Implementation of Realistic Face Generation using Generative Adversial Network

Nikhil Bathini
Computer Science
Binghamton University
Nbathini1@binghamton.edu

Nikhil Bathini
Computer Science
Binghamton University
Nbathini2@binghamton.edu

***Abstract*—Generating photos with realistic imagery using a computer, a Deep Convolutional method. This study develops a model for creating images using the Generative Adversarial Network (GAN). By removing the completely connected layer from the conventional network and using batch normalization and deconvolution processes, this research develops a model consisting of a discriminator network and a generator network. The diversity and quality of the created image are also measured in this study using a hyper-parameter. The model's experimental results on the CelebA dataset demonstrate that it has superior performance for picture generation, particularly for facial image generation. We have further used a small part of the code to generate a photo using the sketch.**

1. Introduction

So generating an artificial image which is as good as the original is the task we are going to achieve here. Is this worth doing it? Yes, the main purpose developing an artificial image set is to increase dataset which in train used for training the face recognition algorithms and make them accurate as possible. Automatic image production based on depth models has drawn more and more interest as deep learning in computer vision has become more widely used. The underlying distribution rule in the data can be quickly mined to produce images with a similar distribution by utilizing the depth model's robust learning capabilities. These created images can be used in a variety of situations, including automatic image synthesis, age-based face image prediction, and creating creative images. High-quality generated images can also be used to increase the dataset's volume of image data, which can reduce the requirement for a large number of training samples during the development of deep learning models and improve the accuracy of practical applications like face recognition.

The use of GAN is widely accepted in the present study on the automatic production of images. In the end, the confrontation between the discriminator and the generator over the data distribution will bring the two into equilibrium.

2. Related work

Deep learning algorithms must learn how to create the data in order to comprehend the input data. Using a generation model that learns the rules of the data and determines the appropriate distribution to represent it is the most promising strategy. We can even create samples that are not part of the training set but have the same distribution by learning to create models. The 2014 proposal of the Generative Adversarial Network [1] as a new generation model framework can produce a composite image that is superior to the prior generation model, and it has since grown to be one of the most well-liked study areas. The Generative Adversarial Network consists of two neural networks: a generator and a discriminator. The generator tries to spoof the discriminator by generating a real sample, while the discriminator tries to tell the difference between the real sample and the one generated by the generator

The area of GAN that has received the most research is by far image generation. Hierarchical approaches, iterative methods, and direct methods are the three basic GAN picture generating techniques. The algorithm under the hierarchical approach includes two generators and two discriminators in its model; the interaction between

the two generators can be parallel or in series, as in Secure Steganography based on GAN (SS-GAN) [4]. The different generators have different functions. Comparing iterative approaches vs hierarchical methods is not appropriate. The models first create images in a range of resolutions, and then each generator recreates the results' specifics. The iterative technique can leverage weight sharing between the generators when the generator has the same structure, whereas the hierarchical methods typically cannot. Stack GAN is a traditional type [5]. The direct techniques use a generator and a discriminator as part of their model, and their generator and discriminator have a simple, branch-free structure. This is true of many GAN models, including DCGAN [3] and Spectral Normalization (SN-GAN) [7]. One of these is the DCGAN [3], which is a classic since many later models adopted its structure and several GAN model network designs, like cGANs [8,] employ it. Compared to layered and iterative procedures, this methodology is easier to develop and apply and typically produces good results.

3. Method

The two-player game in game theory serves as the basis for GAN. A generating model and a discriminative model, respectively, make up the two actors in the GAN model. Create a model G to capture the distribution of the sample data, and create a sample with noise z that closely resembles the genuine training data (uniform distribution, Gaussian distribution, etc.). The objective is to produce samples that are as accurate as actual samples; the discriminant model D is one classifier that calculates the likelihood of producing samples from training data (rather than the generated data). D produces a significant probability if the sample is drawn from actual training data and a modest probability otherwise.

3.1 Model Architecture

This model consists of two parts. A Discriminator part and a Generator part

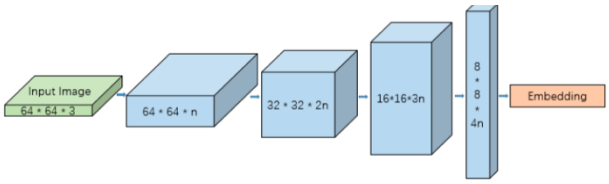


Fig.1 Discriminator Architecture

Discriminator D: We employ a depth encoder and decoder along with an auto-encoder. Each layer has two convolutions. The convolution filter is linearly expanded at each subsampled value to achieve subsampling with a stride of two and to perform nearest neighbor up sampling. As illustrated in Fig. 1, the discriminator receives the three-channel image, acquires the feature map by repeated convolution operations, maps it to a scalar value after deforming it into a vector, and utilizes the Sigmoid function to calculate the discriminator's loss function value. as shown in Fig. 1

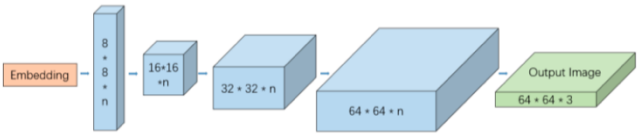


Fig.2 Generator Architecture

Generator G: While using different weights, we employ the same architecture as the discriminator decoder. The fully connected layer's first layer gets the random noise signal vector, which is multiplied by the weight matrix to create a three-dimensional tensor. The subsequent levels are utilized for the deconvolution operation, which uses 22 steps. A three-channel image is produced using the Relu activation function and the tanh activation function in the final deconvolution layer, as shown in Fig. 2.

The Discriminator D, job is to identify which of the image is a real one or the fake one, where as the Generator G job is to generate sample images which are as good as the original ones. As we earlier said this algorithm is a two parts based algorithms in which classification and generate of samples are done at the same time.

3.2 **Parameters**

The best work condition for the model is it would be good to generate a realistic face as possible.

However, there is currently a problem: if the discriminator is unable to distinguish between the generated sample and the real sample, then their anticipated errors and error distribution will be identical. To address this issue, we provide a hyper-parameter, which can assist the generator and discriminator in balancing the assignment jobs, preventing the issue from occurring.

The discriminator's two objectives are to separate the generated picture from the real image and to self-encode the real image. These two objectives are balanced by this hyper-parameter. The optimal value of is between [0, 1], as a low value will lead to poor picture variety because the discriminator is overly worried with self-encoding the genuine image.

3.3 **Loss Function:**

Wasserstein distance. To match the self-loss encoder's distribution to the loss based on the Wasserstein distance, we utilize a self-encoder as a classifier.

The Earthmover (EM) distance, commonly known as the Wasserstein distance, is described as follows:

$W(P,P)$= inf $(^{r\ g}\ \gamma\sim\pi\ \prod(P_N P_e)\ ^{(x,y)\sim\gamma}\quad E$ $[\|x-y\|]$

- (P , P) is a set of all possible joint distributions of $P\ rgr$ and P . Conversely, the edge distribution of each of g (P ,P) is P and P . For each possible joint $rgrg$ distribution γ, a real sample x and a generated sample y can be obtained from the middle sample (x, y) obeying the γ distribution.
- The Wasserstein distance is smooth, and it provides a meaningful gradient when the parameters are optimized using the gradient descent method.
- Calculate the distance of the pair of samples$\|x-y\|$; we can obtain the expected value of the distance of the sample under the joint distribution γ. The lower bound of this mean value in all possible joint distributions. It is the auto-encoder.

G Loss and D Loss. According to the principle of GAN confrontation, the goal of D is to enlarge the distance between the two distributions, that is, to maximize Wasserstein distance: $W(P , P)$, and G to minimize it. The loss function is as follows:

$$\begin{cases} L_D = k_t \times L\big(G(z_D)\big) & for\ \theta_D \\ L_G = L\big(G(z_G)\big) & for\ \theta_G \\ k_{t+1} = \gamma[L(x) \times L_G] + k_t & for\ step\ t \end{cases}$$

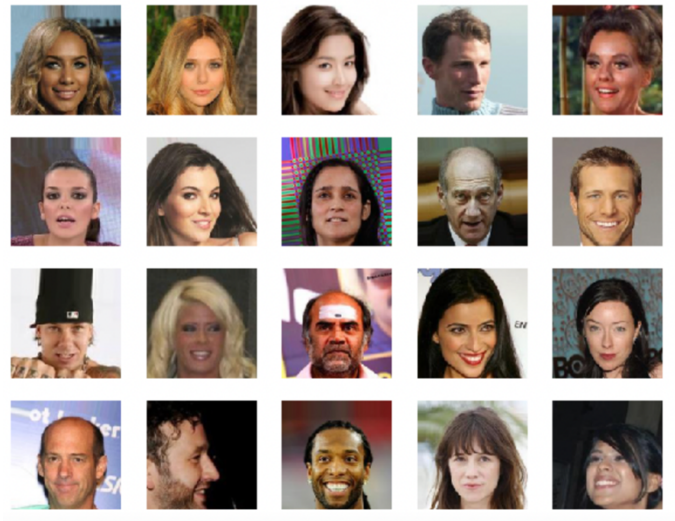
4. **Experimental Setup**

In this paper we run our model through the dataset below dataset and compare the accuracy with the pre-existing models DCGAN and WGAN.

We have used the same model to generate the face images from the respective sketches.

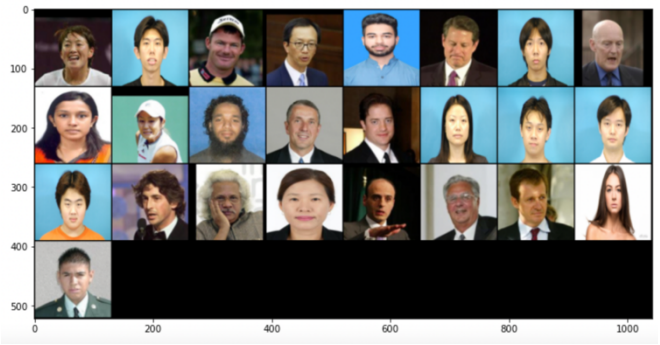
4.1 Dataset

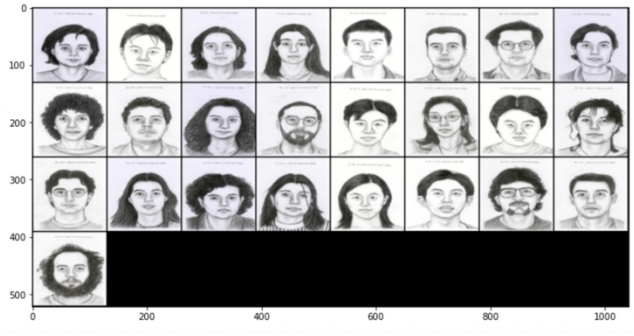
On the CelebA dataset, this paper trained and tested the model. The CelebA Face Dataset [2] is a highly helpful dataset for face-related studies. It is free data from The Chinese University of Hong Kong and contains 202,599 face photos of 10,177 celebrity identities, all of which have been feature-marked. The label for each face is in the identity CelebA.txt file, while each face map's feature tag is in the list celeba.txt file. Whether the person is wearing spectacles, for instance. We create a csv file from the txt file and then separate the facial image based on the csv information. Each folder serves as a representation of an individual, yielding a single dataset of images.



Sample data from the dataset

Sample dataset from my implementation of the model from the research paper.





4.2 **Parameters Setting**

We set our learning rate as 1e-4 and use Adam optimizer for training of our model. Throughout the experiments it's found to be that results are best when learning rate is 0.9 and for every 2000 iterations.

4.3 **Results**

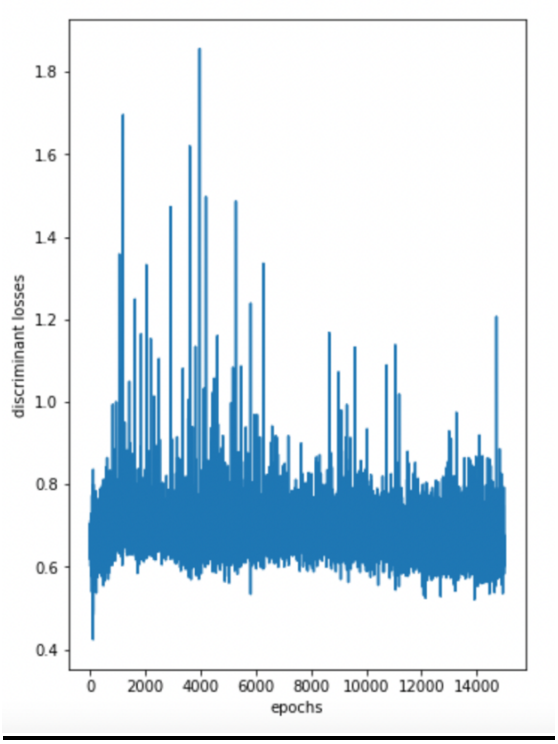
The following are the photos of the following executions:

We also trained the classic DCGAN and WGAN model, as shown in Fig.5, our model has a better performance in generating face images

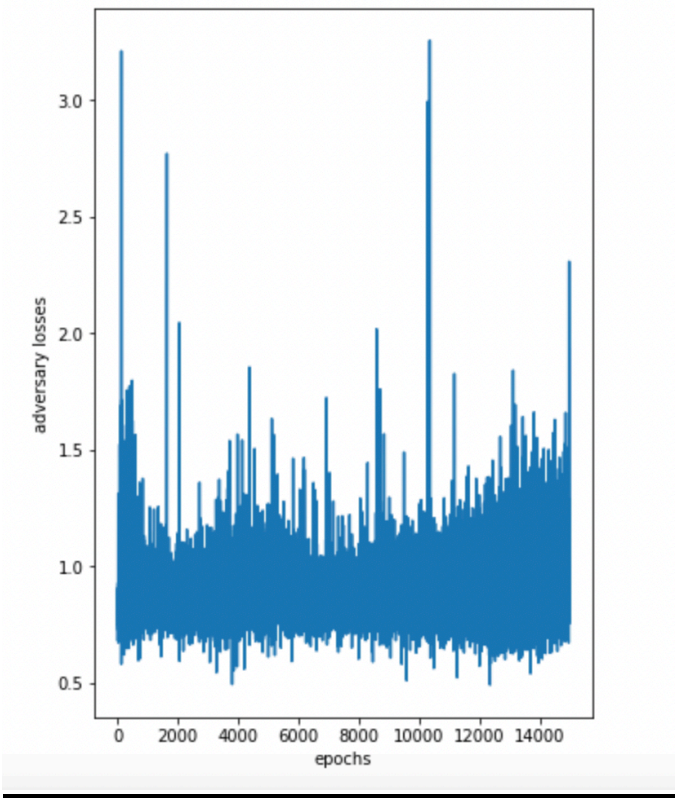
(b)Comparison with other models

(c)Sample output of our model.

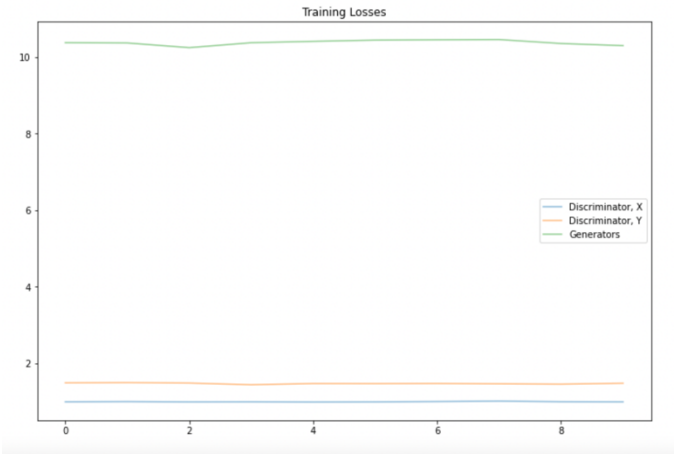
4.4 **Loss function Graph**



(a) Loss function of Discriminator



(b)Loss function of Generator



(c) loss function of our model

The above loss function can be much better if the training value was more. The training time of the model is nearly 23hrs.

4.5 Test and Verify

The dataset's training picture data can be supplemented by high-quality produced images, reducing the need for a substantial amount of training samples during the development of deep learning models.

Table 1 The results of Face recognition

	Can't detect face	Can detect faces but can't identify who
DCGAN	281/1000	0/1000
WGAN	123/1000	0/1000
Our Model	19/1000	0/1000

As indicated in Table 1, we utilize the open source project Face recognition1 on GitHub, which is based on the deep learning model dlib, to recognize the face images produced by three models in order to confirm the realism of the generated face image (DCGAN, WGAN and our model). The findings indicate that our model's generated face image has the greatest recognition rate, at up to 98.10%.

5. Conclusion

This study creates an image creation model using the TensorFlow deep learning framework and suggests a new network architecture based on the Generative Adversarial Network. For the

experiment, we trained DCGAN, WGAN, and our model using the CelebA dataset. We also use the Face Recognition1 open source code on GitHub to find and identify the faces produced by the three models. The outcomes demonstrate that our approach performs better at generating facial images.

6. GitHub link of Code

https://github.com/ageitgey/face_recognition

Our Code:

REFERENCES

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, X. Bing, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," in Advances in Neural Information Processing Systems 27, Curran Associates, Inc., 2014, pp. 2672--2680.

[2] Z. Liu, P. Luo, X. Wang and X. Tang, "Deep Learning Face Attributes in the Wild," in The IEEE International Conference on Computer Vision(ICCV), 2015.

[3] A. Radford, L. Metz and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," Computer Science, 2015.

[4] H. Shi, J. Dong, W. Wang, Y. Qian and X. Zhang, "SSGAN: Secure Steganography Based on Generative Adversarial Networks," in Advances in Multimedia Information Processing -- PCM 2017.

[5] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang and D. N. Metaxas, "StackGAN: Text to Photo- Realistic Image Synthesis With Stacked Generative Adversarial Networks," in The IEEE International Conference on Computer Vision (ICCV), 2017.

[6] M. A. Bottou, S. Chintala and Leon, "Wasserstein Generative Adversarial Networks," in Proceedings of the 34th International Conference on Machine Learning, 2017.

[7] T. Miyato, T. Kataoka, M. Koyama and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," CoRR, 2018.

[8] T. Miyato and M. Koyama, "cGANs with Projection Discriminator," CoRR, 2018.