Proceedings of the 2023 International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/8/20230276

Handwritten digit image generation using improved generative neural network

Furui Xiao

School of Communications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, 210046, China

B20011920@njupt.edu.cn

Abstract. Generative adversarial network (GAN) is a new and effective neural network model. GANs are used in a lot of areas such as images and visual, speech and language, composing music, and creating 3D shapes. Although the GAN has a lot of applications, there are still some problems with it. Because not only the generator but also the discriminator needs to be trained, this model is easy to be unstable. And the GAN may not generate different images in human vision to make the generated images can safely pass the discriminator. Some methods will be shown to solve the problems of unstable and lack of diversity in this article by an example that generates handwritten digits. The methods are changing the optimizer from SGD to Adam or SGD with Adam, adding a Batchnorm, and combining GAN with CNN or RNN, each of the improvements will be contrasted with others by the chart of loss rate and the generated images. The research finds that each of the improvements has some advantages over the original GAN in the area of speed or stability. Especially, GAN with the optimizer Adam and DCGAN have the most stable consequence according to the image from the generator.

Keywords: deep learning, generative neural network, image generation.

1. Introduction

In 2014, Goodfellow et al propose a new generative model --Generative adversarial network (GAN) [1]. Nowadays, GANs have a lot of applications: creating similar photos, changing the style of a picture, creating a super-Resolution image, composing music, creating a 3D shape from 2D images, image inpainting, predicting a person's appearance in old age by aged-GAN [2-4]. The recently controversial artificial intelligence (AI) paintings also use GANs [5]. Its advantages and wide application should be attributed to its different way of thinking from other neural networks [6]. A Markov chain and various approximate reasoning have been abandoned in the training process of GAN. Moreover, to improve the training efficiency of it, no complex variational lower bound is used [7]. By directly generating new images, GAN escaped from direct replication of input images, and raise the diversity of results. However, the GAN especially the original version also has some problems: 1st. Mode collapse which means the the generator will create monotonous safely samples to deceit the discriminator. 2nd. non-convergence is a common problem in the training of machine learning, 3rd.diminished gradient. The discriminator trained too successful, as a result, the generator can never deceit the discriminator. 4th.unbalance between the generator and discriminator will lead to be overfitting And because both the generator and the discriminator need to be trained, this model is easy to be unstable when trained which

DOI: 10.54254/2755-2721/8/20230276

will lead to a completely useless result [8,9]. These problems present difficulties for the practical and further application of this model. Therefore, the author found some ways to solve the problems and improve the original GAN model. In this article, the consequence of some improvements in GAN will be compared to the basic GAN's result by an example that generates handwritten digits. The improvements include changing the optimizer from SGD to Adam or SGD with Adam, adding a Batchnorm, and combining GAN with other neural networks like CNN and RNN.

2. Method

2.1. Dataset

The MNIST (Modified National Institute of Standards and Technology database) database is a enormous handwritten digits database which is widely used in machine learning especially in the programs about images processing. This database is built by re-mixed NIST's original samples. In the essay, the author mainly leverages 60,000 training examples.

2.2. GAN

In the GAN, the generator can be considered as a team of counterfeiters, which is trying to manufacture artificial currency and deceive the police, while the discriminator is like the police, attempting to detect the artificial currency. In that model, both the counterfeiters and the police need to be trained to complete their work. At last the generator will be able to generate the consequence which can even cheat human and other neural network. As the loss function in the article by Ian J. Goodfellow:

$$\min_{G}\max_{D} V(D,G) = E_{x\sim P_{data}(x)}[\log D(x)] + E_{x\sim P_{z}(x)}[\log(1-D(G(z)))] \quad (1)$$

The V(D,G) means the difference between D and G, as a result, when training the discriminator, the training should let the loss function as big as possible. And when training the generator, the discriminator should be frozen and train the generator to let the loss function as small as possible.

Typically, this method used the SGD as an optimizer, the representation of SGD is:

$$\theta_i = \theta_i - \alpha\frac{\partial L}{\partial \theta_i} \quad (2)$$

2.3. Improvements on GAN

2.3.1. GAN with Adam optimizer. Stochastic gradient descent (SGD) is a classic and simple optimizer to do the gradient descent. However, Adam combines the advantages of AdaGrad and RMSProp optimization algorithms and it have a higher convergence speed. So, this improvement changed the SGD optimizer to Adam. And the representation of Adam is:

$$m_i = \beta_1 m_i + (1-\beta_1)\frac{\partial L}{\partial \theta_i} \quad (3)$$

$$v_i = \beta_2 v_i + (1-\beta_2)(\frac{\partial L}{\partial \theta_i})^2 \quad (4)$$

2.3.2. GAN with Adam and SGD optimizer. Although the Adam has some advantages because it has second order momentum, it may have these problems: it may not converge; it may miss the global optimal solution. Hence, one choice to reduce the impact of these problems is using SGD in the discriminator of GAN and using Adam in the generator of GAN.

2.3.3. GAN with Batchnorm. According to the 5 tricks mentioned in the article by Alec Radford [10]. 1) Replace the pooling layers with progressive addition in the discriminator and fractional addition in the generator. 2) Apply the layer of Batchnorm in both generator and discriminator. 3) Withdraw the fully connected hidden layers. 4) Apply ReLU as a activation in generator for all layers besides the output to replace the activation of Tanh. 5) Apply LeakyReLU as an activation in all of the layer in the discriminator.

DOI: 10.54254/2755-2721/8/20230276

This improvement will apply the second of these tricks: Apply Batchnorm in both generator and discriminator to make the GAN model stable.
Learning can be stabilized using BN, which can be used to solve the problems caused by poor initialization.

2.3.4. DCGAN. DCGAN is the deep convolutional generative adversarial networks, according to the article by Alec Radford, build the DCGAN with the tricks mentioned in picture1 above [10]. DCGAN actually is the combination of CNN and GAN, because the employment of CNN in discriminator, it will be easier for the discriminator to discriminate the difference between fake and real images which will lead a better consequence. And it may also be easier for researchers and learners to realize the logic of GAN model. Figure 1 shows CNN in DCGAN with the layers as follows:

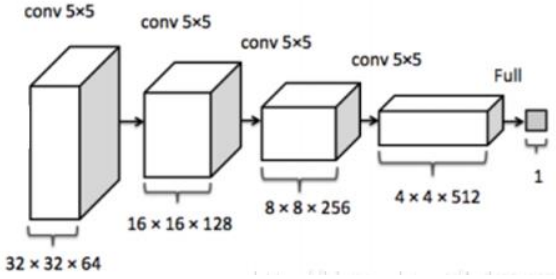

Figure 1. The network structure of CNN in DCGAN.

2.3.5. GAN+RNN. According to the DCGAN, the author wonders if the RNN also can do the same work as CNN in DCGAN. Typically, the author applies the simple RNN to do a simple discriminate. The simple RNN is a fully-connected RNN, which is trained in the circle of output and input.

3. Results

3.1. Result of conventional GAN

By training the basic GAN to generate handwritten digits, the generated images and a chart where the loss function's value changes with training times are as the Figure 2 and Figure 3 below.

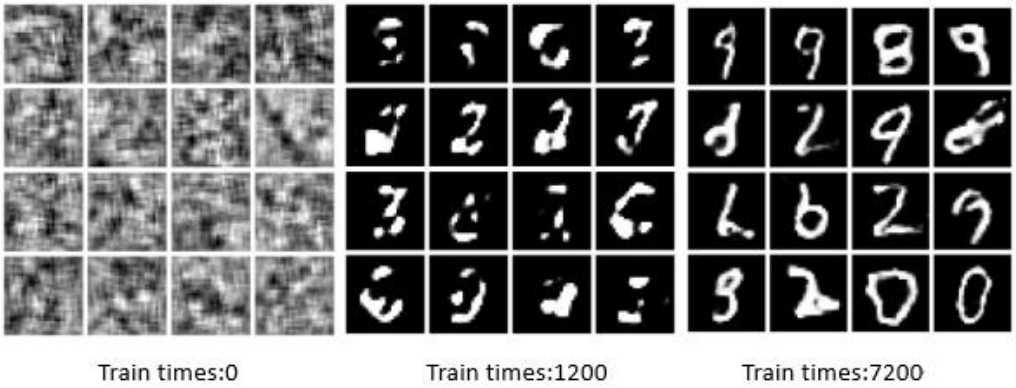

Figure 2. Result visualization of different epochs of conventional GAN.

DOI: 10.54254/2755-2721/8/20230276

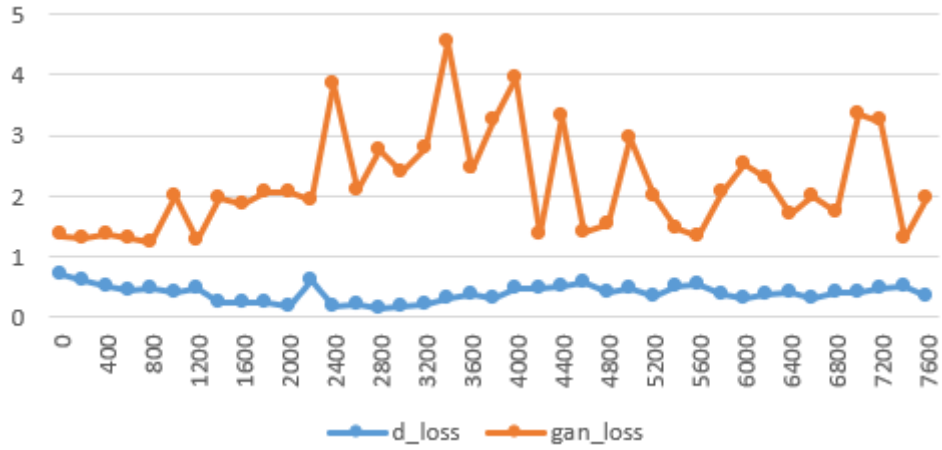

Figure 3. Loss curve of conventional GAN.

3.2. Result of GAN with Adam optimizer

By training the GAN with the optimizer Adam to generate handwritten digits, the generated images, and a chart where the loss function's value changes with training times are as the Figure 4 and Figure 5 below.

Figure 4. Result visualization of different epochs of GAN with Adam optimizer.

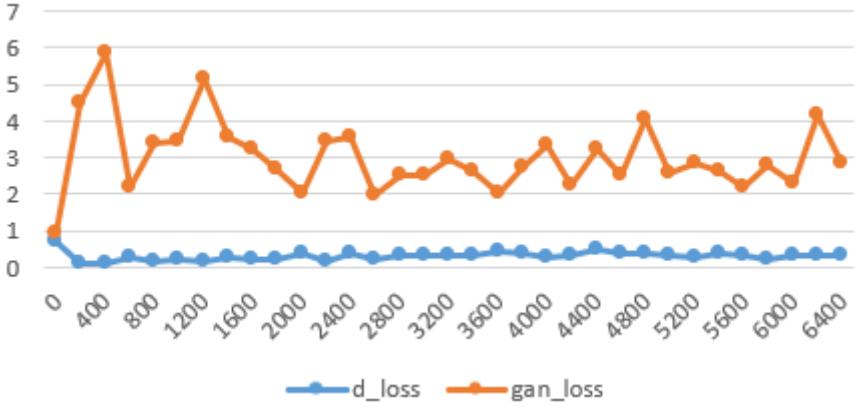

Figure 5. Loss curve of GAN with Adam optimizer.

DOI: 10.54254/2755-2721/8/20230276

3.3. Result of GAN with Adam and SGD optimizer

By training the GAN with the optimizer Adam and SGD to generate handwritten digits, the generated images, and a chart where the loss function's value changes with training times are as the Figure 6 and Figure 7 below.

Figure 6. Result visualization of different epochs of GAN with Adam and SGD optimizer.

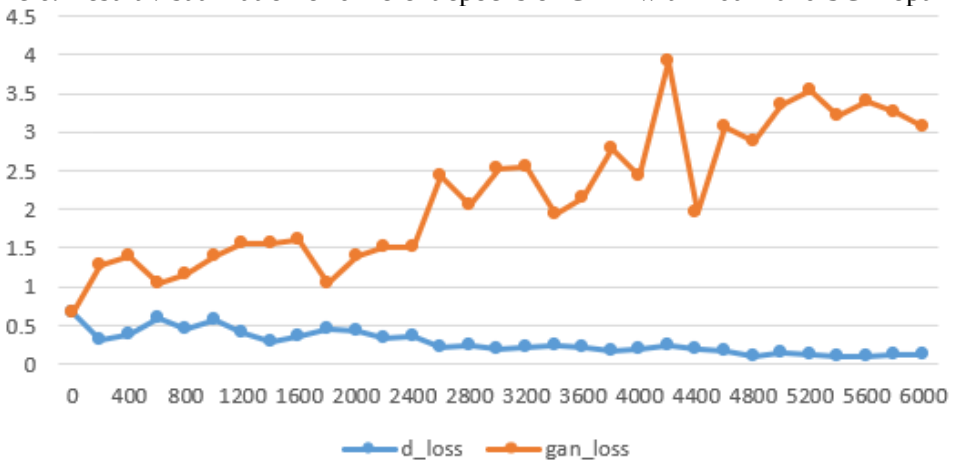

Figure 7. Loss curve of GAN with Adam SGD optimizer.

3.4. Result of GAN with Batchnorm

By training the GAN with the layer of batchnorm to generate handwritten digits, the generated images, and a chart where the loss function's value changes with training times are as the Figure 8 and Figure 9 below.

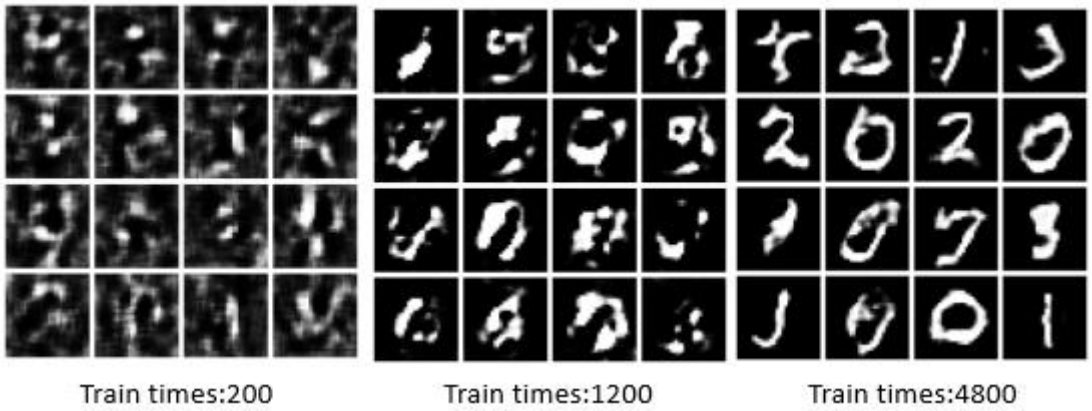

Figure 8. Result visualization of different epochs of GAN with Batchnorm.

DOI: 10.54254/2755-2721/8/20230276

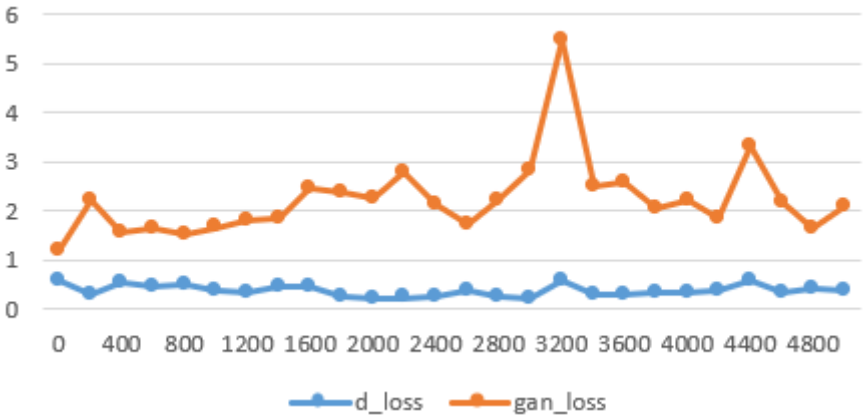

Figure 9. Loss curve of GAN with Batchnorm.

3.5. Result of DCGAN

By training the DCGAN to generate handwritten digits, the generated images, and a chart where the loss function's value changes with training times are as the Figure 10 and Figure 11 below.

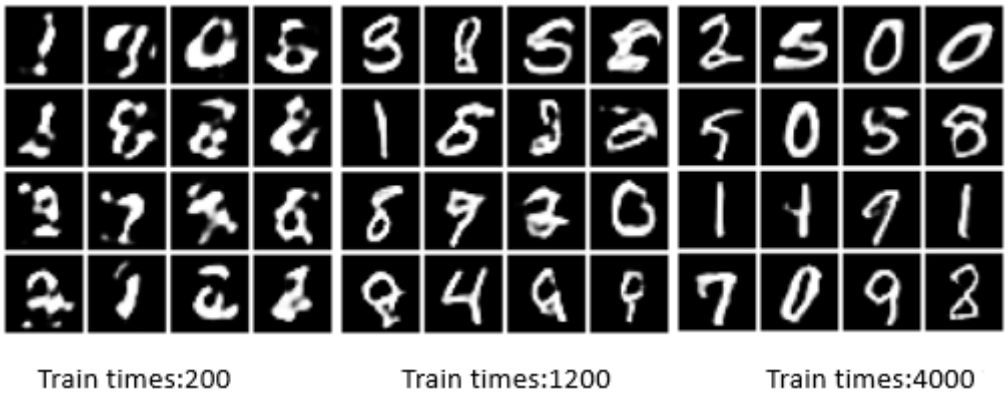

Figure 10. Result visualization of different epochs of DCGAN.

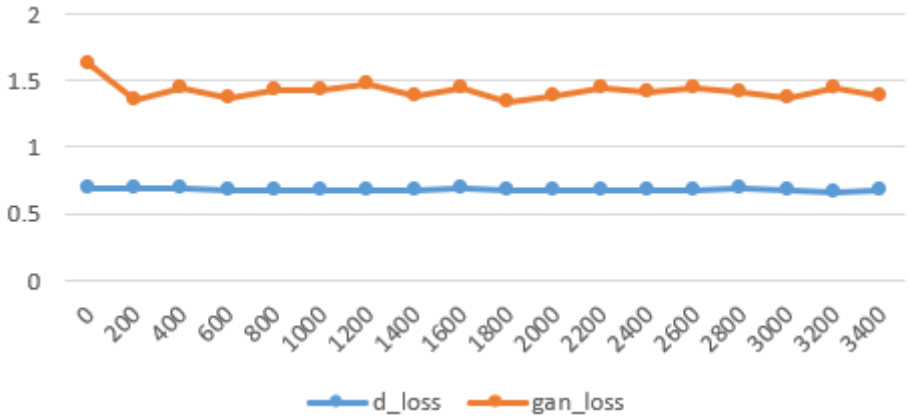

Figure 11. Loss curve of DCGAN.

3.6. Result of DCGAN with RNN

By training the model of GAN+RNN to generate handwritten digits, the generated images, and a chart where the loss function's value changes with training times are as the Figure 12 and Figure 13 below.

DOI: 10.54254/2755-2721/8/20230276

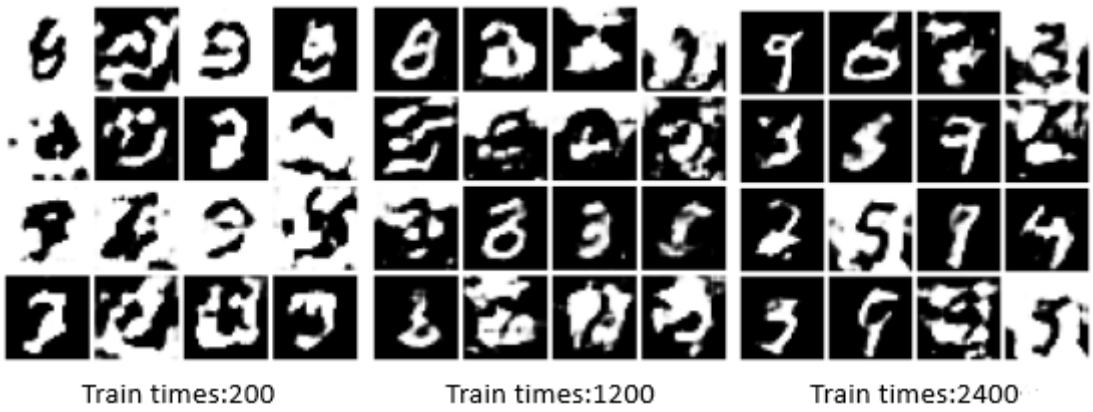

Figure 12. Result visualization of different epochs of DCGAN with RNN.

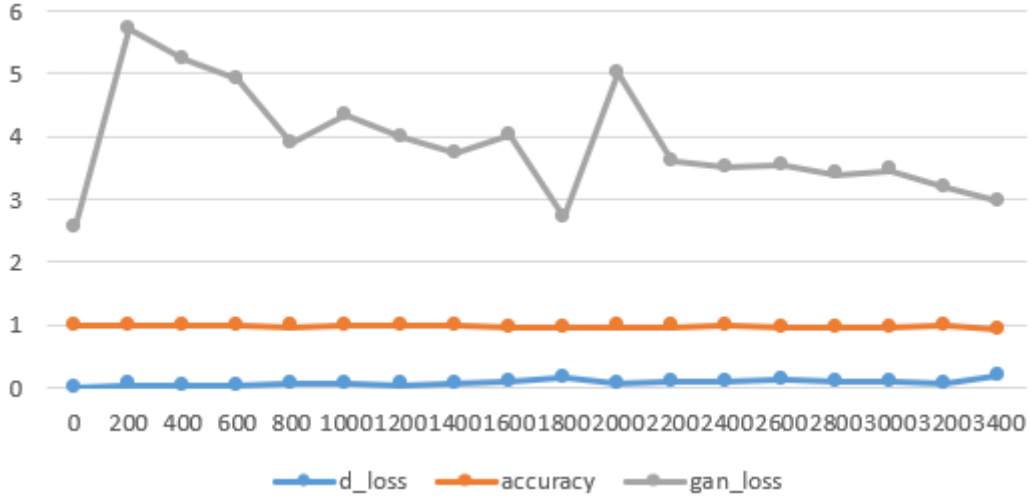

Figure 13. Loss curve of DCGAN with RNN.

4. Discussion

The basic GAN has an acceptable result after 7200 times' training, but too much training times waste a lot of time, as a result, there are some possible ways to get an even better result in less time. The chart of loss rate can't reflect whether the training situation is like the rate of loss in other models because the GAN trained both the discriminator and generator in a loop.

In contrast to the result of basic GAN, the GAN with the optimizer Adam can get a better result in just 5800 training times. Hence, the optimizer Adam can lead to a more stable result than the optimizer SGD in the basic GAN.

Although the basic GAN with the optimizer Adam and SGD can get the same result sooner than basic GAN. Its result is not better than the result of the GAN with the optimizer Adam, the chart of the loss of GAN also shows that the loss function is hard to converge in the model. It may be because the author just uses the default parameter in this model, while the optimizer SGD always needs to be set at the best learning rate. The basic GAN with Batchnorm doesn't have a very good result, it may because the trick of Batchnorm should be used with other tricks mentioned in the article by Alec Radford & Luke Metz [10].

The DCGAN has a better result than any other models above. The basic GAN+RNN has some special consequences and problems. In contrast to other consequences, the result of this improvement has some simple with black characters on a white background, if the parameter of the simple RNN will be changed this model may have a better result.

DOI: 10.54254/2755-2721/8/20230276

Compared with the basic GAN, all the improvements have some advantages in the training time or the realness of results. Some of the improvements can be more stable by changing some parameters. Although DCAGN has the most stable and natural consequence in this article, this improvement of GAN still has the problems of no creditable value to measure the quality of the trained model and no ability to generate designated results. The former problem can be solved by WGAN, while the latter can be solved by CGAN.

5. Conclusion

By the generated images and charts where the value of the loss function changes with the number of trainings this study finds that the improvements mentioned in the article: changing the optimizer from SGD to Adam or SGD with Adam, adding a Batchnorm and combining GAN with CNN or RNN all have some advantages over the original GAN in the area of speed or stable. Especially, GAN with the optimizer Adam and DCGAN have the most stable consequence according to the image from the generator. GAN is a new and effective neural network model, however, there are still some problems of it. These improvements in this study can effectively solve the problems of unstable and lack of diversity. As a result, GAN can be used more stably and widely. Furthermore, the improvements in the article all can be improved by changing some parameters. Undeniable, there are many other ways to improve GAN like WGAN which can be used in most cases to get a stable model. GAN has great application value in images and visual computing, speech and language processing, information security, and other fields. In the long term, the application of GAN to promote the development artificial intelligence, improve the ability of artificial intelligence to understand the world, and even make the creativity of artificial intelligence be possible is worth thinking about.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., et al. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144.

[2] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. IEEE signal processing magazine, 35(1), 53-65.

[3] Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X., & Wang, F. Y. (2017). Generative adversarial networks: introduction and outlook. IEEE/CAA Journal of Automatica Sinica, 4(4), 588-598.

[4] Aggarwal, A., Mittal, M., & Battineni, G. (2021). Generative adversarial network: An overview of theory and applications. International Journal of Information Management Data Insights, 1(1), 100004.

[5] Sun, L., Chen, P., Xiang, W., Chen, P., Gao, W. Y., & Zhang, K. J. (2019). SmartPaint: a co-creative drawing system based on generative adversarial networks. Frontiers of Information Technology & Electronic Engineering, 20(12), 1644-1656.

[6] Yongjie, M. A., Xiaodong, X. U., Ru, Z. H. A. N. G., Yirong, X. I. E., & Hong, C. H. E. N. (2021). Generative adversarial network and its research progress in image generation. Journal of Frontiers of Computer Science & Technology, 15(10), 1795.

[7] Turner, R., Hung, J., Frank, E., Saatchi, Y., & Yosinski, J. (2019). Metropolis-hastings generative adversarial networks. In International Conference on Machine Learning, 6345-6353.

[8] Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862.

[9] Gui, J., Sun, Z., Wen, Y., Tao, D., & Ye, J. (2021). A review on generative adversarial networks: Algorithms, theory, and applications. IEEE transactions on knowledge and data engineering, 1-28.

[10] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.