 \leftarrow BatchNorm(Relu(PixelShuffle(Conv(O_6))))$
  {Use formula (1), formula (3), formula (7) and PixelShuffle algorithm to continuously upsample. get high resolution image O_7}
  $O_6 \leftarrow O_7$
  {Store the result of one iteration to O_6 and go to the next iteration}
end for
$im \leftarrow tanh(Conv(O_7))$
{Output image im through formula (6)}

The loss function of the network is shown in formula (19):

$$\begin{cases} L_G = \lambda_2 E_{c\sim class}[\log P(C|X_{fake})] + E_{z\sim P_z}[\log(D(G(z)))] \\ L_D = -E_{z\sim P_z}[\log(1 - D(G(z)))] \\ \quad - E_{x\sim P_{data}}[\log D(x)] \\ \quad + \lambda_1 E_{x\sim P_{data},\sigma\sim N(0,cl)}[\|\nabla_x D(x+\sigma)\| - K]^2 \\ \quad + \lambda_2\{E_{c\sim class}[\log P(C|X_{real})] \\ \quad + E_{c\sim class}[\log P(C|X_{fake})]\} \end{cases} \tag{19}$$

where, L_D is the discriminator loss, L_D item 1 is the real-fake loss, L_D item 2 is the classification loss, L_G is the discriminator loss, L_G item 1 and 2 are the real-fake loss, the third is the gradient penalty loss, and the fourth is the classification loss, λ_1 is the parameter of the gradient penalty formula (12, 13), λ_2 is the parameter of the gradient penalty formula (10, 11), z is the noise vector, x is the real data, σ is a random variable, E is the expectation, and P is the Distribution function.

The training model adopts the stochastic gradient descent method. Stochastic gradient descent is an optimization algorithm of batch, which can make the network parameters converge to the minimum value of the loss function through continuous iteration. Due to the limited memory size, it is impossible to process all the data in one iteration, so it is necessary to split the training samples into batches for training.

The training process of the network is shown in Algorithm (2). Where N is normal distribution, U is uniform distribution, BCE is binary cross-entropy, CE is cross-entropy, GradientDescent is gradient descent algorithm, grad is gradient, and GradClip is gradient clip.

Algorithm 2 Model Training

Input: real data r, real data tag r_l, real data quantity r_n, feature vector z, $D(x)$ is the discriminator, $G(x)$ is the generator, learning rate lr, number of epoch e, batch size b, gradient Cut parameter C, gradient penalty parameter K.

Output: new discriminator parameters $D_{\theta 2}$, new generator parameters $G_{\theta 2}$.

$D_\theta, G_\theta \sim N(0, 0.0004)$
{Initialize generator and discriminator parameters G_θ, D_θ}
for i from 1 to e **do**
  for i from 1 to r_n/b **do** {Optimize the discriminator:}
    $x, x_l \leftarrow D(r)$
    {Input real samples to the discriminator to get a 34-dimensional tag x_l and a 1-dimensional true and false sample mark x }
    $L_{rd} \leftarrow BCE(x, 1) + 0.02 * CE(r_l, x_l)$
    {After inputting the real sample, the loss function L_rd of the discriminator is obtained }
    $x, x_l \leftarrow D(G(z))$
    {After inputting the generated samples to the discriminator, the 34-dimensional tag x_l and the 1-dimensional true and false sample mark x}
    $L_{fd} \leftarrow BCE(x, 0) + 0.02 * CE(z_l, x_l)$
    {After inputting the generated samples, the loss function L_{fd} of the discriminator is obtained, where z_l is the last 34 dimensions of z}
    $x_r \sim U(0, 1)$
    {Generate random matrix x_r from uniform distribution}
    $L_d \leftarrow L_{rd} + L_{fd} + 0.5 * (\|grad(D(x_r), D_\theta)\| - K)$
    {Use formula (19.2) to calculate the loss function L_d of the discriminator}

$L_{dgrad} \leftarrow GradClip(grad(L_d, D_\theta), C)$
{Find the gradient of the discriminator through the loss function, and truncate the gradient of the discriminator to C }
$D_{\theta 2} \leftarrow GradientDescent(L_{dgrad}, lr)$
{Get a new discriminator parameter $D_{\theta 2}$ through gradient descent }
$D_\theta \leftarrow D_{\theta 2}$
{Update discriminator parameters D_θ }
{Optimization generator: }
$x, x_l \leftarrow D(G(z))$
{Input generated samples through the discriminator to get 34-dimensional tags and 1-dimensional true and false sample flags}
$L_g \leftarrow BCE(x, 1) + 0.02 * CE(z_l, x_l)$
{Calculate the loss function L_g of the generator using formula (19.1), where z_l is the last 34 dimensions of z}
$L_{ggrad} \leftarrow grad(L_g, G_\theta)$
{Find the gradient L_{ggrad} of the generator through the loss function}
$G_{\theta 2} \leftarrow GradientDescent(L_{ggrad}, l_r)$
{Get the new generator parameter $G_{\theta 2}$ by gradient descent}
$G_\theta \leftarrow G_{\theta 2}$
{Update generator parameters g_θ }
$lr \leftarrow lr * 0.1^{((rn*j+i*b)/50000)}$
{Update the learning rate lr }
end for
end for

V. EXPERIMENT AND RESULT ANALYSIS

In this chapter, the model SRResNet [21] proposed by Y Jin et al. is used as the baseline model, and experiments show that TransSRGAN has better generation ability.

A. DATASET

The training set of the model comes from crawling 43,740 pictures of anime characters from the Internet through crawler. These images come from different characters and have different resolutions. To intercept the avatar of the character, the Lbpcascade Animeface based on Adaboost is used to locate and crop the face of the character in the big picture. These face images need to be uniformly scaled to a size of 128×128.

Illustration2Vec is used to classify the samples in the training set. This is a CNN-based classifier that can add categorical tags to images (eg ''green hair'', ''red eyes'', etc.). To compare TransSRGAN with the baseline model, 34 tags consistent with the baseline model are selected for image classification. The classification tags are shown in Table (1).

TABLE 1. Tags.

Group	Tag
Hair Color	Pink, Brown, Blonde, Black, Green, Red, Blue, Purple, Silver, Orange, Aqua, White, Gray
Hair Length	Long Hair, Short Hair
Ponytail	Twintails, Ponytail
Drill	Hair True, False
Smile	True, False
Blush	True, False
Open Mouth	True, False
Hat	True, False
Ribbon	True, False
Glasses	True, False
Eye Color	Black, Brown, Blue, Pink, Purple, Green, Red, Yellow, Aqua, Orange

B. INCEPTION SCORE

The Inception Score (IS) is introduced to evaluate the definition and diversity of generative adversarial networks. In 2016, IS [26] was proposed by T Salimans et al., which calculates the KL divergence between the edge distribution of the classification vector and the classification vector through the pre-trained InceptionV3 neural network as a measure of the quality of the generated model. The representation is:

$$IS(x) = exp(E_{x \sim P_g} D_{KL}(p(y \mid x) \mid p(y))) \tag{20}$$

where x represents the generated sample, Y represents the classification vector, and Dkl is the KL divergence. The larger the value of IS, the clearer the generation effect IS and the more diverse the generated samples are.

C. FRÉCHET INCEPTION DISTANCE

To evaluate the performance of the samples generated by the GAN, the Fréchet Inception distance (FID) [27] is used to evaluate the realism of the samples generated by the GAN. FID was proposed by Martin Heusel et al. in 2017. It is calculated on the final output of the pre-trained InceptionV3 neural network, and the Fréchet distance of the approximate normal distribution of the real sample features and the generated sample features is calculated as a measure of the quality of the generated model. The representation is:

$$FID(x, g) = Tr(\Sigma_x + \Sigma_g - 2\sqrt{\Sigma_x \Sigma_g}) + \left\| \mu_x - \mu_g \right\|_2^2 \tag{21}$$

where x and g represent the generated sample and the real sample, Tr is the trace of the matrix, μ_x and μ_g represent the feature mean of the middle layer extracted by the generated sample and the real sample through the same InceptionV3, and Σ_x and Σ_g are the covariance matrix of the eigenvalues of the middle layer. The smaller value of FID is, the generated sample is closer to the real sample and better the generation effect.

FID is more sensitive to model collapse and also has better robustness to noise. If there is only one photo, the FID score will be very high. Therefore, FID can evaluate not only the similarity between the generated sample set and the real sample set, but also the diversity of the generated sample set.

D. HYPERPARAMETERS

Several groups of different hyperparameters are tested in the experiment by binary search. It is found that different

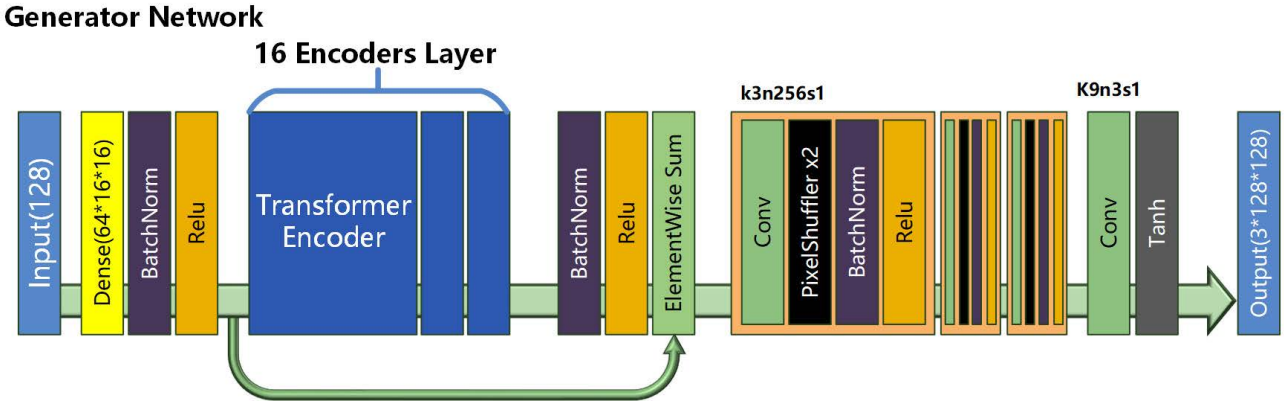

FIGURE 1. Generator Architecture.

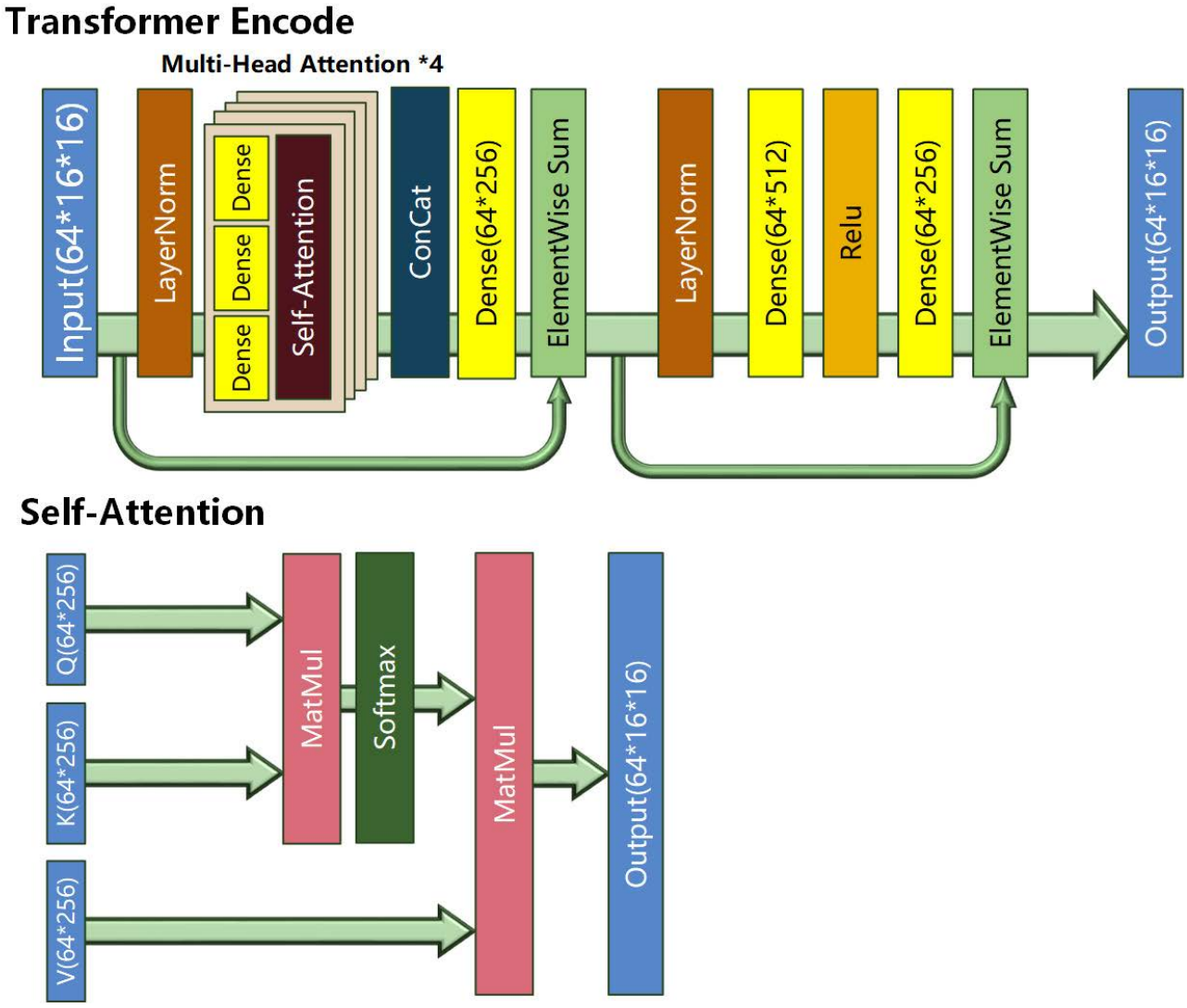

FIGURE 2. Transformer Encoder Architecture.

hyperparameter values have different results on the training of the model. A better set of hyperparameters is chosen for model training. Table (2) describes the hyperparameters used and lists their values.

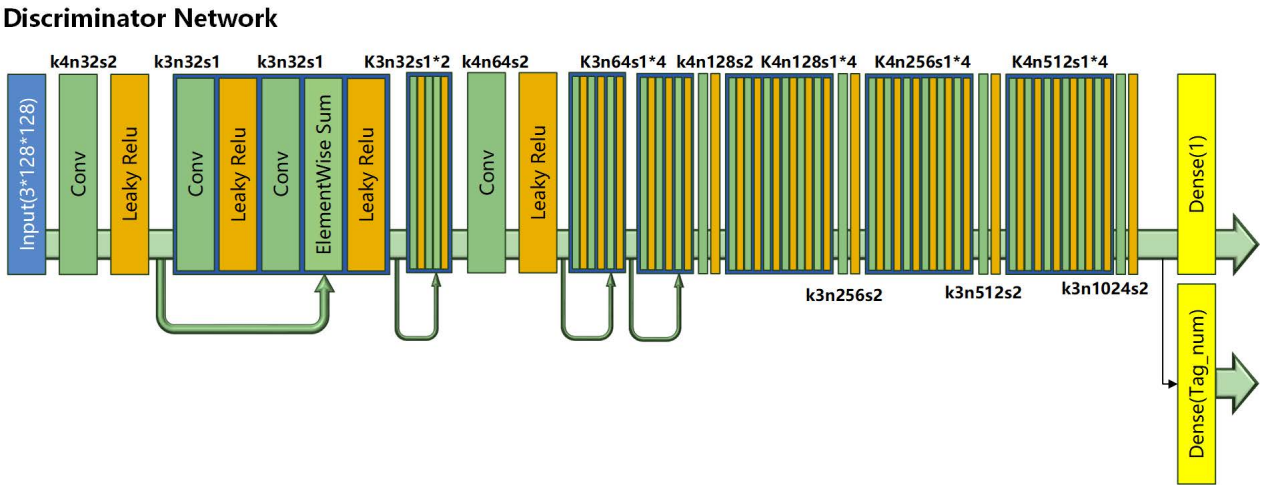

FIGURE 3. Discriminator Architecture.

FIGURE 4. Generated samples.

This model uses 43740 data for training. Both the discriminator and the generator are optimized by the momentum-based stochastic gradient descent optimization algorithm Adam [28]. For all pictures, all are scaled to 128*128 size. To prevent the loss function from jumping out of the minimum value due to the high learning rate at the end of the training, which leads to training failure, the dynamic learning rate is used to stabilize the training process. The learning rate drops to 0.1 times the learning rate of the previous round after 50,000 iterations.

E. GENERATING RESULTS

Figure (4) shows the image generated by the generator of this model. The 128-bit random noise and the 34-bit tag vector are input to the generator. The 128-bit random noise is randomly generated from a normal distribution with a standard deviation of 1 and an expectation of 0. The 34-bit tag vector is a randomly generated One-Hot code.

To evaluate the IS score of the model, we generated 10000 images through the generator. Generator input vector and label vector are random vectors. This is to ensure that the prior distribution of the sampled labels is the same as that of the training dataset labels. Then input all the images into InceptionV3. Each image outputs a 1000-dimensional classification vector. Then, the feature vectors were substituted into formula (20) to calculate the average IS of 10000 images. Table 3 shows the average value of IS between the proposed model and other models. It can be seen that compared with

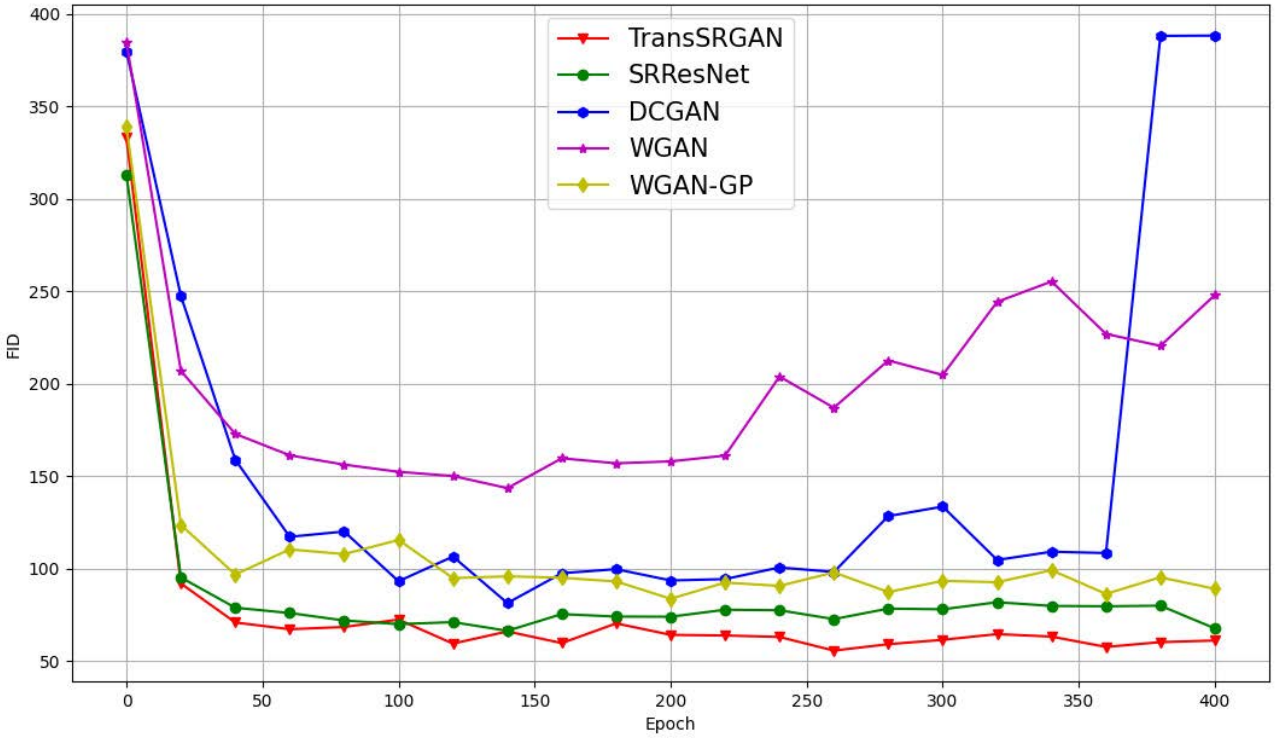

FIGURE 5. TransSRGAN and other models for FID.

TABLE 2. Hyperparameter.

Parameter	Explanation	Value
λ_1	Gradient Penalty Parameter	0.5
λ_2	ACGAN Multiplier	0.02
K	Gradient Penalty Parameter	1
β_1	Adam First-Order Momentum Decay Parameter	0.5
β_2	Adam Second Order Momentum Decay Parameter	0.999
lr	Initial Learning Rate	0.0002
C	Gradient Clip Parameter	10
e	Epoch	400
b	Batch Size	100

TABLE 3. FID and IS of TransSRGAN and other models.

Model	FID	IS
DCGAN	81.44±0.06	2.39±0.04
WGAN	143.58±0.24	2.33±0.03
WGAN-GP	83.79±0.03	2.34±0.03
SRResNet(Baseline)	66.46±0.12	2.41±0.05
TransSRGAN	**55.75±0.40**	**2.53±0.04**

the baseline model, the model in this study has higher IS and can generate clearer and more diverse images.

To evaluate the FID score of the model, 10,000 images are sampled from the real dataset. Then generate fake data with tags as same as the tags of real data. The generator input vector is a random vector. The tag vector is the same as the real dataset. It is to ensure that the sample tags have the same prior distribution as the training dataset tags. Then all images are input into InceptionV3. Each image outputs a 2048-dimensional feature vector. After that, the feature vectors are substituted into formula (21) to calculate FID. Five groups of images are generated, and five FIDs are calculated. Then calculate the average of five FIDs. Table (3) shows the average value of the FID of the TransSRGAN model and the other models. The TransSRGAN model has a lower FID than the baseline model. While generating images that are closer to real samples, the model also avoids mode collapse and ensures the diversity of generated samples.

Figure (5) and Figure (6) show the change of FID and IS of the model under different epoch times. It can be seen that after the 100th epoch, the FID and IS of each epoch of TransSRGAN training are lower than the baseline model, while the FID and IS of TransSRGAN tend to be stable after the 250th epoch of training. DCGAN suffered a mode collapse after 350 rounds. WGAN cannot learn sample distribution well due to weight clipping.

F. COMPUTATIONAL COST

Table (4) shows the video memory occupancy comparison between TransSRGAN and pure CNN model and pure Transformer model at 30 batchs. It can be seen that compared with full Transformer model, TransSRGAN can achieve better generation quality improvement only by increasing lower video memory consumption than full CNN model.

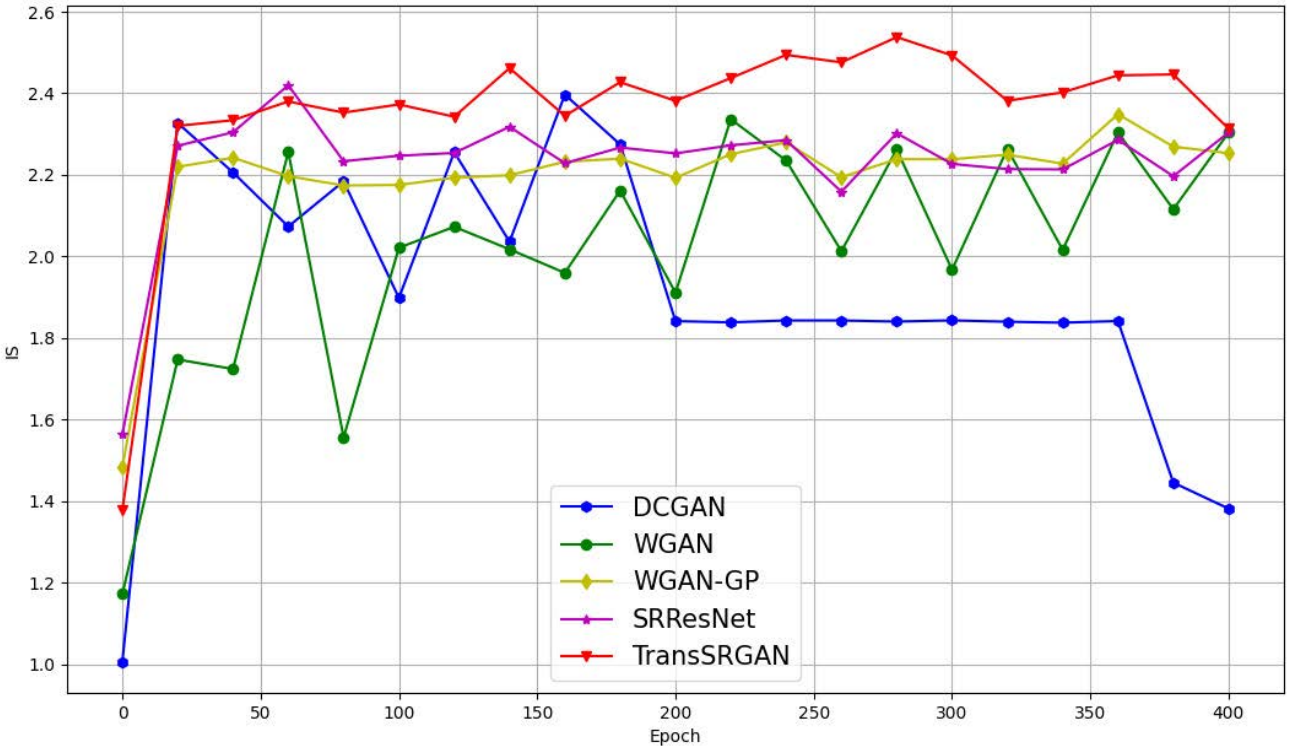

FIGURE 6. TransSRGAN and other models for IS.

TABLE 4. Computer Cost.

Model	Memory Used
Full-CNN	3496Mb
TransSRGAN	8916Mb
Full-Transformer	24862Mb

VI. CONCLUSION

In order to make the generation model produce more realistic images, TransSRGAN based on the GAN is proposed, which generates clear images of anime characters. The main improvement of this study is that Transformer Encoder is used as a sub-module to generate adversarial network generator. After a series of self-attention calculation and feature up-sampling, the generated samples are closer to the distribution of original samples. How to further reduce the memory usage of the model while making the image more realistic is still a problem worthy of further research.

REFERENCES

[1] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[2] H. Hu, M. Liao, W. Mao, W. Liu, C. Zhang, and Y. Jing, "Variational auto-encoder for text generation," in *Proc. IEEE 5th Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Jun. 2020, pp. 595–598.

[3] V. Aggarwal, M. Cotescu, N. Prateek, J. Lorenzo-Trueba, and R. Barra-Chicote, "Using vaes and normalizing flows for one-shot text-to-speech synthesis of expressive speech," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 6179–6183.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 139–144.

[5] S. Gencoglu and H. Y. Keles, "Sign language video synthesis using skeleton sequence," in *Proc. 28th Signal Process. Commun. Appl. Conf. (SIU)*, Oct. 2020, pp. 1–4.

[6] L. Zhao and Y. Zhang, "Generative adversarial networks for single image with high quality image," *KSII Trans. Internet Inf. Syst.*, vol. 15, no. 12, pp. 4326–4344, 2021.

[7] M.-K. Back, S.-W. Yoon, and K.-C. Lee, "GAN-based augmentation for populating speech dataset with high fidelity synthesized audio," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2020, pp. 1267–1269.

[8] Y. Li, J. Li, and Y. Wang, "Privacy-preserving spatiotemporal scenario generation of renewable energies: A federated deep generative learning approach," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2310–2320, Apr. 2022.

[9] Y. Li, B. Wang, Z. Yang, J. Li, and C. Chen, "Hierarchical stochastic scheduling of multi-community integrated energy systems in uncertain environments via Stackelberg game," *Appl. Energy*, vol. 308, Feb. 2022, Art. no. 118392.

[10] Y. Hong, U. Hwang, J. Yoo, and S. J. A. C. S. Yoon, "How generative adversarial networks and their variants work: An overview," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–43, 2019.

[11] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.

[12] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.

[13] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–17.

[14] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.

[15] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "MMD GAN: Towards deeper understanding of moment matching network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[16] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The Cramer distance as a solution to biased Wasserstein gradients," 2017, *arXiv:1705.10743*.
[17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
[18] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of GANs," 2017, *arXiv:1705.07215*.
[19] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–16.
[20] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.
[21] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the automatic anime characters creation with generative adversarial networks," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 1–16.
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
[24] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7794–7803.
[25] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1874–1883.
[26] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.
[27] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

HANGYU LIU was born in 1997. He is currently pursuing the Graduate degree with the School of Computer and Control Engineering, Yantai University, Yantai, Shandong, China. His research interests include big data, intelligent information processing, and data mining.

QICHENG LIU was born in 1970. He received the Ph.D. degree. He is currently a Professor with the School of Computer and Control Engineering, Yantai University, Yantai, Shandong, China. His research interests include big data, intelligent information processing, and data mining.