Anime Face Generation using Generative Adversarial Network

Aleksander Madajczak
The University of Oulu
Pentti Kaiteran katu 1, 90570 Oulu, Finland
Aleksander.Madajczak@student.oulu.fi

Abstract

In the world of NFTs and generated collectibles anime girls can be very attractive and lucrative digital good to posses. The recent advancements in Deep Learning and Artificial Intelligence open up new world of possibilities. The generation of such portraits can be accomplished with the easy generative adversarial network. This work demonstrates simple and straight forward ways of using the DCGAN architecture to obtain bright new and unique 'waifus' the heart can desire.

1. Introduction

The abilities of the neural networks can nowadays amaze even the most bitter sceptics. With advancements in image classification and object recognition by the Convolutions Neural Networks (CNNs) being at the forefront of that excitement. Nevertheless the ability to generate fictional yet believable image can be arguably the most exiting part of the artificial intelligence. Using the Generative Adversarial Network (GAN) one can generate an image of a dog or a face of a non existing human with stunning resemblance. This wide range of possibilities opens up the way for generating almost anything provided one have a vast dataset beforehand.

The anime characters are beloved all over the world. The most impressive ones are design and drawn by a skilled artists which can take time and effort. On the other hand many people would like to own one, given that the era of the NFTs and digital art is in full swing.

To meet that demand it is proposed to generate such images by using GAN networks. The model would train on the available images and then it will generate unique anime girl face with preferably quality not different from the original images. Unsurprisingly there are multiple websites offering such services [10] [4] that use modern GAN architectures.

This project will try to accomplish similar thing, but using older and simpler architecture from 2016 called DCGAN [17]. This model uses small 64 by 64 pixel images and overall should have simpler architecture with quicker training times than those used in the state of the art. In this project the question about the feasibility of generating believable images by such models will be answered and potential improvements to the network will be suggested.

2. Related Works

The techniques of generating believable images had become very effective in recent years. The Generative Adversarial Network first proposed in the 2014 [6] this deep learning approach focuses on two neural network playing min max game against each other. One responsible for generating new fake images is called generator. The other is called discriminator and it is determined to distinguish between fake and real images.

This approach has been used in 2016 [17] to generate small (64 x 64 pixel) bedroom images from the LSUM dataset [20]. To accomplish that task new DCGAN (Deep Convolutional Generative Adversarial Network) architecture was proposed. For both generator and discriminator architectures are CNN (convolutional neural network) based. All fully connected layers have been swapped with the convolutional layers in case of the discriminator and convolutional-transpose layers in the generator. Furthermore the guideline for stable training was constructed. All those improvements enhanced the stability of the GAN training. The article proved the feasibility of generating non existing human faces by training on scrapped from the internet faces of 64 by 64 pixel size and then generating new ones.

This field of science is constantly evolving and in 2017 many new architectures were proposed. The LSGAN (Least Squares Generative Adversarial Network) [15] generate higher quality images than the DCGAN while testing on the LSUM [20] dataset. Moreover by using least squares loss function the model's generator was able to generate better fake images than by using the sigmoid cross entropy loss function used in literature at that time. The WGAN (Wasserstein Generative Adversarial Network) architecture improved in the 2017 article [7]. The discriminator is re-

placed with the critic network that scores the likeliness of an image being real. The researchers propose using Wasserstein distance based loss function which increases the model stability and coverage, but increasing the training time. Furthermore they insist on eliminating weight clipping as it may lead to the optimisation difficulties. Improving over DCGAN architecture they present generated images on the LSUM [20] dataset.

The BEGAN architecture proposed by Berthelot in 2017 [3] uses autoencoder as a discriminator combining it with Wasserstein distance based loss calculations. This in the researchers own words can simplify training procedure and network size. They use a dataset of 360K images of celebrities to generate believable human faces.

Recently the task of generating anime characters seems to be solved by new and powerful GAN networks. On of them is the StyleGAN [12] proposed by Nvidia which is using style based generator that successfully generates images of human faces. Application of the StyleGAN on the anime characters is presented on "This Waifu Does Not Exist" website [4] where user can generate 1024 x 1024 pixel images of anime girls. Similar website titled "Make Girls Moe" [10] enables a user to generate a anime character with preferences regarding traits such as hair color. More can be read in the scientific report [11].

3. Methodology

The project implements the DGAN architecture [17] designed to generate 64 by 64 images of girl anime portraits. Initial code was heavy inspired by the gray literature articles [1, 8, 9]. Proposed GAN contains a generator and a discriminator which are trained side by side on a preprocessed training images. Project implements such features as training the models with epoch based visualization, loading and saving the models and generating images using trained generator model. The code is done in Python 3.10 with usage of Pytorch [16] library version 1.12 and one GPU Nvidia GeForce RTX 3070 is used as a calculation hardware. This project code is available on Github [14].

3.1. Dataset

The combined dataset of anime girl portraits consist of 100 thousand color images. Most of them were downloaded from the Internet from publicly available Kaggle datasets [2, 5, 18]. The rest (about 36 thousands) has been made available for this project by the Oulu University. All images where resized to the 64 x 64 pixels. Sparingly some augmentation techniques were used to increase the variety of images in the dataset during training such as applied randomly horizontal flips as well as random changes in color contrast, saturation, brightness and hue. Enhanced brightness function that allows to imitate flash light was applied. Samples were also normalized to the (-1, 1) range. Snippet

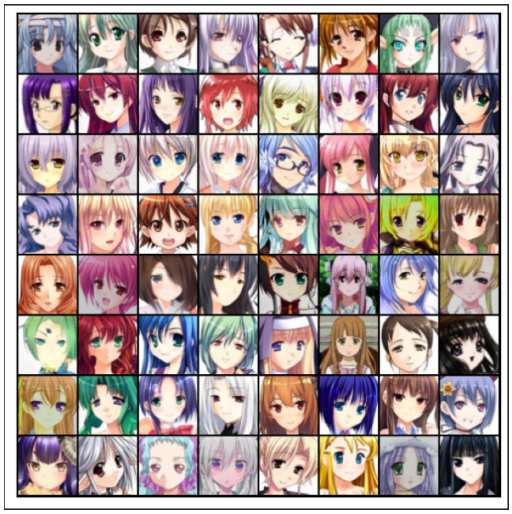
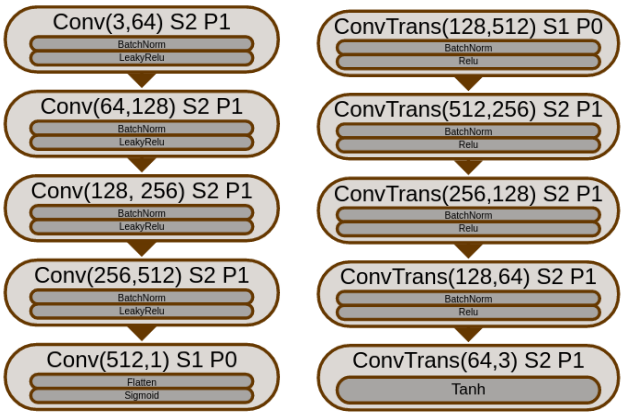


Figure 1. Images from the combined dataset used in GAN training with randomly applied transformations.



(a) Architecture of the Discriminator Model

(b) Architecture of the generator model

Figure 2. This is an system's architecture graph. Both generator Fig. 2b and discriminator Fig. 2a networks structure is visualized. Numbers in parenthesis represent the input and output size of the layers. The 'S' represents stride of the layer and the 'P' represents the padding.

of the combined dataset with randomly applied transformations can be seen in the Fig. 1.

3.2. Architecture

Architecture is heavily inspired by this DCGAN article [17]. In which researchers insist on removing all fully connected layers leaving only convolutional and transposed convolutional layers with pooling layers replaced by batch norm. This can be observed on the Fig. 2, five layered generator and discriminator architecture are used.

Discriminator architecture is presented on a Fig. 2a con-

tains of five convolutional layers with batch norm and LeakyReLU activation function with negative slope parameters set to 0.2. This network is given the 3 x 64 x64 image as an input and it's role is to return the answer weather it recognises the image as generated or as an original one and therefore it return either one or zero.

Where as generator network presented on a Fig. 2b composes of five transposed convolutional layers with batch norm and ReLU activation function. First layer in feeded with random latent vector of size 128. The Last layer returns generated 3 x 64 x 64 color image and uses the hyperbolic tangent activation function.

3.3. Training

Discriminator and generator are trained together, but the loss is calculated for the generator and discriminator independently and the binary cross entropy loss is used. In one epoch first discriminator is trained on real pictures. Real images are loaded from the dataset with batch size 128 and then the generator tries to 'foul' the discriminator by generating believable image which are then classified as either false or real by the discriminator. Discriminators loss is calculated based on the correct classification of both true and generated images.

The next step is to calculate the generators loss. Again the generator creates an fake image and the discriminator determines it's origin this time the loss is calculated with reversed labels. For the back propagation algorithm the Adam optimizer [13] is used with momentum beta1 = 0.5 as in the DCGAN article [17] it was discovered that such value of the beta1 grants better training stability. Moreover to train the networks default learning rate is set as 0.0002 as instructed in the DCGAN article.

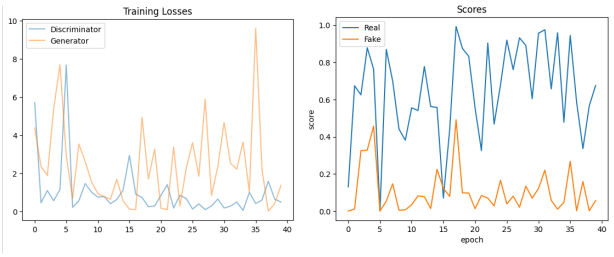
The initialization of the weights of the generator and the discriminator with mean = 0 and std = 0.02 using code implemented in DCGAN Pytorch tutorial [8] proved to have instrumental impact on the quality of the generated images and was applied in this project.

3.4. Improvements

Some techniques diverging from the DCGAN paper [17] architecture are used. They may be considered as various many small improvements proposed by this project. All of them were chosen empirically after multiple try and error experiments.

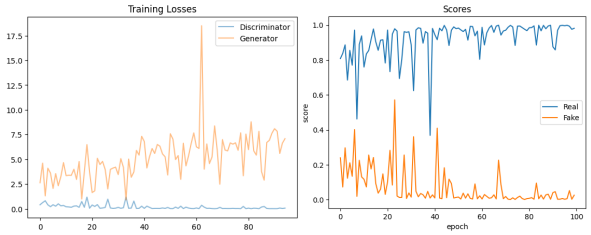
Firstly in the project the aforementioned extension of training data by using images from multiple datasets is conducted. Furthermore the dataset is diversified by using data augmentation.

Next improvement implemented in the project is the discriminator's loss smoothing which is one of the techniques suggested by the 2016 article [19] about possible improvements in GAN networks. The idea is to multiply output of a



(a) Training Loss for experiment 8.4, x - epoch, y - loss value (b) Training discriminator score for experiment 8.4

Figure 3. This Graphs show the training process in the exercise 8.4



(a) Training Loss for experiment 9.7, x - epoch, y - loss value (b) Training discriminator score for experiment 9.7

Figure 4. This Graphs show the training process in the exercise 9.7

loss function for the discriminator on real examples by the factor of 0.9.

Initial learning rate of 0.0002 for both discriminator and generator was modified during the training by usage of the MulitStelLR scheduler that allows to reduce the learning rate by a factor gamma when training process hits certain epoch called milestones. Different values of gamma and the milestones were used in different experiments.

4. Results

About 35 different experiments were conducted during this project. Main goal was to find right hyper parameters for believable image generation. All models were compared subjectively by looking at the 64 generated images from each model. Usually the comparison was conducted after 40 epochs (or about 1 hour) of training, but sometimes the model needed more epochs to catch up. The hyper parameters of the best experiments can be seen on the Tab. 1. As presented in this table most experiments were conducted on the 36000 images dataset acquired form the Oulu University. This is due to the time consuming nature of training model on bigger datasets. Most commonly used architecture was DCGAN architecture [17] (called "simple") as the results were satisfactionary and bigger models not al-

ID	Dataset	Augment	Architecture	Epochs	Learning Rate	Smooth	Scheduler gamma
8.3	Oulu	1.0*	simple	40	0.0002	0.9	No
8.8	Oulu	0.4	extended	40	0.0002	0.9	No
8.4	Oulu	No	simple	40	0.0002	No	No
9.4	full	0.2	simple	80	g=0.0003 d=0.0001	0.9	No
9.5	Oulu	No	simple	80	g=0.0003 d=0.0001	0.9	No
9.6	Oulu	0.2	buffed	55	g=0.00025 d=0.0001	0.9	No
9.7	full	0.2	simple	100	g=0.00025 d=0.0001	0.85	0.75
9.8	full	0.4	simple	40	g=0.0005 d=0.0001	0.8	0.75
10.0	Oulu	0.2	wgan	100	0.0002	0.8	0.75

Table 1. Here are the hyper parameters of the best experiments. ID column represents the internal naming convention of the experiments. Dataset column shows the dataset used. Augment column presents the probability of augmentations. Epoch column shows the number of epochs of training. Learning Rate is either presented for both generator and discriminator or for each of them labeled as 'g' and 'd'. * In experiment 8.3 additional augmentation "RandomEqualize" was applied.

ways guaranteed better results. The generated images generated by the network trained in those experiments can be seen in the Fig. 5 where all 9 best experiment results are shown. Loss and score calculation graphs (such as the one on the Fig. 3 for the exercise 8.4) were instrumental tool to spot broken training patterns as models sometimes tended to collapse with training loss of the generator dropping to zero and the loss of the discriminator skyrocketing therefore sometimes it was advantageous to stop the training reload model from the checkpoint and star again.

All experiments were unique in the sense that all tried to test different combinations of the parameters. The experiment 8.3 as seen in the Fig. 5a established that the augmentation techniques should be used less boldly and with more caution. The Random Equalisation augmentation technique was dropped due to personal taste as it made the pictures look less "cute". The experiment 8.4 is the one conducted on the DCGAN architecture [17] without any major changes. The results shown on the Fig. 5b after 40 epochs shows that this architecture with little Oulu dataset is tailored to this problem. On the other hand one can see slight blurry effect present on images. The loss history can be seen in the Fig. 3a as one can see the loss does not seen to stabilize and tend to jump in value especially the generators loss. The similar story presents itself in the Fig. 3b discriminator scores history chart although overall the real and fake pictures are correctly classified the result can varies between epochs. In the exercise 9.6 the "buffed" architecture with more filters in convolutional layers is used. Besides the increase of the time needed for the training the results shown in the Fig. 5f do not differ form the 8.4 results. Similar results one can experiences in the Fig. 5c with usage of the "extended" model with more filters and 6 layered discriminator. This arguably proves that even though the results from the more complicated network may be more saturated they have not improved over the "bare" 8.4 network.

More experiments were conducted using the "simple" model architecture testing different training strategies. The difference between the results in the experiments 9.5 conducted on the Oulu data and the 9.4 on the full dataset, can be seen on the Fig. 5e (9.5) and the Fig. 5d (9.4). The images generated in the experiment 9.4 are arguably more saturated and more colorful then those generated in 9.5. This is a positive result of having more images in the data set. The experiment 9.7 tries to perform longer training of 100 epochs. The results shown on the Fig. 5g are as good as (depending on a taste) as all others. The loss and score charts presented on the Fig. 4 suggest that the generators loss was increasing and further training wouldn't improve the generated images. In the 9.8 experiment the scheduler is used to reduced initially high learning rate. This initial bump in training speed improves brightness of the images.

Last experiment was conducted on the WGAN architecture [7]. The code was implemented accordingly to the Github code found in [21]. The results are similar to the other experiments as can be seen on the Fig. 5i, but the model learns much more slowly. As the result on the epoch 40 or 80 were much more distorted.

Overall judging by "short" training time up to 100 epochs the increased model complexity or increased amount of the convolutional filters haven't subjectively improved the generated images. Comparable results can be produced by using basic "simple" DCGAN model [17] with minimal dataset. No model could get rid off the badly generated images as there were some in all generated samples. Increasing the epochs can improve the results but the risk of the model collapsing increases. Some augmentation techniques can be harmful for the end effect as it is example of the experiment 8.3 Fig. 5a but others were beneficiary increasing image brightness and reducing blurry effect. Changing learning rate up had big initial importance on first couple of epochs making it easier to generate better pictures. Decreas-

ing it after a while had reportedly better effect on model focusing on details of the image rather than changing it all. Although it was never truly proven weather this is beneficial in the long run.

5. Conclusion

Training GAN network is very nuanced task. Slight changes to the hyper parameters could have astounding effect on the results or be completely unimportant. The training can collapse without accomplishing the task. Moreover judging the models performance is a non trivial task that requires some level of the subjective assessment by a human judge. This project shown that the believable anime character images can be generated with simple network and quick training time of about one hour with various techniques.

6. Acknowledgments

This project has been tasked by the University of Oulu by the Faculty of Information Technology and Electrical Engineering. It is a part of the Deep Learning course hence the template code and the initial dataset had been prepared by the teaching staff. Therefore I credit Li Liu, Shuzhou Sun, Huali Xu, Jiehua Zhang and Zhuo Su.

References

[1] Afnan Amin Ali. Gan (generative adversarial network). https://medium.com/swlh/gan-generative-adversarial-network-3706ebfef77e, 2020. Accessed: 2023-01-18. 2

[2] Andy8744. Re:zero rem anime faces for gan training, 2022. 2

[3] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks, 2017. 2

[4] Gwern Branwen. Thiswaifudoesnotexist.net. https://www.thiswaifudoesnotexist.net/, 2019. Accessed: 2023-01-18. 1, 2

[5] Spencer Churchill and Brian Chao. Anime face dataset, 2019. 2

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017. 1, 4

[8] Nathan Inkawhich. Dcgan tutorial. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html. Accessed: 2023-01-18. 2, 3

[9] Nathan Inkawhich. Lesson 6: Generative adversarial networks and transfer learning. https://jovian.com/learn/deep-learning-with-pytorch-zero-to-gans/lesson/lesson-6-image-generation-using-gans. Accessed: 2023-01-18. 2

[10] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Make girls moe. https://make.girls.moe, 2017. Accessed: 2023-01-18. 1, 2

[11] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks, 2017. 2

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018. 2

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 3

[14] Aleksander Madajczak. Animerush. https://github.com/theATM/AnimeRush. Accessed: 2023-01-20. 2

[15] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2016. 1

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 2

[17] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. 1, 2, 3, 4

[18] Arnaud ROUGETET. selfie2anime, 2019. 2

[19] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. 3

[20] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop, 2015. 1, 2

[21] Zeleni9, Jeremy Fix, CharlesLiu7, and Hsuan-Wei Fan. pytorch-wgan. https://github.com/Zeleni9/pytorch-wgan/tree/master/. Accessed: 2023-01-20. 4

(a) Exp 8.3 (b) Exp 8.4 (c) Exp 8.8

(d) Exp 9.4 (e) Exp 9.5 (f) Exp 9.6

(g) Exp 9.7 (h) Exp 9.8 (i) Exp 10.0

Figure 5. Experimental Result of the experiments, generated images by the generator at the end of the training.