The portraits generation by using DCGAN, GRAGAN and StyleGAN2-ada

Hongyao Song [*1]

Abstract

Machine learning simulates human learning behavior through computer calculations and programs: training models from practical experience and applying them to similar scenarios. The emergence of Generative adversarial networks (GANs) has built a more solid bridge between artistic creation and machine learning. According to Deep Regret Analytic Generative Adversarial Networks (DRAGAN), this project improved it by building a model with Conv-BN-Relu structure instead only the 3 layers and implemented the DCGAN. I used a dataset of human portraits to train a Deep Convolutional GAN (DCGAN) model and StyleGAN2-ada (adaptive discriminator augmentation) created by Nvidia and the improved DRAGAN model, then made them randomly generate portraits. Meanwhile, I've tried to add more layers into the generator and discriminator networks, but the outputs were not ideal. From the structure and generation results of Deep Regret Analytic GAN, Deep Convolutional GAN and StyleGAN2-ada models, the differences between the two are analyzed.

1. Introduction

The development of machine learning technology is becoming more and more mature, and it has a wide range of applications, such as search engines, medical diagnosis, detection of credit card fraud, stock market analysis, DNA sequencing sequences, voice and handwriting recognition, strategy games and the use of robots (Xue & Zhu, 2009). At the same time, people started thinking about applying technologies of machine learning to "upgrade" art works, such as the restoration of artifacts and noises removals, even the creation of paintings (fou, 2019).

A French collective, Obvious, has created a painting, a portrait of a stocky gentleman in 18th-century dress, which has been sold for 432,500 dollars in December 2018 (BrainStation, 2019). They used Generative adversarial networks (GANs) to "paint" art works. The creation of Generative adversarial networks and the work of Obvious have proved that it's possible for people to produce artworks, or even new styles of art by using machine learning. Generative adversarial networks train a discriminator and a generator, the generator network can use hidden vector z to produce images after capturing the features of training datasets, while the discriminator can estimate an image comes from the generator or not (Goodfellow et al., 2014).

To improve the qualities of generated images, the Deep Convolutional Generative Adversarial Networks (DCGAN) have been proposed by Alec Radford in 2016, which introduced the convolutional network into the structure of GAN for the first time and used the powerful feature extraction ability of the convolutional layer to improve the effect of GAN [5]. In this project, I used 5 convolutional layers generator and discriminator to train the portrait dataset. However, some kinds of problems may exist during the training time, including mode collapse (poor diversity) and hyperparameter sensitivity (Arjovsky et al., 2017). In this project, I tried to build 5, 6, 7 and 8 layers to build my networks.

In order to reduce the impact of the problems that have been mentioned above, Naveen Kodali proposed Deep Regret Analytic Generative Adversarial Networks (DRAGAN)(Kodali et al., 2017). The author mentioned that the model collapse is due to the local balance in the non-convex case. He observed that the local balance always showed a sharp gradient around the real data in the discriminant function. Therefore, Kodali proposed DRAGAN and introduced gradient penalty mechanism to avoid local equilibrium. The results show that DRAGAN trains faster and is more stable than GANs with less model collapse. In this paper, I improved the original model structure by using 5 convolutional layers to build my networks instead of just using one linear layer.

I also used StyleGAN2-ada (adaptive discriminator augmentation) model to train my portrait dataset, which has 4400+ images. Compared with StyleGAN and StyleGAN2, StyleGAN-ada doesn't need a powerful dataset, while the training process of high-definition face used 140 thousand images(Karras et al., 2020). In order to avoid the overfitting problem due to the small amount of data images, data augmentation can be performed on the data during training, such as random cropping, horizontal flipping and adding noise,

etc. However, data augmentation will lead to corresponding enhancement of the generated image. For example, adding noise to the original image will cause the generated image to have noise, which we do not expect to see. StyleGAN-ada can solve this problem by improving bCR (balanced consistency regularization) and using adaptive discriminator augmentation. So I choose StyleGAN-ada to train my dataset in this project.

The remaining part of this paper is organized as follows: Section 2 briefly discusses related work. Section 3 describes the preprocessing of datasets, the structures of proposed system and their implementation. Section 4 shows the randomly results of all the models. Finally in section 5, I discuss about some changes of parameters I made during training time and some thoughts about that.

2. Related work

2.1. Deep Convolutional Generative Adversarial Networks

As I mentioned before, DCGAN mainly improves the original GAN in the network architecture. Both the generator and the discriminator of DCGAN use the CNN structure to replace the fully connected network of the original GAN (Radford et al., 2016).

Both the generator and discriminator of DCGAN abandon the pooling layer (pooling layer) of CNN, the discriminator retains the overall architecture of CNN, and the generator replaces the convolutional layer with a deconvolutional layer (ConvTranspose2d). And the BN (Batch Normalization) layer is used in the discriminator and generator to accelerate model training and improve the stability of training. The generator network uses ReLU as the activation function, while LeakyReLU is used as activation function in the discriminator network.

2.2. Deep Regret Analytic Generative Adversarial Networks

Compared with GAN, the author added gradient penalty into the discriminator of DRAGAN to calculate the whole loss to avoid that multiple z vectors map to a single output x.

For some empirical optimization considerations, the equation of gradient penalty is showing as follow:

$$\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0,cI)} \left[\| \bigtriangledown_x D_\theta \left(x + \delta \right) \|^2 \right] \qquad (1)$$

2.3. StyleGAN and StyleGAN2

The main change that network structure of StyleGAN and StyleGAN2 have consist of two parts.

The first is the Mapping network, which can encode the input vector into an hidden vector z, just like what GAN does, and the hidden vector z will be subsequently passed to the generation network to obtain 18 control vectors, so that different elements of the control vector w can control different visual features.

The second part is the Synthesis network, which is used to generate images. The innovation is that A and B are fed to each layer of sub-networks. A is an affine transformation converted from w, which is used to control the style of the generated image. B is the transformed random noise, which is used to enrich the details of the generated image, for example, each convolutional layer can adjust the "style" according to the input A, and adjust the details through B.

3. DATA PREPROCESSING AND NETWORKS DESIGN

3.1. Data preprocessing

I've got 4427 images to train my three kinds of models, the original datasets are from Wiki-Art: Visual Art Encyclopedia on Kaggle and Mendeley Data.

The original images of these two datasets are mostly half-length portraits or full body portraits. So first, I imported face recognition on Pytorch to capture the face part of all the images and restored them as jpg with RGB. Second, I resized these portraits to 512 x 512 pixels. In this process, to make sure that the images are all squares, I calculated the filled the image when there's difference between its length and width.

For DCGAN and DRAGAN: During the training part of DCGAN and DRAGAN, I could use transforms. Resize to make sure the images are right size that I wanted. I trained 4 different sizes of portraits for 4 times, they are 64x 64, 128 x 128, 256 x 256 and 512 x 512, I also designed 4 different DCGAN networks of different number of layers of them to train the models.

For StyleGAN-ada: In this part, the StyleGAN-ada requires tfrecords format images with high qualities to train its model. So I converted 512 x 512 jpg dataset to tfrecords that I could use ffhq512 as preprocessing model for first time.

3.2. Model Design

Three models I've used in this project are all GANs, so their main working structures are same, which include 2 separate networks: generator and discriminator. The training process is shown as follow:

DCGAN:I used ConvTranpose-BN-Relu as the generator structure and Conv-BN-LeakyRelu as the discriminator

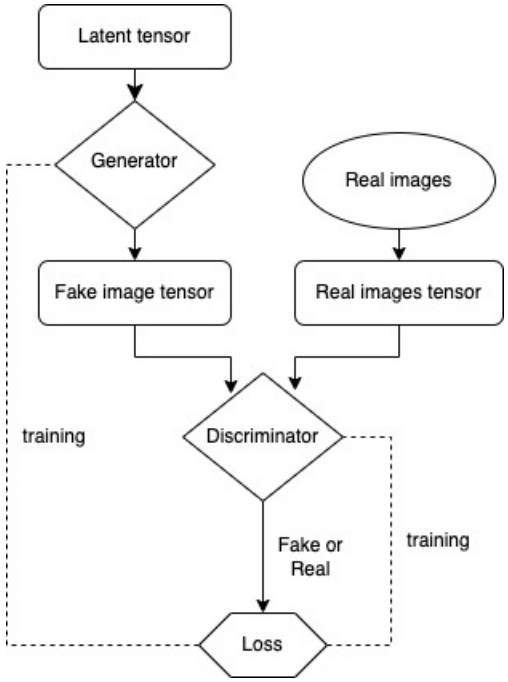

Figure 1. The main process of training GAN

structure. The specific figures of generator and discriminator I've designed are shown in Figure 2 and 3.

In these 2 figures, filter is the number of convolutional kernels, kernel represents the size of the convolutional kernels, stride is the size of moving convolutional kernels, padding is filling methods (Yi et al., 2021).

With 5 layers of ConvTranpose-BN-Relu, the generator can capture rich features from the sample images and generate hidden vectors which have same shapes as the samples have (Ioffe & Szegedy, 2015), so that the output can be sent to discriminator.

As for the discriminator structure, it used LeakyRelu, which ensures that output will still exist even the input is less than 0. Meanwhile, the output of this network is a single value, which represents the probability that the discriminator estimated if the input "image" is a real one or not.

I choose Adam optimizer, and β_1 = 0.5 and β_2 = 0.99.

During the training time, the hyperparameters of DCGAN I set are shown in table 1.

To train this whole network, I had to calculate the loss of generator and discriminator separately. The loss function here I used is binary cross entropy. The loss of discriminator in DCGAN consists of 2 parts. First, with the fake images tensor has been generated from the generator, I calculated the cross-entropy loss between this tensor and a zero-labels

Table 1. Hyperparameter configuration.

PARAMETERS	VALUE
DISCRIMINATOR LEARNING RATE	2E-4
GENERATOR LEARNING RATE	1E-4
BATCH SIZE	64
IMAGE SIZE	64
EPOCHS	500
FEATURES	64
HIDDEN VECTOR LENGTH	128
CHANNELS	3

tensor, which has same shape as the tensor generator has produced. Second, I needed to calculate the cross-entropy loss between the real images' tensor with an one-labels tensor, both has same size and shape. These calculations help discriminator to be trained and improved its ability of discrimination[13]. Then added the loss results I've mentioned above and divided by 2. The calculation formula is shown as equation (2):

$$D_{loss} = E_{\widetilde{x}\sim P_g}\left[CEloss(D(\widetilde{x}),0)\right] + E_{x\sim P_r}\left[CEloss(D(x),1)\right] \tag{2}$$

To calculate the generator loss, firstly I used generator to produce a fake images' tensor and make it as input of discriminator to make probabilities tensor, and then calculated the cross entropy of it and an one-labels tensor, which tried to confused the discriminator and made generator learn from it to generate images that look close to real images.

The calculation formula is shown in equation (3):

$$G_{loss} = E_{\widetilde{x}\sim P_g}\left[CEloss(D(\widetilde{x}),1)\right] \tag{3}$$

DRAGAN: For DRAGAN, I didn't use the structure that the original article proposed, which only has three layers. I used the structures of both generator and discriminator that I choose for DCGAN to build the DRAGAN networks, so that the features of input images can be captured more perfectly.

The only difference I've made for DRAGAN is its key part, the gradient penalty. The gradient penalty is used to be added into the calculation of discriminator's loss [11].According to the gradient penalty I mentioned before, the loss formula of DRAGAN's discriminator is shown as equation (4):

$$\mathrm{D}_{loss} = \left(E_{\widetilde{x}\sim P_g}\left[CEloss(D(\widetilde{x}),0)\right] + E_{x\sim P_r}\left[CEloss(D(x),1)\right]\right)/2 + \lambda \cdot \mathbb{E}_{x\sim P_{real},\delta\sim N_d(0,cI)}\left[\left(\|\bigtriangledown_x D_\theta\left(x+\delta\right)\| - k\right)^2\right] \tag{4}$$

In addition, the λ = 10 in this project.

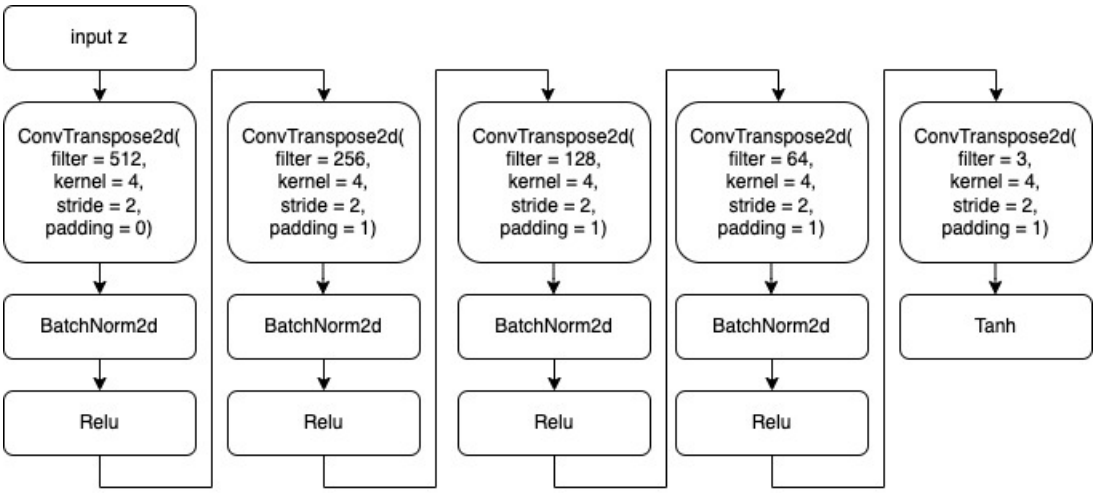

Figure 2. The layers structure of generator

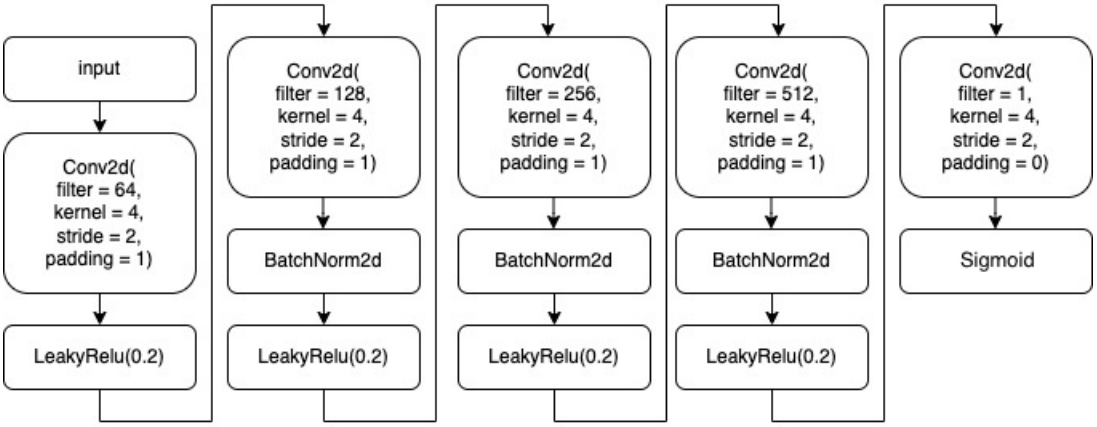

Figure 3. The layers structure of generator

StyleGAN2-ada: Based on StyleGAN2, the main structure of StyleGAN2-ada (adaptive discriminator augmentation) is similar with StyleGAN2's.

However, in other to prevent dealing with small quantity dataset from overfitting problem, some methods of data augmentation should be used during the training time. Then the output of networks may have some noises due to the data augmentation.

StyleGAN2-ada used BCR (balanced consistency regularization)(Zhao et al., 2020) to add the data augmentation and improved it. The basic idea of the BCR method is that if two different data augmentations are performed on the same input image, the output images obtained by the two enhancements should be the same. BCR also adds consistency regularization terms to the discriminator loss.

But for StyleGAN-ada, the authors of it decided to perform data augmentation on all the images, included the input of discriminator and generator, and the CR loss item of discriminator has been removed. The designed figure is shown in figure (4).

In this figure, the p represents the probabilities that decides whether to do data augmentation to the images or not. This p is generally set hyperparameter by authors, which's value has a great influence on the generated results.

So the adaptive discriminator augmentation of StyleGAN2-ada is to define and fix the value of p by proposing two overfitting heuristics, r_v andr_t[8].

The formulas of calculation process of these two parameters are shown in equation (5) and (6):

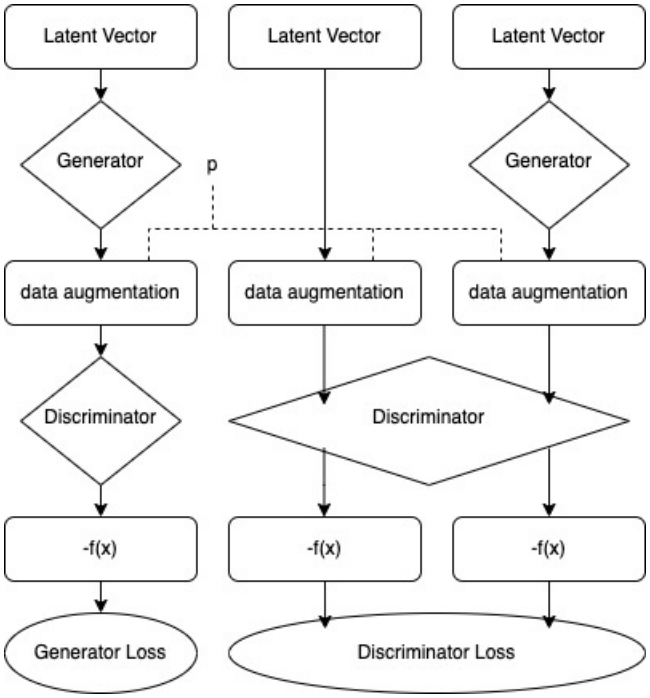

Figure 4. The data augmentation of StyleGAN based on BCR

$$r_v = \frac{\mathbb{E}\left[D_{train}\right] - \mathbb{E}\left[D_{discriminator}\right]}{\mathbb{E}\left[D_{train}\right] - \mathbb{E}\left[D_{generated}\right]} \tag{5}$$

$$r_t = \mathbb{E}\left[sign\left(D_{train}\right)\right] \tag{6}$$

For the two heuristic indicators r, when r=0 means no overfitting, when r=1 means completely overfitting. They adjusted the value of p to make r reach an appropriate value, if the value of r is too large or too small, increase or decrease r by a fixed amount respectively.

4. Experimental process and results

4.1. Running process

All experiments are conducted on Colaboratory, the GPU I used was Tesla P100.

The DCGAN and DRAGAN networks were written by Python 3.7, while the StyleGAN2-ada requires to be running on TensorFlow 1.14 strictly. I've tried to used TensorFlow 1.15, but there was an error. Also, I used a tutorial ipynb file which written by Derrick Schultz to trian my StyleGAN2-ada.

4.2. Results

DCGAN and GRAGAN: The random results generated by DCGAN and GRAGAN after 450 epochs training are shown as figure (5) and (6).

Figure 5. The generated images of DCGAN after 450 epochs

Figure 6. The generated images of GRAGAN after 450 epochs

Also, the random results generated by DCGAN and GRAGAN around 200 epochs training are shown in figure (7) and (8):

Figure 7. The generated images of DCGAN around 275 epochs

Figure 8. The generated images of GRAGAN around 275 epochs

From these figures, we can easily see that the performance of DCGAN is worse than GRAGAN due to the model collapse. And meanwhile, the training rate of GRAGAN is higher than DCGAN, so that we can see the generated portraits' human face of GRAGAN clearly around epoch 200, while the results of DCGAN are blurrier.

Also, the generator and discriminator loss of DCGAN and GRAGAN are shown in figure (9) and (10):

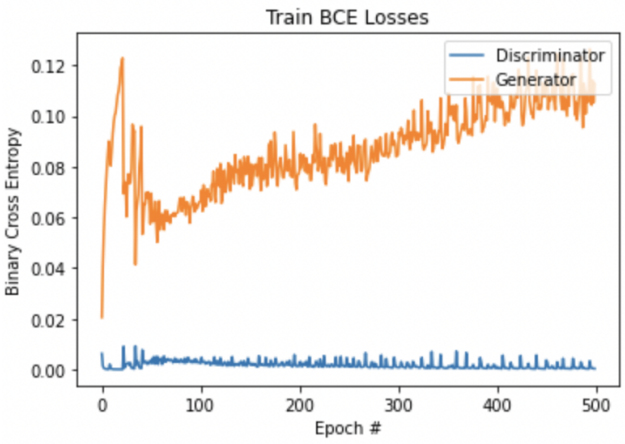

Figure 9. The generator and discriminator loss of DCGAN

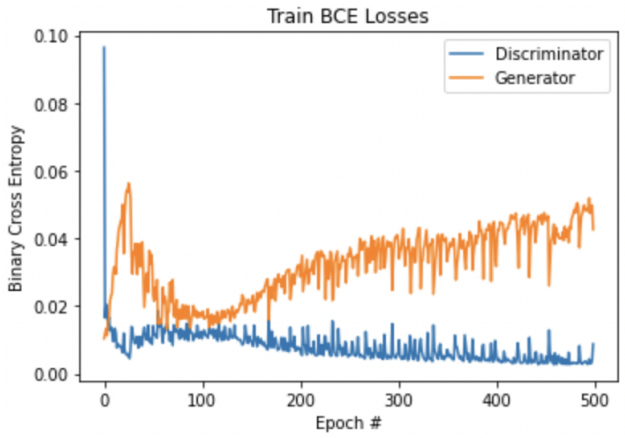

Figure 10. The generator and discriminator loss of GRAGAN

We can easily see that the trend of loss of generator and discriminator of both networks are similar, that the loss of discriminator sharply went down around epoch 50, and then gently increase. Meanwhile, the loss of discriminator is higher than the generator's.

On the other hand, the loss of discriminator of GRAGAN is lower than DCGAN's.

StyleGAN2-ada: As for StyleGAN2-ada, I've trained it for around 250 kimg, and I used ffhq512 as the preprocessing model to train 512 x 512 pixels images for first time.

The generated result of StyleGAN2-ada after training 100 kimg is shown in figure (11).

Figure 11. Generated results of StyleGAN2-ada after training 100 kimg

The generated result of StyleGAN2-ada after training 250 kimg is shown in figure (12).

Figure 12. Generated results of StyleGAN2-ada after training 250 kimg

After that, I used the rained model to generate 6 images randomly, the parameter seeds was set as 600-605.

Figure 13. Randomly results of StyleGAN2-ada (seed = 600)

Figure 14. Randomly results of StyleGAN2-ada (seed = 601)

Figure 15. Randomly results of StyleGAN2-ada (seed = 602)

From these figures, we can see that the model I've trained by using StyleGAN2-ada is very mature, and the protraits it generated are really vivid, the features were captured perfectly.

Also, because I used StyleGAN2 instead of StyleGAN, there is barely noises around the edges or the portraits' hair.

5. Some personal thoughts

During the training time of DCGAN, I've tried some other parameters, image sizes, loss function and different depth of convolutional layers.

5.1. the number of layers

I would like to train the dataset which 128 x 128 pixels, 256 x 256 and 512 x 512, so in other to capture as many as the features when I was using such high qualities images, I added the amount of convolutional layers. I set 6 layers of Conv-BN-Relu for 128 x 128 images, 7 layers for 256 x 256 images and 8 layers for 512 x 512 images.

The results of these changed models are quite bad, which are shown in figure (16), (17) and (18).

Figure 16. Results of 6 layers of DCGAN

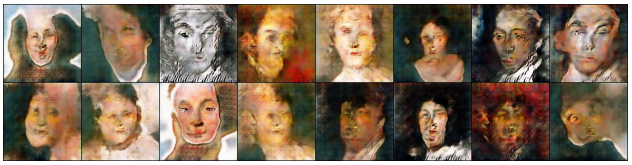

Figure 17. Results of 7 layers of DCGAN

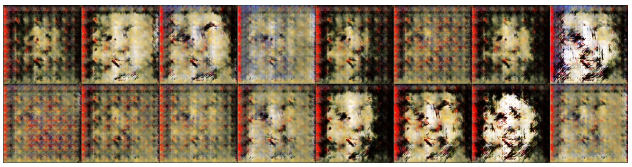

Figure 18. Results of 8 layers of DCGAN

CNN needs to extract features from pictures, and the size of feature map generated by CNN convolution is related to filter size, and its output is affected by input size, filter size, padding size, and stride size. Flatten is performed after CNN, and the output layer is classified, then the picture is too large, and the convolution layer is small, that is, shallow, the dimension of flatten will be high, and the final output layer parameters will be large.

On the other hand, if the depth of the convolution layer is blindly increased, the parameters of the output layer can be reduced to a certain extent, but the depth will be too large, the model will degenerate, and the complexity will be too high, and such a model is also unhealthy. Also, the more features networks get, the more overfitting problems we will face.

So we have to set reasonable image size and number of convolutional layers.

5.2. loss function

I used MSE (main square error) as loss function to train DCGAN firstly, and then I found that I should changed it to BCE (cross entropy error).

Mean square error (MSE) is used to find the average of the squares of the difference between the n outputs of the n samples in a batch and the expected output.

At the same time, cross-entropy error is used to evaluate the difference between the probability distribution obtained by the current training and the true distribution. It depicts the distance between the actual output (probability) and the expected output (probability), that is, the smaller the value of cross entropy, the closer the two probability distributions are(sus, 2017).

There are 2 difference between MSE and BCE:

- **Cross-entropy loss weights update faster**: When the gradient is small, the step size should be reduced (otherwise it is easy to oscillate around the optimal solution), but if MSE is used, when the gradient is small, it is impossible to know whether it is far away from the target or already near the target. The gradient is very small when training place is close to the target or far away from the target(Nielsen, 2015). While BCE's weights are affected by the error, so when the error is large, the weight update is fast, and when the error is small, the weight update is slow.

- MSE is a non-convex optimization problem and Cross-entropy is a convex optimization problem.

But actually, I didn't find obvious difference of the performances of DCGAN when I used BCE and MSE to train my model respectively. I would like to read more articles about that to figure out the applications of both loss function on GANs.

6. Conclusion

In this project, I learned the model structure of DCGAN, GRAGAN and StyleGAN2-ada. According to the knowledge I've got from the process of writing my project, I implemented the DCGAN and GRAGAN models, at the same time, I improved GRAGAN model based on the original one and got a better performance. By using these three models, I trained them on the portrait dataset and generated randomly results with trained models.

This project has broadened my horizons. While learning more about the structure and knowledge of GANs, it made me realize that art and machine learning, two seemingly unrelated things, are closely related.

In recent years, more and more people have been engaged in the use of machine learning for artistic creation, not only limited to photography, painting, but even sculpture, video creation and other aspects.

Some people say that the use of machines for artistic creation is an insult to the artist, but when the camera first appeared, people did not think that photography was an art.

Art transmission through machine learning can not only promote the development of machine learning technology, but also increase the creative inspiration of artists, and even create new art forms.

References

What is the different between mse error and cross-entropy error in nn. [Online], 2017. https://susanqq.github.io/tmp_post/2017-09-05-crossentropyvsmes/.

Machine learning for artists: the latest trends. [Online], 2019. https://www.foundry.com/insights/machine-learning/for-artists?utm_medium=social-post&\utm_source=linkedin&utm_campaign=insights-hub&utm_content=machine-learning-artists.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/arjovsky17a.html.

BrainStation. How ai and machine learning are being used to make art. [Online], 2019. https://brainstation.io/blog/how-ai-and-machin\e-learning-are-being-used-to-make-art.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. Training generative adversarial networks with limited data, 2020.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. On convergence and stability of gans, 2017.

Nielsen, M. *Neural Networks and Deep Learning*. Determination Press, 2015. URL https://books.google.ca/books?id=STDBswEACAAJ.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06434.

Xue, M. and Zhu, C. A study and application on machine learning of artificial intellligence. In *2009 International Joint Conference on Artificial Intelligence*, pp. 272–274, 2009. doi: 10.1109/JCAI.2009.55.

Yi, Z., Wu, G., Pan, X., and Tao, J. The research of anime character portrait generation based on optimized generative adversarial networks. In *2021 33rd Chinese Control and Decision Conference (CCDC)*, pp. 7361–7366, 2021. doi: 10.1109/CCDC52312.2021.9602217.

Zhao, Z., Singh, S., Lee, H., Zhang, Z., Odena, A., and Zhang, H. Improved consistency regularization for gans, 2020.