Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 5
AIDML 2024

Exploring the Effectiveness of Hyperparameters in Deep Convolution Generative Adversarial Networks

Zhiqiao Kang

School of Future Tech, Guangzhou International Campus, South China University of Technology, Guangzhou, Guangdong, 510006, China

202230051139@mail.scut.edu.cn

Abstract. Traditional machine learning models are often only able to classify or predict data, and cannot generate new simulated data, while Generative Adversarial Network (GAN) provides the possibility for machines to generate high-quality and diverse data. GAN can be used for image data generation, image-to-image transformation, translation of image information and text information into each other, and so on. There are many different types of GAN can realize different types of functions. GAN has a strong generating ability, can generate some high-quality simulation data for human reference. The framework idea of GAN is also very innovative and has been widely developed for applications in unsupervised or semi-supervised learning. In addition, GAN can be integrated with other machine learning models to form a more powerful model, so as to solve the problems that are difficult to deal with traditional models. Therefore, GAN is a very promising model in the field of artificial intelligence. In this thesis, the author completed the basic reproduction of Deep Convolution Generative Adversarial Network (DCGAN), and analyzed the quality of DCGAN generated images, using the control variable method. Some parameters are modified, such as learning rate, batch size, latent size, and explored the influence of these parameters on the training effect of DCGAN. The characteristics of basic parameters on DCGAN image generation results are summarized.

Keywords: Generative adversarial network; convolutional neural network; image generation.

1. Introduction

Generative Adversarial Network (GAN) is a generative deep learning model. It includes two parts: generator and discriminator. The generator produces simulated data and the discriminator judges whether the input data is real or generated. Training the generator makes the generated data closer and closer to the real data, while training the discriminator makes its judgment more accurate. It is trained by letting the two neural networks -- the generator and the discriminator -- compete against each other to extract useful information from unlabeled data. The relationship between the two is adversarial, hence the name GAN.

Deep Convolution Generative Adversarial Network (DCGAN) makes some improvements on ordinary generative adversarial networks by introducing a convolutional neural network structure that enhances the performance of the network model and obtains the better images. In contrast to traditional GANs, DCGAN usually has faster training speed, higher quality generated images, and more stable training process.

GAN is an unsupervised learning that studies the latent distribution of unlabeled data samples and simulates the genuine data sample distribution to generate realistic sample data. The adversarial learning idea of zero-sum game in the network structure has had a profound impact on the field of deep learning. Its objective function idea is different from other traditional models and does not require a strict data expression, which also avoids the problem of training crash when the data complexity is too high. Therefore, GAN has a broad research prospect.

GAN initially caused a sensation in image generation, there are a lot of related papers research, such as generating realistic faces, generating cartoon character characters, with the development of GAN, there are a lot of new techniques, such as style migration, the application cases of this technology are facial aging generation, the change of seasons in the landscape and so on. Gradually, GAN can

accomplish tasks such as picture-picture translation, image super-resolution, video frame prediction, and text-image interpreting. Currently, GAN has a broad variety of applications and has made significant and excellent contributions in the fields of image processing, computer vision, language and speech processing, and information security.

In this thesis, the author mainly modifies several hyperparameters and explores the influence of the size of these parameters on the model training effect, the degree of model training can be seen by the G_loss, D_loss plot and the generated picture.

2. Related Work

GAN was first proposed by Goodfellow et al. in 2014 [1], and in the same year, Mirza et al. presented Conditional Generative Adversarial Networks (CGAN) [2], which enables the generator to generate samples of a specific class based on a specific condition. In 2015, Radford et al. suggested a Deep Convolutional Generative Adversarial Networks DCGAN [3], which added the convolutional neural network structure to the initial network architecture, making the training process more robust and the training effect better. In November of the same year, the adversarial autoencoder (AAE) [4] appeared, combining the features of GAN based on autoencoder (AE), and a year later, the combination of variational autoencoder (VAE) and GAN appeared [5]. The integration of the original traditional generative model and the respective features of GAN enables the generation of more realistic samples. In 2017 Martin Arjovsky et al. put forward the Wasserstein generative adversarial networks (WGAN) [6], which innovatively proposed the use of Wasserstein distance as a yardstick of the difference between the generated distribution and the true distribution, solving the problems of unstable training and pattern collapse problem of the conventional GAN, and in the same year Jun-Yan Zhu et al. proposed cycleGAN [7], which put forward the cyclic consistency concept that can realize the image style transformation.

In 2018, Andrew Brock et al. proposed BigGAN [8], which uses a large-body model and distributed training to achieve high-quality, high-resolution image generation. Also in this year, Zhang, Han et al. came up with Self-attention generative adversarial networks (SAGAN) [9], which uses a self-attention mechanism to establish long-distance dependencies between different regions to further improve image quality. In 2019, Tero Karras et al. proposed styleGAN [10], which introduces techniques such as style coding and adaptive instance normalization to take into account the realism and diversity of images.

Since its proposal, GAN has continuously attracted extensive attention and research in academia and industrial solution. The application prospect of GAN will be more and more extensive, and will have far-reaching impact on many fields.

3. Method

3.1. Overall Structure

Fig. 1 demonstrate the overall structure of GAN. The latent space with the addition of noisy data is transmitted to the generator, which generates fake sample data and feeds it to the discriminator along with the real data to get the discriminative result. The two network models are optimized simultaneously. The image generated by the generator will be closer and closer to the real data, and the discriminator's discriminative effect will be gradually improved.

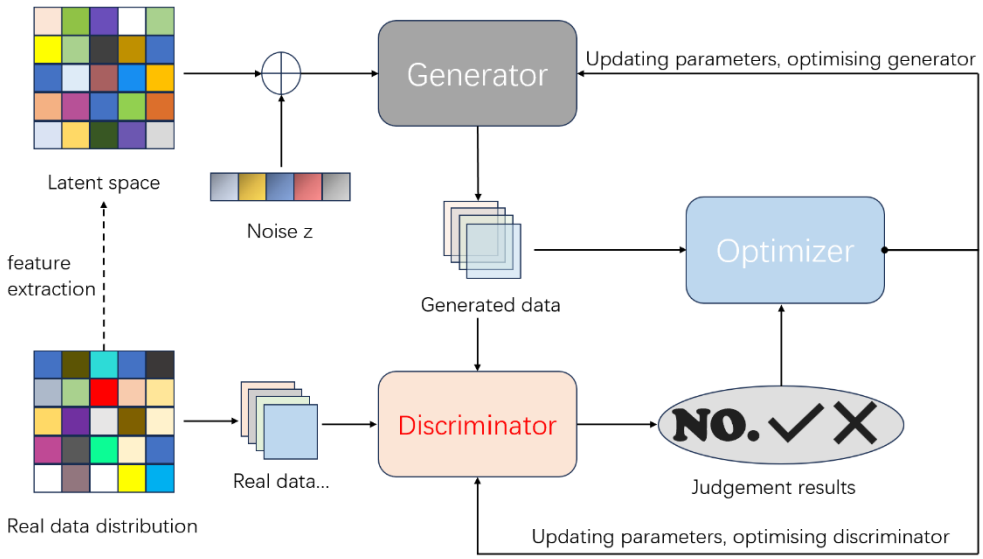

Fig. 1 Structure overview of GAN (Figure Credits: Original).

3.2. Mathematical Principles of GAN

According to that paper published by Goodfellow et al. in 2014, the objective function of GAN is the following formula:

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

G represents the generator and D represents the discriminator. x represents independently randomized genuine data, and $p_{data}(x)$ represents the probability distribution of the actual data. z stands for independent stochastic noise data (from the generator), and $p_z(z)$ represents the probability distribution of the data samples that the generator has learned. D(x) denotes the discriminator's prediction for a sample x from the real data distribution, that is, the probability that the discriminator considers a given sample to be a real data. So as to D(G(z)), which is the probability that the discriminator considers a given sample to be a generated data. That expectation term at the front of the formula represents the expectation of the discriminator's prediction of the true data, and it measures the degree to which the discriminator determines the true data to be true. The latter, represents the expectation of the discriminator's prediction of the false data generated by the generator, which measures the extent to which the discriminator determines the generated false data to be false.

This formula is very classic and expresses the core idea of adversarial learning in a simple, straightforward way. Both the generator and the discriminator are trained mutually. $\min_G$ means to minimize the difference between the generated sample data and the authentic data, so as to achieve the simulation effect of "fake to real". And $\max_D$ means to maximize the discriminator's capability to discriminate between authentic data and generated data.

3.2.1. Optimization of the Generator

It is expected that the data created by the generator to be like the genuine data very much, meaning the generated distribution should be similar to the distribution of the real data, i.e.,

$$\theta \longrightarrow \theta_{data} \Leftrightarrow P_G(x;\theta) \rightarrow P_{data}(x) \quad (2)$$

θ denotes the model parameters, θ_{data} denotes the target learning parameters, $P_G(x;\theta)$ represents the data sample distribution simulated by the generator, and $P_{data}(x)$ denotes the real data sample distribution. They are equivalent to P_G and P_{data} in the following section respectively. The maximum likelihood method can be used, which is equivalent to maximizing the following equation:

$$\theta^* = argmax_\theta L = argmax_\theta \prod_{i=1}^{k} P_G(x_i;\theta), x_i \in \{x_1, x_2, \ldots, x_k\} \quad (3)$$

After some statistically relevant derivations of the integral and combining it with the definition of the Kullback-Leibler Divergence, the following equation could be used:

$$\theta^* = arg\ max_\theta KL(P_{data}||P_G) \quad (4)$$

It can be observed that trying to make the distribution of the generated data approximate the distribution of the genuine data is equivalent to minimalise the KL divergence of the two distributions. This also makes sense when interpreted from the definition of KL divergence. Definition of KL divergence is the degree of difference between two distributions.

3.2.2. Optimization of the Discriminator

At the same time, it is also exptected that the discriminator to have the greatest discriminatory power and want it to be able to accurately discriminate between real and generated data. For a given loss function, this work uses the maximum likelihood method to find the best discriminator as follows:

$$\hat{D}(x) = \frac{P_{data}}{P_G + P_{data}} \quad (5)$$

Substituting this back into the loss function, making integral changes to the functional form, and combining it with the definition of the Jensen-Shannon Divergence, the following objective equation can eventually be derived:

$$L_D = -2log2 + 2JS(P_{data} \parallel P_G) \quad (6)$$

Ultimately, this expression tells us that trying to improve the discriminator ability, i.e., maximizing the loss function of the discriminator, is tantamount to achieving the goal of maximizing the JS divergence between the genuine data and the generated data. This is also consistent with the meaning of JS divergence: the degree of similarity between two distributions.

4. Expreiment and Result

4.1. Training Details

Experiments using the python version of 3.9, the implementation of the model network architecture is mainly dependent on pytorch, the author experiments are run with the GPU. About pytorch version information for torch==2.2.1+cudnn118, torchvision==0.17.1+cudnn118, in addition, the environment is also loaded with three packages, respectively, version 1.26.4 of numpy, version 3.8.3 of matplotlib and version 4.66.2 of the tqdm.

The image dataset used by the author is Anime Face Dataset NTU-MLDS [11]. Each picture's pixel is 64*64.

4.2. Experimental Settings

Details of the generator network is listed as the follows. Five convolutional layers with 512, 256, 128, 64 intermediate nodes, kernel size is 4, stride is 2, padding is 1. The first four of these convolutions are followed by a batch norm module and a leakyrelu module, and the last one uses Tanh to limit the output to between 0 and 1.

The discriminator network consists of five convolutional layers, the number of intermediate nodes is 64, 128, 256, 512, kernel size is 4, stride is 2, padding is 1. The first four of these convolutions are followed by batch norm and leakyrelu, and the last one uses sigmoid to normalize the outputs.

The learning rate is a parameter that controls the step size of each parameter update, β1 is used for first-order moment estimation of the gradient, and β2 is used for second-order moment estimation of the gradient. δ is the parameter used in gradient estimation to prevent the occurrence of a divide-by-zero error. Generator is optimized by Adam with learning rate = 0.0002, β1 = 0.5, β2 = 0.999, δ = 0.0005. Discriminator is optimized by Adam optimizer, with learning rate = 0.0001, β1 = 0.5, β2 = 0.999, δ = 0.0005. The training parameters includes epochs = 20, batchsize = 64, learning rate is 0.0002, latent_size is 128.

The controlled experiments is used to investigate the effect of each parameter as well as the network structure on the training results of the whole GAN. Based on the benchmark experimental group, variable modifications are conducted.

The learning rate includes 2*10e-2, 2*10e-3, 2*10e-4, 2*10e-5, 2*10e-6, where the modified learning_rate is equal to the learning_rate of opt_g, and the learning_rate of opt_d is always half that of opt_g. The latent size includes 32, 64, 128, 256, 512. The batch size includes 16, 32, 64, 128, 256. The number of intermediate nodes in the two major networks is uniformly changed to 16, 32, 64, 128, 256, 512

There are four major groups of experiments in total, the first three groups explore the effects of learning_rate, latent_size and batch_size on the training effect of the network model, and the fourth group of experiments is considered to be a small change to the original DCGAN structure (the number of intermediate nodes in the generator and the discriminator are both gradual and progressive layer by layer). There are five groups of experiments in each of the first three major groups, and six groups of experiments in the fourth major group, for a total of twenty-one groups of experiment

4.3. Quantitative Results

Results on different batch sizes are shown in Fig. 2. For the graph on the left, as can be seen, all of them are initially on the rise. If the batch_size is smaller, the G_loss for each round of training is smaller, however, the trend of G_loss for all five cases is still decreasing, and it may take more rounds of epochs to see whose G_loss is smaller in the end. For D_loss on the right side, they all go down in the beginning, but the smaller the batch_size is, the D_loss instead doesn't go down as the epochs grow. This shows that with enough training epochs, the larger the batch_size is, the better the discriminator's recognition effect is.

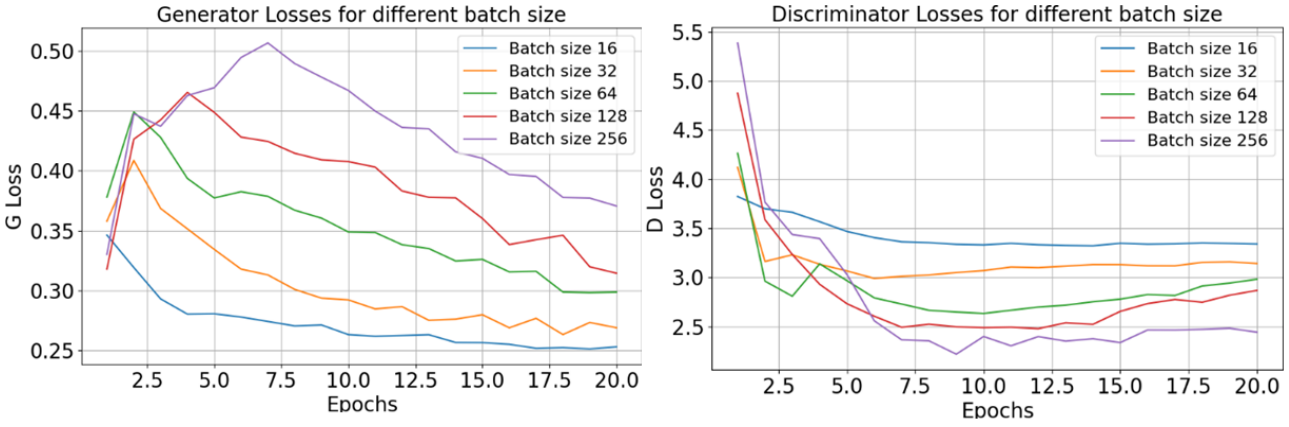

Fig. 2 Generator and Discriminator losses with different batch sizes (Figure Credits: Original).

Results of different latent sizes are shown in Fig. 3. Let's start by looking at the G_loss on the left-hand side. all of them go up and then down initially, except for the case where latent_size is 512. But in the end all cases pretty much converge together. For the D_loss on the right side, the smaller the latent_size is, the more obvious the tendency of the image to go down and then up is, and in the end it converges more or less the same way. When the data image size is small, the latent size shoulc be small.

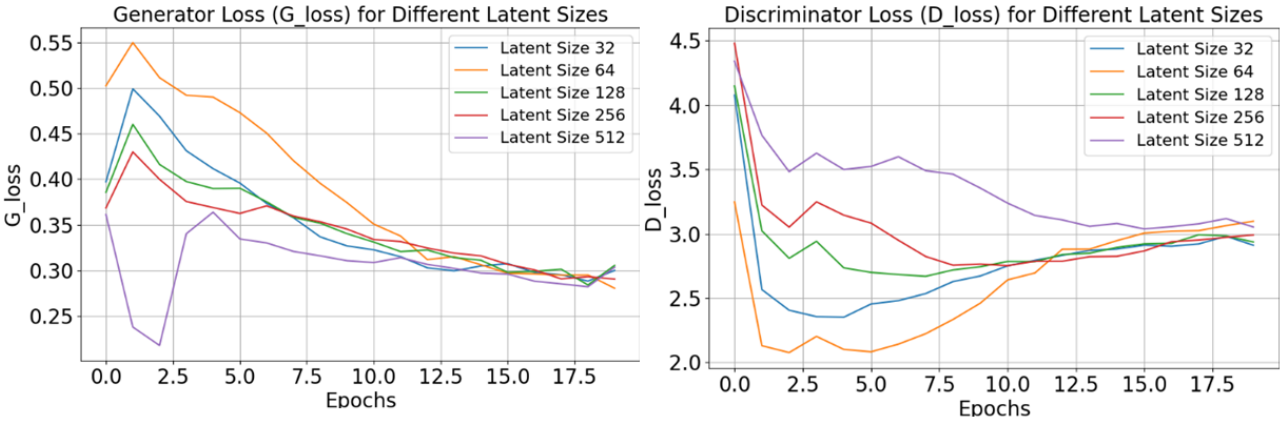

Fig. 3 Generator and Discriminator losses with different latent sizes (Figure Credits: Original).

Results of different learning rates are demonstrated in Fig. 4. Looking at G_loss, it could be observed that overall, the larger the learning_rate is, the more obvious the image tends to fall, as if the learning has spiked. However, if the learning_rate is too large, the image will be more ups and downs less stable, and the final convergence of the value is higher, the generated map is not exquisite. In the case where the learning_rate is equal to 2*10e-6, the image seems to converge well, but in reality, no features are learned, as can be seen in the generated image below. As for the D_loss on the right side, several lines seem to have no obvious pattern and feel very random. It's better to compare the generated image below. For the exploration of the variable learning_rate, it is important to judge it in conjunction with the effect of the generated image.

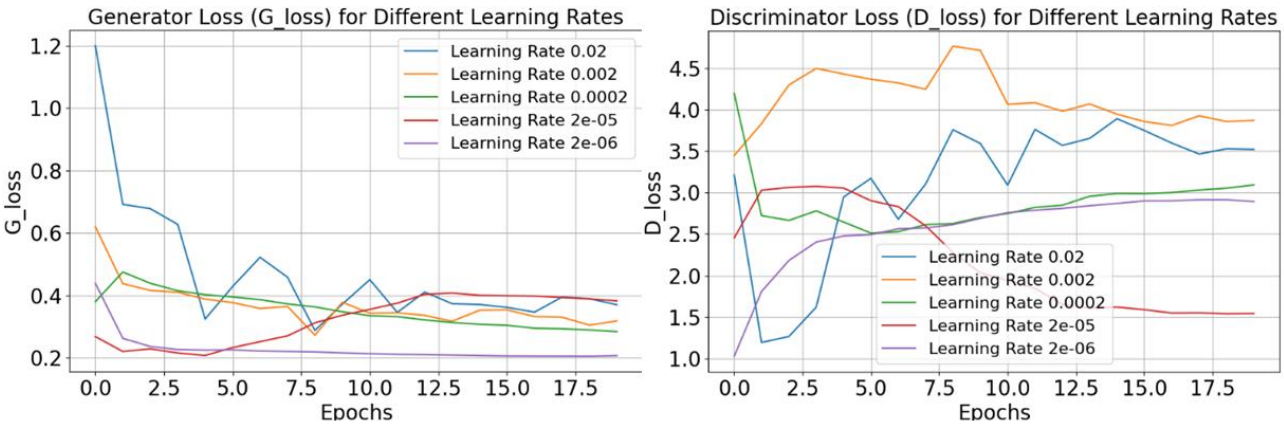

Fig. 4 Generator and Discriminator losses with different learning rates (Figure Credits: Original).

Results of different number of intermediate nodes per convolutional layer are displayed in Fig. 5. Let's look at the G_loss first, when nodes are larger, the G_loss is generally smaller and the generated graph quality is theoretically higher, while it seems that the G_loss can't converge when nodes <= 64, and the generated graph quality is theoretically lower. Looking at the D_loss, the situation is reversed, the D_loss is generally higher when the number of nodes is larger, initially decreasing a little and then gradually increasing. When nodes <= 64, D_loss first rises a little, then falls, and then increases slowly. All cases have an eventual tendency to slowly increment beyond the initial level. Ultimately it's important to look at the generated graph effects for a more detailed comparison.

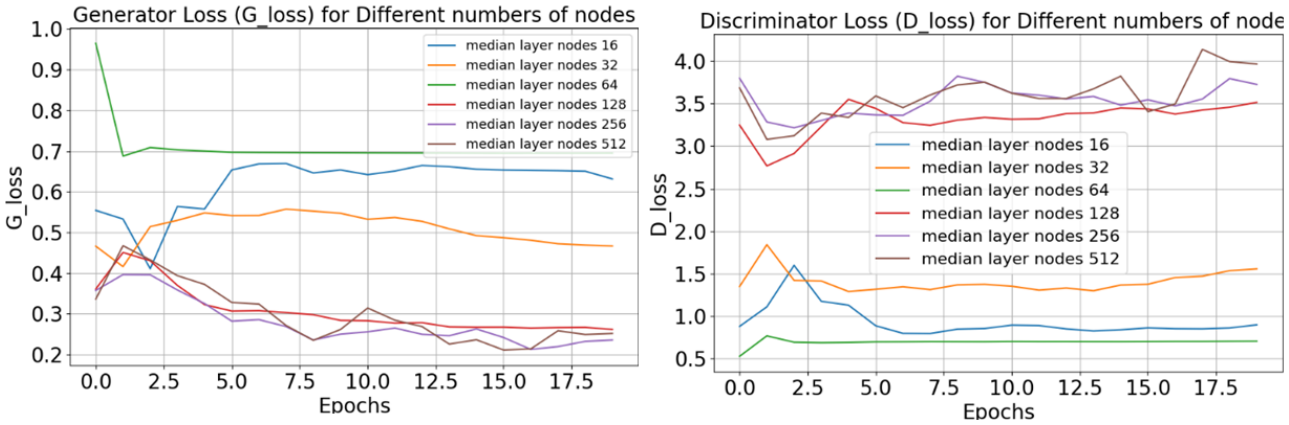

Fig. 5 Generator and Discriminator losses with different number of intermediate nodes per convolutional layer (Figure Credits: Original).

4.4. Visualization Results

The following results visualize the results from 1st, 3rd, 5th, 7th, 9th, 11th, 13th, 15th, 17th, and 19th epochs. Fig. 6 demonstrates the visualization results with differnt batch sizes. It looks as if groups with smaller batch_size have the best generated graphs and learn the most features per round of longitudinal comparisons. Overall, the production pictures work well for all groups except for the batch_size equal to 256 group, where the clarity of that group seems to be a bit unlearned.

Fig. 6 Visualization results. From top to dowm are with batch size = 16, 32, 64, 128 and 256 (Figure Credits: Original).

Fig. 7 presents the visualization results with differnt latent sizes. Results show that the group with smaller latent size learned slightly better, just from the generated graph alone. Once again, the latent_size doesn't need to be that much when the size of the image data used for training is small.

Fig. 7 Visualization results. From top to dowm are with latent sizes 32, 64, 128, 256, and 512 (Figure Credits: Original).

Fig. 8 shows the visualization results of different learning rates. Results illustrate well that the learning rate cannot be too large or too small. If it's too large, as the first line demonstrates, only vagarous results could be seen, but the quality of the image doesn't gradually improve as the epoch grows. Although the first epoch (the first image) learns quickly, the upper limit ends up being too low. Corresponds to G_loss. Look at the penultimate row, when the learning rate is too small, it is still a pile of mosaic after 20 rounds of epochs, and then look at the second to last row of the learning_rate is equal to 2*10e-5, the effect is a little better, the characters in the gradual development of the shape of the gradual clearer, but the learning efficiency is obviously low. The middle line of the generated map is the best, the face features are distinct, and the clarity is high.

Fig. 8 Visualization results. From top to dowm are with learning rates 2*10e-2, 2*10e-3, 2*10e-4, 2*10e-5, and 2*10e-6 (Figure Credits: Original).

Fig. 9 demonstrate the visualizations of different intermediate nodes. The overall generation of graphs for networks without a layer-by-layer gradient in the number of intermediate nodes was not as good

as in the benchmark experimental group. When the number of nodes is low, the generated graphs are blurry and the facial features are not obvious, and progress is slower as the number of training epochs increases. When the number of nodes is high, the generated graph becomes better, but is prone to crash generation and colour disorder. (The sixth row is more obvious.) From the figure, the fourth row(nodes = 128) has the best effect, and there is almost no crash problem while keeping the quality of the generated graph high. However, it is still not as effective as the benchmark group.

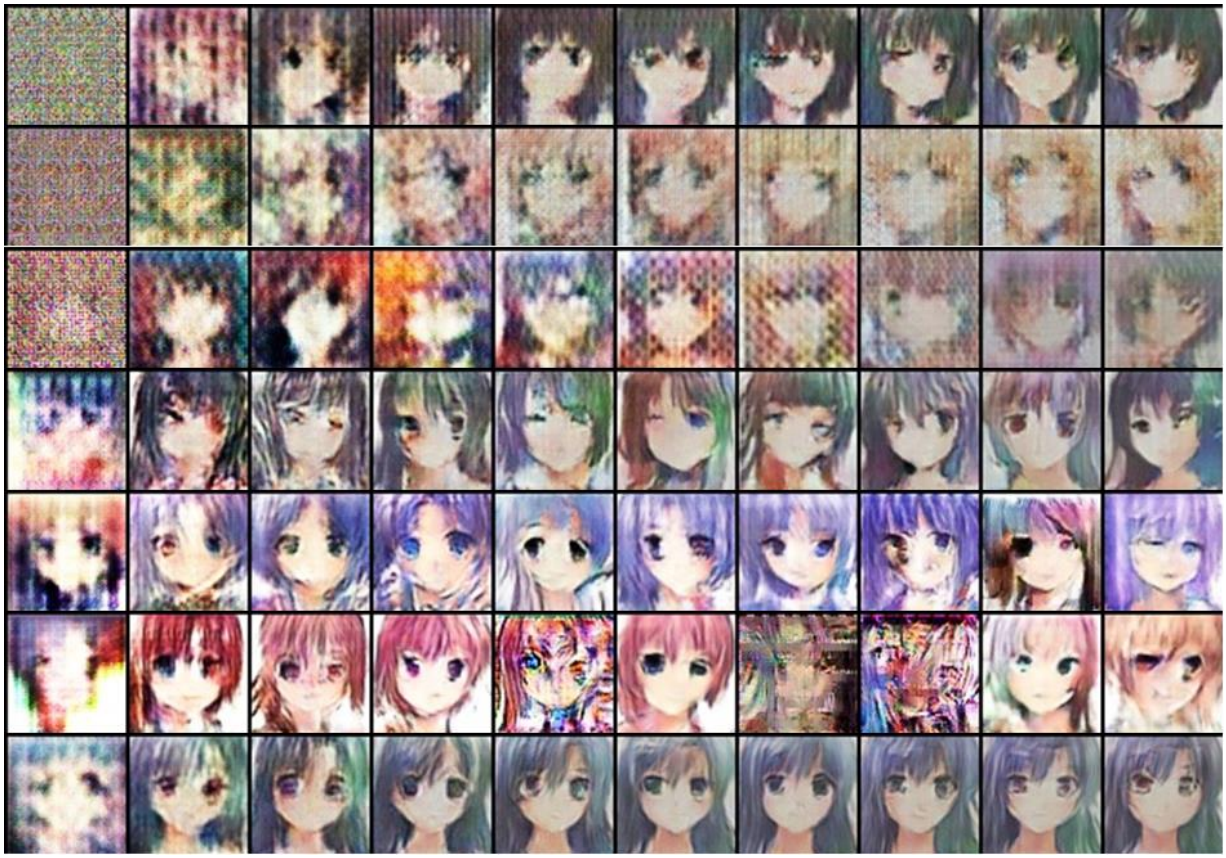

Fig. 9 Visualization results. The first six rows from top to down are with the number of intermediate nodes 16, 32, 64, 128, 256, and 512. The last row is the results for the benchmark experimental group (Figure Credits: Original).

5. Discussion

From results of different batch sizes show that the smaller the batch_size is, the more is learned in a epoch training. When comparing the images generated in each round of training under different batch_sizes vertically, the experimental group with a batch_size of 16 has the clearest images, and the features of the anime faces are the most obvious. When the batch_size is small, better generation results can be obtained in fewer training rounds. If the batch size is too large and good quality images are expected, the time required will be longer.

As for the latent size, to achieve a better result in fewer training rounds, latent_size cannot be too small or too large. When latent_size is gradually increased from 0, at first, more and more things will be learned in each round of training, and when latent size is large to a certain extent, the things learned in each round of training will slowly decrease. As the fifth row (latent_size = 512) in Fig. 5 shows the effect, the character features in the fifth row are not as good as the above rows. The maximum value of this latent_size is related to the properties of the image itself, and the image data used in this experiment are all 64*64 color images. Combined with the generated graph and the loss graph, it can be inferred that this extreme point has a high probability to be located in the interval (256,512).

As for the learning rate, there is a similarity between learning_rate and latent_size, in order to achieve better training results faster, the learning_rate cannot be too small or too large. It is obvious from the

generated graph that when the learning_rate is too small, few features are learned in each round of training, and the learning speed is slow. When the learning_rate is too large, the understanding of the features will not be very good, and the generated graph does not get progressively better as the number of rounds increases. The parameters do not converge around a correct value. The best learning_rate in this experiment should be in the neighborhood of 2*10e-4.

The quality of the generated graphs was generally poor. The group that worked best was the group with nodes=128, which guaranteed a high quality picture with few problems. Still not as good as the benchmark group

6. Conclusion

In this thesis, some exploration are conducted on some parameters of DCGAN, using the method of control variables to study the general influence of some basic parameters on the training effect of the model when the size of the given data image is small (64*64) and the clarity is low.

In this experiment, these results could be concluded: Learning rate is too small, will not get a better quality generated image in a relatively short time, while too large will never learn a higher quality production image, G_loss convergence can not be achieved. Latent size is too large or too small each round of learning is not much, and it needs more training rounds to generate higher quality graphs. For batch size, if it is too big, the training effect is not as good as a smaller one, which is contrary to the usual GAN model and may be caused by the low quality of the image dataset itself.

The fourth major set of experiments in this thesis also explores a rarer distribution of the amount of intermediate nodes in the construction of a DCGAN structure and compares it to the usual DCGAN (where the number of intermediate nodes in the convolutional layers is increasing or decreasing layer by layer) for generating graphs. The design of the number of intermediate nodes in DCGAN that increases or decreases layer by layer has its own beauty. This original design can save the arithmetic consumption during training, relatively simplify the model, and also ensure a higher quality of the generated graph effect.

There are many extensions that can be made on this experiment:

1. Try experimenting with wider intervals and denser gradients of parameter tuning, which allows for a more pronounced study of the asymptotic trends of parameter tuning on the model's effects.

2. Try more sets of repetitions to prevent chance and increase the accuracy of the experiment.

3. Try to modify the network structure of transformer, G and D, or the parameters therein for further study.

4. Try selecting other optimizers.

5. Try to repeat the experiment with another dataset to explore and compare the effect of DCGAN on image data of different sizes, qualities, and materials.

6. Try combining DCGAN with other traditional machine learning models vs. DCGAN alone and traditional models alone.

References

[1] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems, 2014, 27: 1-9.

[2] Mirza, Mehdi, and Simon Osindero. Conditional generative adversarial nets. ArXiv Preprint, 2014:1411.1784.

[3] Radford, Alec, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. ArXiv Preprint, 2015:1511.06434.

[4] Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. ArXiv Preprint, 2015: 1511.05644.

[5] Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In International conference on machine learning, 2017: 2391-2400.

[6] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In International conference on machine learning, 2017: 214-223.

[7] Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision, 2017: 2223-2232.

[8] Brock, Andrew, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. ArXiv Preprint, 2018: 1809.11096.

[9] Zhang, Han, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In International conference on machine learning, 2019: 7354-7363.

[10] Karras, Tero, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019: 4401-4410.

[11] Anime Face Dataset NTU-MLDS, Kaggle, URL: https://www.kaggle.com/datasets/lunarwhite/anime-face-dataset-ntumlds. Last Accessed 24/03/09