Chinese Painting Generation Using Generative Adversarial Networks

Guanyang Wang
Stanford University
guanyang@stanford.edu

Ying Chen
Yahoo!
ychen107@stanford.edu

Yuan Chen
Stanford University
ychen12@stanford.edu

Abstract

This project implements different types of Generative Adversarial Networks (GANs) such as cGANs, DCGANs, WGANs, and our modified WGANs on our own Chinese painting dataset to recover original paintings from edge maps, and generate realistic-looking paintings. We also compared the results of different GANs. Empirically, we found that WGANs and our modified WGANs are more stable and are able to generate images with higher quality. In particular, the modified WGANs performs well in getting rid of the problem of mode collapse.

1. Introduction

In recent years, deep learning has been proven to be one of the most powerful tools in artificial intelligence, and is starting changing our lives. There are many successful applications of deep learning. For example, Google DeepMind developed AlphaGo, an artificial intelligence which teaches deep convolutional neural network to play the board game Go [8]. Recently, AlphaGo defeated Ke Jie, world's top-ranked player for the third consecutive game. In the context of computer vision, deep learning has been widely used in face recognition, street view image recognition, image retrieval, and so on.

Another interesting discussion today within deep learning is how it might impact and shape our future cultural and artistic production. However, generating artworks is a challenging task, especially for Chinese landscape paintings. The difficulties lie in the following aspects: (a) The background of Chinese landscape paintings are harder to recognize, Chinese painters like to use clouds and fogs to create a hazy atmosphere; (b) There are usually many objects such as mountain, river, bridge and fog in one Chinese painting; (c) The shapes of those objects are usually not regular; (d) There is no existing open source datasets of Chinese paintings.

The goal of our project is to create Chinese Paintings using deep neural network. We use Deep Convolutional GANs (DCGANs) and Wasserstein GANs (WGANs) to train thousands of Chinese landscape paintings and then synthetically generate realistic paintings. We also use conditional GANs (cGANs) to tackle the image-to-image translation problem. The input is the edge map of a painting and we want to reconstruct that target image.

2. Related Work

GAN was first introduced by Goodfellow et al. [10] in 2014, in this paper, the authors use GANs to generate natural images using the MNIST [17], TFD [23] and CIFAR-10 [16] datasets. It turns out that images generated by GANs are significantly sharper than those trained using other methods based on maximum likelihood training objectives. Later on, GANs were primarily applied to modelling natural images and different variations of GANs were introduced. In 2014, Mirza et al. proposed conditional generative adversarial nets (cGANs) [20], which control on modes of the data being generated. However, GANs are known to be unstable to train and often suffer from the problem of mode collapse. In 2015, Radford proposed a more stable architecture called deep convolutional generative adversarial nets (DCGANs) which scales up GANs using convolutional neural networks (CNNs) to model images. Recently, Arjovsky et al. [5] defined a new form of GANs called Wasserstein GANs (WGANs). WGANs improve the stability of learning and are able to generate images of different modes. There are many other variants of GANs such as LSGANs [19], Bayesian GANs [25], improved WGANs [11], Cramer GANs [6].

Thanks to the strong ability of generating high-quality images by GANs, there are lots of interesting applications of GANs including face generation [9], text to image synthesis [22], image to image translation [14], unpaired image to image translation [27], auto colorization [15], photo realistic single image super-resolution [18].

However, there has been very limited papers in trying to generate artworks using GANs, recently, ArtGANs [24] was introduced to generate western paintings. ArtGANs use conditional GANs based on the label information. But one concerning is some western paintings, especially for Abstract paintings, portray objects that have been "abstracted"

from nature. Those kinds of paintings might be harder for non-experts to understand than representational paintings. Thus it might not be easy to compare the results of Art-GANs with other variants of GANs visually.

3. Methods

In this section, we will first briefly review GANs and then introduce the algorithms of several variants of GANs such as DCGANs, WGANs and cGANs.

3.1. Generative Adversarial Networks

Generative Adversarial Networks (GANs), as its name suggests, contains two adversarial networks, a generator and discriminator. It can be formulated as a two-player minimax game. The generator G generates images from random noise, i.e., it learns a mapping from a random noise vector z to an image y, $G : z \rightarrow y$. The discriminator D simultaneously learns a mapping from an image x to some values between 0 to 1, which indicates the probability that the input comes from the true data distribution. The goal of the discriminator D is to distinguish samples from the generative model and samples from the training data, while the goal of G is to generate images which is indistinguishable from those from training images (i.e., fool the discriminator D), the procedures are shown in Figure 1 [3]. At the equilibrium point, the generator will generate real images and the discriminator will output probability of 0.5.

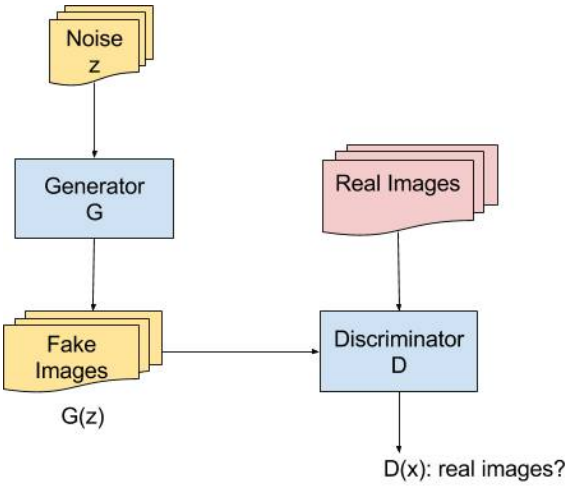


Figure 1: Structure of GAN

To be specific, the procedure can be regarded as minimax the following target function.

$$\begin{aligned}\min_G \max_D \mathcal{L}_{GAN}(D,G) =& \min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] \\ &+ \mathbb{E}_{z\sim p_z}[\log(1-D(G(z)))].\end{aligned}$$

3.2. Deep Convolutional Generative Adversarial Networks

Deep Convolutional Generative Adversarial Networks (DCGANs) is an improved version of original GAN which adds convolutional layers in both generator and discriminor's architecture. The architecture differs from original GANs in the following main aspects, as suggested by [21]:

- Use strided convolutions for the discriminator and transpose convolutions for the generator.
- Use batch normalization in both generator and discriminator
- Remove fully connected hidden layers for deeper architectures.

DCGANs is more stable than original GANs and output sharper images, but there are still some forms of instability. Both GANs and DCGANs suffer from the problems of mode collapse and gradient vanish.

3.3. Wasserstein Generative Adversarial Networks

Wasserstein Generative Adversarial Networks (WGANs) is a recently proposed GAN training algorithm which has both nice theory supplement and good empirical results on most common GAN datasets. In theory, the loss function in original GANs uses the Jensen-Shannon divergence (JS divergence) to characterize the distance between the probability distribution of the generated images P_g and the real images P_d. However, if the support of the two probability measures have no overlap, then by definition the JS divergence of P_g and P_d will be a constant log2, which causes the problem of gradient vanish. In particular, suppose the support of P_g and P_d are low-dimension manifolds in finite dimension Euclidean space, then the two measures have no overlap support almost surely!

These insights suggest us to find another metric to characterize the distance between two probability measures in our case. It turns out that the Wasserstein distance could be a proper choice. The Wasserstein distance of P_d and P_g is defined by:

$$W(P_d, P_g) = \sup_{|f|_{L\leq 1}} \mathbf{E}_{P_d} f(X) - \mathbf{E}_{P_g} f(X),$$

here $|f|_{L\leq 1}$ means $|f(x) - f(y)| \leq |x - y|$ for any x, y. Although the above formula is hard to evaluate directly,

the idea of WGANs is to use a neural network to approximate the Wasserstein distance. The algorithm procedure of WGANs is shown in Figure 2 from [5].

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.
while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Figure 2: Algorithm of WGAN

In particular, WGAN is different from original GAN in the following main aspects:

- Do not apply sigmoid at the output of the discriminator
- Remove log term in the loss function
- Use RMSProp instead of ADAM
- Clip the weight of discriminator.

3.3.1 Modified Wasserstein Generative Adversarial Networks

While training our own dataset, we found that Wasserstein GAN generates visually better results than DCGAN, and is much more stable. However, it seems that the images generated by WGAN still suffer from the problem of mode collapse, although better than DCGAN. Thus we modified the algorithm of WGAN to the following form:

- Still apply sigmoid in the output of D, so the output of the discriminator is still a probability
- Train the discriminator three times and train the generator twice in each iteration.

In conclusion, the difference between DCGANs, WGANs and modified WGANs is shown in the following table.

DCGAN	WGAN	Our modified WGAN
Basic Model	1. Remove log term in the loss function 2. Use RMSProp Optimizer 3. Clip the weight 4. No sigmoid on discriminator output	1. Remove log term in the loss function 2. Use RMSProp Optimizer 3. Clip the weight

Figure 3: Comparsion between DCGAN, WGAN and modified WGAN

3.4. Conditional Generative Adversarial Networks

Conditional GANs preserves almost all the structures of GANs, there is still a generator G and discriminator D and they play a minimax game. The difference is the input of G and D. Now the generator G learns a mapping from observed image x and random noise vector z, to an image y, $G : (x, z) \rightarrow y$. Similarly, D learns a mapping from (x, y) to $[0, 1]$, trying to distinguish the image from training data and the generator. The training procedure is diagrammed in Figure 4 from [14].

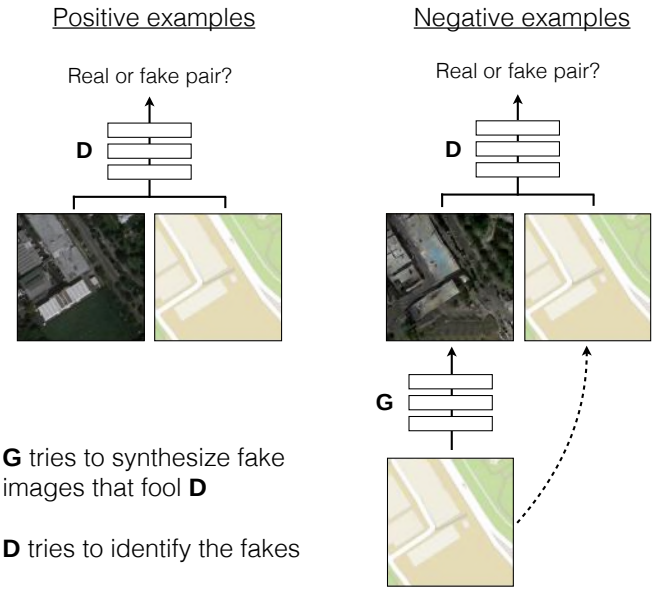


Figure 4: Structure of cGAN

To be specific, the procedure can be regarded as minimax the new target function.

$$\min_G \max_D \mathcal{L}_{cGAN}(D, G)$$

where

$$\mathcal{L}_{cGAN}(D, G) = \min_G \max_D \mathbb{E}_{x,y\sim p_{\text{data}}(x,y)}[\log D(x, y)] + \mathbb{E}_{z\sim p_z, y\sim p_y(y)}[\log(1 - D(y, G(y, z)))].$$

4. Dataset and Features

Since there is no previous painting generation project on Chinese painting, it could be a problem to get the dataset. There is a paper claiming they have made a dataset of Chinese paintings and calligraphy, but provides no detail on how to access the dataset [13]. We tried to contact the authors but got no response yet.

We decided to get our own dataset by scraping the images on Google and Baidu. To maximize the number of

images, we tried different keywords (i.e. different types of Chinese paintings) to search. Our keywords include *guohua, gongbihua, shuimohua, shanshuihua,* etc. We used the methods described in [26] to get the URL of the images, and then used the code in [12] to download the images. After we got the images, we preprocessed them to be the input of the model. An image was first reshaped to be 256×256. If the image is not square, the shorter side is reshaped to be 256 and the central part of the image is taken. This has the advantage of keeping the aspect ratio of the pictures without losing the most important information at the center. The reshaped paintings also act as the input of DCGAN and WGAN to generate new painting. Then, we extracted the edges of the painting using Canny edge detection [7]. The preprocessing code can be found in [4]. Given pairs of edges and paintings as inputs, we can train the cGAN model to color the pictures. Figure 5 shows a pair of the painting and the edge in our dataset, together with the original painting.

One problem of this method is that sometimes the search engines give irrelevant or nonsense results. This happens when you reach the end of the search results. In many cases there seems to be a boundary between good and bad results. We avoided downloading bad results after that boundary. Sometimes there are bad results in the middle. We just remove them once they are discovered. Another problem of this method of data acquisition is that there could be duplicate results from different keywords or search engines. This changes the weight of data points in the training process. To address this problem, we used a software called Gimini 2 [1]. This software is very powerful, it could even find paintings that differ in tones but are otherwise the same. After that, we also checked if there are duplicate paintings in the cropped files. Since the cropping methods for different files are the same, if their original files are the same, the cropped should be the same, too. We wrote some script to check if there are the same files in the cropped paintings and we found none. In the end, we got a total of 5798 valid paintings in the dataset.

5. Experiments and Results

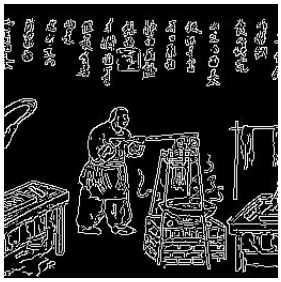
5.1. Image reconstruction from edge maps using cGANs

We built a base model using the convolutional network architecture from [21]. Both generator and discriminator use modules of the form Convolution-BatchNorm-ReLu. The implementation is built from project pix2pix [4].

The dataset is divided into a training set (contains 80% images) and a test set (contains 20% images), the whole training procedure is done on the training set. The model is trained by 200 epochs and the following figures present the result of several typical test examples. During the training,

(a)

(b)



(c)

Figure 5: An example of the pre-processed data point. (a) The original figure, (b) the cropped figure, (c) the edge of the cropped figure.

we will feed in the edge of the Chinese painting as input, while the target picture is the colored version of the picture (corresponding Chinese painting). In this case, the generator is trying to learn how to reconstruct an edge image.

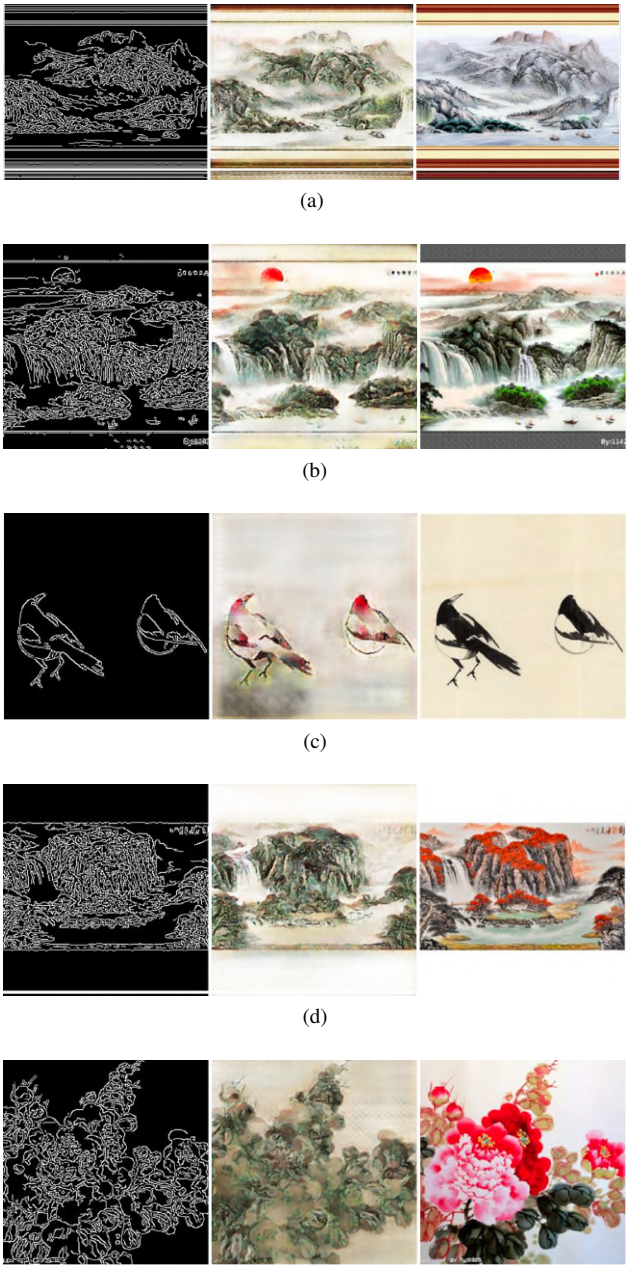
The first two rows of Figure 6 are successful examples of our results, actually it is nearly indistinguishable between the fake image and the real image. It is worth mentioning that in the second example, although the sun is partially obscured by the clouds, cGAN is still able to recognize the sun and give correct colors. The third example is quite interesting, the target image is actually a black-white image, but the bird is dyed reasonably, just like it may appear in other Chinese paintings. This result indicates that the tones and colorization methods of Chinese paintings are learned by cGANs during training. The last two examples are two failure examples, the object is dyed green however the ground truth should be red. One possible explanation is that our dataset consists many landscape paintings , while the main objects in landscape paintings are mountains, trees and grasses, and they are often dyed green. This may result in errors when the test examples have bright colors like red.

Figure 7 shows the losses of our model while training.

5.2. Chinese painting creation using DCGANs, WGANs and modified WGANs

5.2.1 Experiment Settings

The neural network structure of DCGANs, WGANs and modified WGANs are basically the same. We use strided convolutions for the discriminator and transpose convolu-



(a) (b) (c) (d) (e)

Figure 6: Five typical examples of the training output. The left column are our input (edge maps), the middle column are our output (paintings reconstructed by cGANs), the right column are our target images

tions for the generator. Batch normalization is used in both discriminator and generator. However, the $-\log$ term is removed in WGAN and modified WGAN, which is different from DCGAN. For DCGAN we use the Adam optimizer with learning rate 0.0002, for WGAN and modified WGAN we use RMSProp optimizer with learning rate 0.00005.

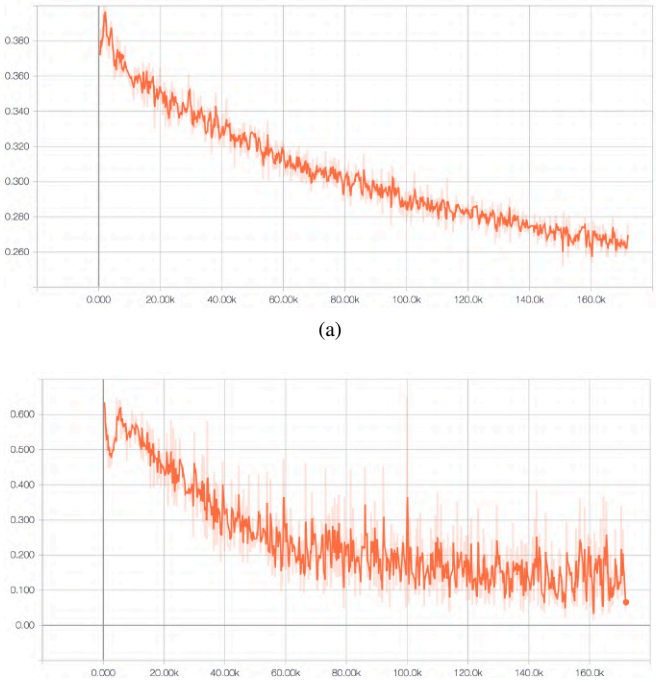


Figure 7: (a) The loss of cGAN generator while training; (b) The loss of cGAN discriminator while training.

The weight of discriminator is clipped into $[-0.01, 0.01]$ in WGAN and modified WGAN, but not in DCGAN. The output of the discriminator is a probability for DCGAN and modified WGAN (apply the sigmoid in the discriminator), however, the discriminator of WGAN just outputs a score, which can be negative. The input are 3-channel images of 256×256 pixels in size. The code for the base DCGAN model we used can be found here [2].

5.2.2 Experiment Results and Comparisons

Our first experiment trained each of DCGAN, WGAN and modified WGAN for 200 epochs, with batch size 64, and output size $32 \times 32 \times 3$. The final results are shown in Figure 8. We found that under this setting, all these three algorithms are capable to create reasonable images, but it seems the images generated by WGAN and modified WGAN are visually better than DCGAN. For example, about half of the outputs of DCGAN is dyed gray, which can be considered as the color of mountains or stones, but the outputs of WGAN and modified WGAN are obviously more colorful and have larger variety. Meanwhile, the outputs of WGAN and modified WGAN are more vivid and have a clearer outline. However, it is hard to analyze in details due to the low pixel and large batch size of our output. Therefore we did a second experiment, all the three algorithms are trained for 500 epochs, with batch size 16, and output

size $256 \times 256 \times 3$.

Figure 9 presents the artwork synthetically generated by DCGANs, WGANs and modified WGANs after 500 epochs. We can discuss the result in the following points:

- Image quality: Both three algorithms generate high-quality images. However, comparing with WGANs and modified GANs, the image quality of DCGANs is relatively poorer. WGANs and modified GANs generate sharper lines and more colorful images than DCGANs. In particular, images generated by modified GANs has a rich stereoscopic feeling. For example, the image in the first row, second column in Figure 9 (c) depicts the scene of a waterfall flows down from the mountains, which is very compelling.

- Mode collapse: As shown in Figure 9, DCGAN algorithm has significant degree of mode collapse, more than half of the images look similar to the painting in the second row, first column in Figure 9 (a) (looks like stones). WGAN performs better than DCGAN, but still suffers from the problem of mode collapse. About half of the images look similar to the painting in the first row, first column in in Figure 9 (b) (although they have different color styles). We did not see evidence of mode collapse for the modified WGAN algorithm. Thus we claim that our modified WGAN performs best in getting rid of mode collapse among these three algorithms on our Chinese landscape painting algorithm.

- Stability: From Figure 10 we find that WGAN and modified WGAN are much more stable than DCGAN. The shape and tone of WGAN and modified WGAN basically stay the same during between two consecutive epochs, while DCGAN outputs change drastically.

- Learning Speed: As shown in Figure 11, the training speed of modified GANs is the slowest. The shapes of DCGAN and WGAN outputs after epoch 100 is much clearer than modified GANs. One explanation is comparing with WGAN, we add sigmoid in the output of discriminator, and the derivative is $\sigma(x)(1-\sigma(x))$ which is less than 1, this will slow down the training procedure. This can be regarded as a downside of modified WGAN.

5.2.3 Quantitative Results

Another interesting phenomenon happens in the plots of discriminator losses, as shown in Figure 12. The loss of DCGAN looks fine, however, as suggested by [5], the discriminator loss should be an approximation of the Wasserstein distance between P_d and P_g, and this quantity correlates well with the quality of the generated samples. However, in our examples, the losses of both WGAN and modified WGAN oscillate a lot and do not have any tendency of convergence. On the other hand, WGAN and modified WGAN still generate high-quality images when the number of epochs increases.

It is unclear to us why this phenomenon happens, one possible explanation is: the loss of the discriminator is still an approximation of Wasserstein distance, therefore in our case maybe the neural network we use does not provide an accurate approximation to $\sup_{|f|_L \leq 1} \mathbf{E}_{P_d} f(X) - \mathbf{E}_{P_g} f(X)$, the Wasserstein distance between P_d and P_g (basically we are using composition of linear and nonlinear functions to approximate the best f for Wasserstein distance, but the basis functions we used in our neural network may not be a suitable choice), thus the loss does not seem to have correlation with the quality of the output images. The loss oscillates might because the discriminator is competing with the generator while training.

If the above explanation makes sense, then a natural question is: is there a way to build the discriminator such that it could approximate the Wasserstein distance pretty well? This question may be of interest to both theorists and practitioners.

6. Conclusion and future work

We implement different GAN algorithms on our own Chinese painting dataset. The cGAN algorithm for image reconstruction works pretty well overall, the algorithm could learn the tones and colorization methods of Chinese paintings. As for artwork generation, we implemented DCGAN, WGAN and our modified WGAN algorithms and compared them. Empirically, WGAN and modified WGAN create visually better images than DCGAN, and are much more stable than DCGAN. Meanwhile, modified WGAN performs best in creating larger varieties of paintings, which may suggest it could be a candidate to solve the mode collapse problem. However, quantitatively, WGAN and modified WGAN have oscillated loss functions, which does not have strong positive correlation with the quality of the output images. This phenomenon is different from [5] and the reason behind this is still open to us.

In the future, there are several things that are worth trying. First we could implement our modified WGAN on other datasets to see if it is still robust in providing large variety of outputs, or explain this theoretically. Then we could keep investigating the reason of the oscillated discriminator loss of WGAN and try to carry out a reasonable explanation. Last but not least, we could explore other kinds for GANs (maybe use the total variation distance, Hellinger distance or Levy-Prokhorov distance) that may lead to promising empirical results.

(a) (b) (c)

Figure 8: $32 \times 32 \times 3$ output after 200 epochs. Left: DCGAN, middle: WGAN, right: modified WGAN

(a) (b) (c)

Figure 9: $256 \times 256 \times 3$ output after 200 epochs. Left: DCGAN, middle: WGAN, right: modified WGAN

(a) (b) (c)

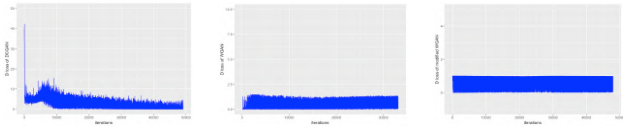
(d) (e) (f)

(g) (h) (i)

Figure 10: Results of three consecutive epochs. Left: DCGAN, middle: WGAN, right: modified WGAN

(a) (b) (c)

Figure 11: The results after epoch 100 using DCGAN (left), WGAN (middle), modified WGAN (right) respectively.



(a) (b) (c)

Figure 12: (a) D loss of DCGAN, (b) D loss of WGAN, (c) D loss of modified WGAN.

References

[1] Gimini 2: The intelligent duplicate file finder. https://macpaw.com/gemini.

[2] A tensorflow implementation of "deep convolutional generative adversarial networks". https://github.com/carpedm20/DCGAN-tensorflow.

[3] Deep learning 13: Understanding generative adversarial network. https://ireneli.eu/2016/11/17/, 2016.

[4] M. Akten. Source code and pretrained model for webcam pix2pix. https://github.com/memo/webcam-pix2pix-tensorflow, 2017.

[5] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[6] 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. *arXiv preprint arXiv:1705.10743*, 2017.

[7] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[8] C. Clark and A. Storkey. Teaching deep convolutional neural networks to play go. *arXiv preprint arXiv:1412.3409*, 2014.

[9] J. Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014(5):2, 2014.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[12] Z. He. Learning ganfrom fundamental principles to implementing a demo (in chinese). https://zhuanlan.zhihu.com/p/24767059, 2017.

[13] H.-z. L. Hong Bao, Ye Liang and D. Xu. A dataset for research of traditional chinese painting and calligraphy images. In *International Proceedings of Computer Science and Information Technology*, volume 42, pages 136–143. International Association of Computer Science and Information Technology Press, 2010.

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[15] S. Koo. Automatic colorization with deep convolutional generative adversarial networks.

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[17] Y. Lecun and C. Cortes. The MNIST database of handwritten digits.

[18] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[19] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016.

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

[21] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[22] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 3, 2016.

[23] J. M. Susskind, A. K. Anderson, and G. E. Hinton. The toronto face database. *Department of Computer Science, University of Toronto, Toronto, ON, Canada, Tech. Rep*, 3, 2010.

[24] W. R. Tan, C. S. Chan, H. Aguirre, and K. Tanaka. Artgan: Artwork synthesis with conditional categorial gans. *arXiv preprint arXiv:1702.03410*, 2017.

[25] D. Tran, R. Ranganath, and D. M. Blei. Deep and hierarchical implicit models. *arXiv preprint arXiv:1702.08896*, 2017.

[26] K. Ullah. How to download all full sized images from google image search without any software - 2 easy steps only. https://32hertz.blogspot.com/2015/03/download-all-images-from-google-search.html, 2015.

[27] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.