Automatic Anime Characters Creation with Generative Adversarial Networks

Anirban Saha Anik (id: 18-36207-1)[a], Bishowjit Datta (id: 18-37372-1)[a], Tonmoy (id: 18-37390-1)[a], Md. Ariful Islam (id: 18-37734-1)[a]

[a] *Department of Computer Sciences, American International University-Bangladesh*

Abstract

In this paper we investigate the training of GAN models specialized on an anime facial image dataset. We address the issue from both a data and a model perspective. We show that our efforts result in a stable and high-quality model through quantitative analysis and case studies. We discuss Generative Adversarial Network approach more details. We also train the discriminator for a few epochs, then train the generator for a few epochs, and repeat. This way both the generator and the discriminator get better at doing their jobs.

Keywords: Generative Adversarial Network (GAN), Generator, Discriminator

1. Introduction

Deep neural networks are mainly used for supervised learning (classification or regression). Generative Adversarial Networks or GAN use neural networks for a very specific purpose: Generative modelling that automatically discovering and learning the regularities or patterns in input data that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. If we gives a human face dataset then we can train a model which can learn to generate new human face.Goodfellow et al. [2014] We have basically a Generator and a Discriminator neural networks here. Normally so far we have seen that there has only been one model and we give an input to the model and it gives a prediction. We compare the predictions with the targets and then we perform gradient descent and that trains the model and then we can use the model for making inferences. That is typically how a classification or regression model works. Now, a generative adversarial network has two different models, so two different neural networks. There is a generator network and there is a discriminator network. Now the generator network input to it some random input vector. So we feed in a random vector into the generator network and it generates an image or it at least tries to generate an image. And this is the model that we really want to train. We want to make this model better and better at generating images which look like they may have been drawn from a particular dataset which consists of over 63,000 cropped anime faces. Here generative modelling is an unsupervised learning task, so the images do not have any labels. And to do that we use a second network called a discriminator, so the discriminator is a model. It's only job is to be able to differentiate between real images and generated image which is generate by generator. So, the discriminator is responsible for taking real images and generated images and discriminating between them, telling which one is which.Mirza and Osindero [2014]

2. Literature Review

The authors present a method for stabilizing Generative Adversarial Networks (GANs) by specifying the generator objective in terms of an unrolled discriminator optimization. They demonstrate how this technique overcomes the problem of mode collapse, stabilizes GAN training with complicated recurrent generators, and boosts the generator's variety and coverage of the data distribution.[Liu and Tuzel [2016]]

In this review article, the authors provide an overview of GANs for the signal processing community, drawing on familiar analogies and concepts where possible. In addition to describing several approaches for training and generating GANs, we also discuss the theory and implementation of GANs. [Creswell et al. [2018]]

Goodfellow et al.[Li et al. [2017]] proposed Generative Adversarial Networks (GANs), who explained the theory of GANs learning using a game-theoretic scenario. GANs have been applied to various specialized tasks, such as picture generation, image super-resolution, text to image synthesis, and image to image translation, demonstrating their remarkable capabilities for unsupervised tasks.

The discriminator is hypothesized as a classifier with the sigmoid cross entropy loss function in regular GANs. However, they discovered that using this loss function throughout the learning process can result in vanishing gradients. To address this issue, the author proposes Least Squares Generative Adversarial Networks (LSGANs) in this research, which use the least squares loss function as the discriminator. They show that decreasing the LSGAN objective function reduces the Pearson X^2 *divergence*.[*Metzet al.* [2016]]

3. Proposed Method

3.1. ***GAN***

Generative adversarial networks (GANs) are algorithmic architectures that use two neural networks, pitting one against the other (thus the "adversarial") in order to generate new, synthetic instances of data that can pass for real data. They are used widely in image generation, video generation and voice generation

3.1.1. ***Discriminator Network***

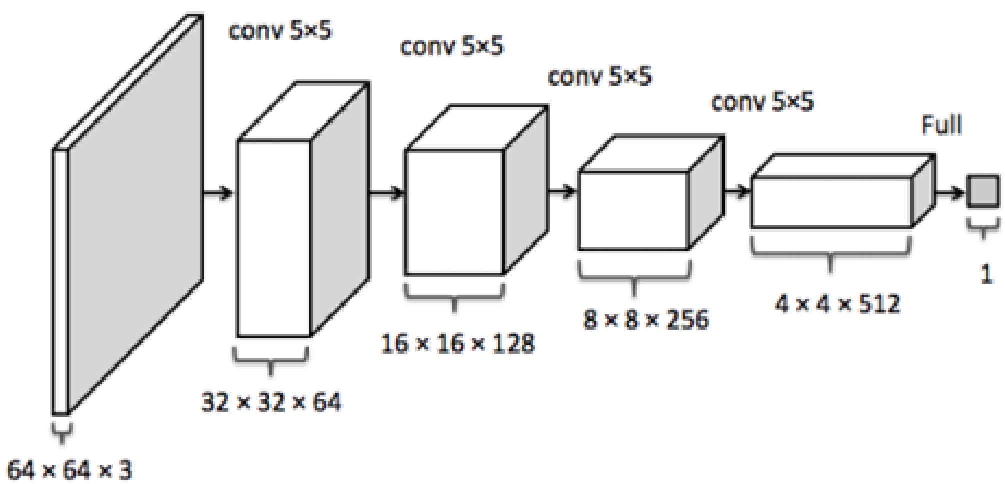

Figure 1: Discriminator Network

The discriminator takes an image as input, and tries to classify it as "real" or "generated". In this sense, it's like any other neural network. We'll use a convolutional neural networks (CNN) which outputs a single number output for every image. We'll use stride of 2 to progressively reduce the size of the output feature map. we're using the Leaky ReLU activation for the discriminator.

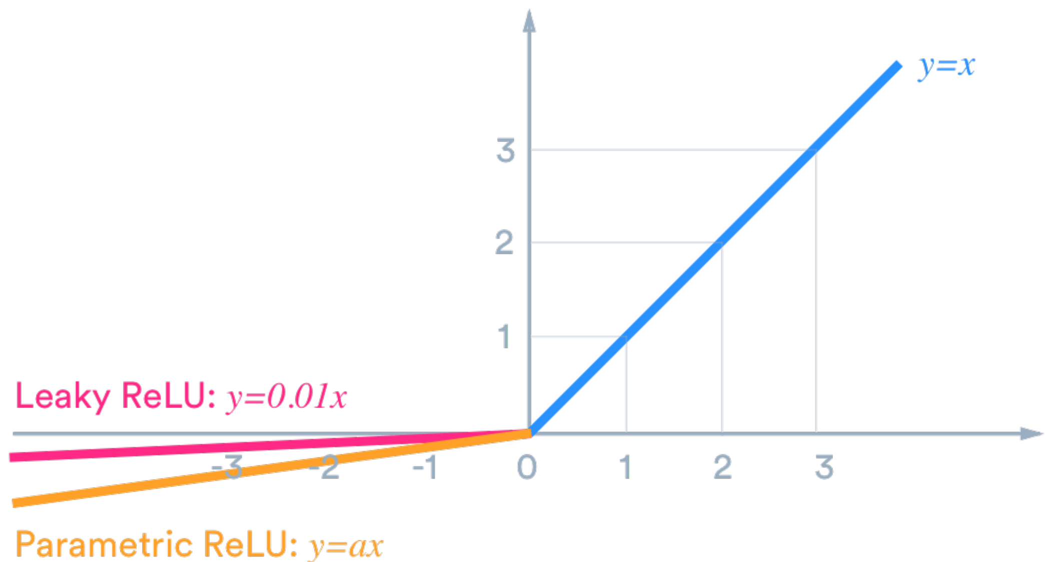

Figure 2: Leaky ReLU

Different from the regular ReLU function, Leaky ReLU allows the pass of a small gradient signal for negative values. As a result, it makes the gradients from the discriminator flows stronger into the generator. Instead of passing a gradient (slope) of 0 in the back-prop pass, it passes a small negative gradient.

3.1.2. ***Generator Network***

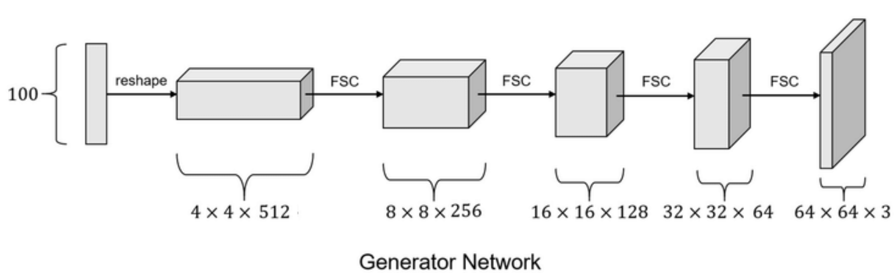

Figure 3: Generator Network

The input to the generator is typically a vector or a matrix of random numbers (referred to as a latent tensor) which is used as a seed for generating an image. The generator will convert a latent tensor of shape (128, 1, 1) into an image tensor of shape 3 x 28 x 28. To achive this, we'll use the ConvTranspose2d layer from PyTorch, which is performs to as a transposed convolution. We use the TanH activation function for the output layer of the generator.

3.1.3. *Discriminator Training*

Since the discriminator is a binary classification model, we can use the binary cross entropy loss function to quantify how well it is able to differentiate between real and generated images.

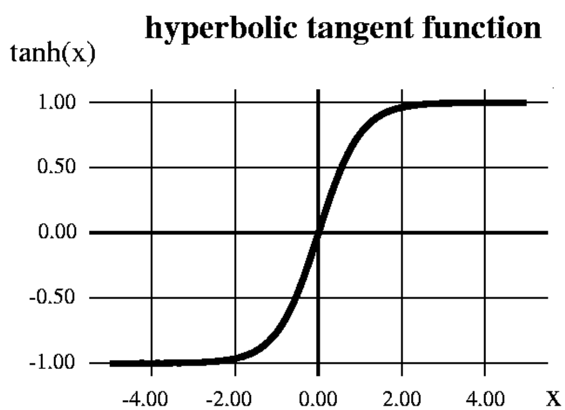

Figure 4: Tangent Function

"The ReLU activation (Nair and Hinton [2010]) is used in the generator with the exception of the output layer which uses the Tanh function. We observed that using a bounded activation allowed the model to learn more quickly to saturate and cover the color space of the training distribution. Within the discriminator we found the leaky rectified activation (Xu et al. [2015])to work well, especially for higher resolution modeling."

Here are the steps involved in training the discriminator.

- We expect the discriminator to output 1 if the image was picked from the real Anime Face Dataset dataset, and 0 if it was generated using the generator network.

- We first pass a batch of real images, and compute the loss, setting the target labels to 1.
- Then we pass a batch of fake images (generated using the generator) pass them into the discriminator, and compute the loss, setting the target labels to 0.
- Finally, we add the two losses and use the overall loss to perform gradient descent to adjust the weights of the discriminator.

It's important to note that we don't change the weights of the generator model while training the discriminator (opt_d only affects the discriminator.parameters())

3.1.4. ***Generator Training***

Since the outputs of the generator are images, it's not obvious how we can train the generator. This is where we employ a rather elegant trick, which is to use the discriminator as a part of the loss function. Here's how it works:

- We generate a batch of images using the generator, pass the into the discriminator.
- We calculate the loss by setting the target labels to 1 i.e. real. We do this because the generator's objective is to "fool" the discriminator.
- We use the loss to perform gradient descent i.e. change the weights of the generator, so it gets better at generating real-like images to "fool" the discriminator.

3.1.5. ***Full Training Loop***

Let's define a fit function to train the discriminator and generator in tandem for each batch of training data. We'll use the Adam optimizer with some custom parameters (betas) that are known to work well for GANs. We will also save some sample generated images at regular intervals for inspection.

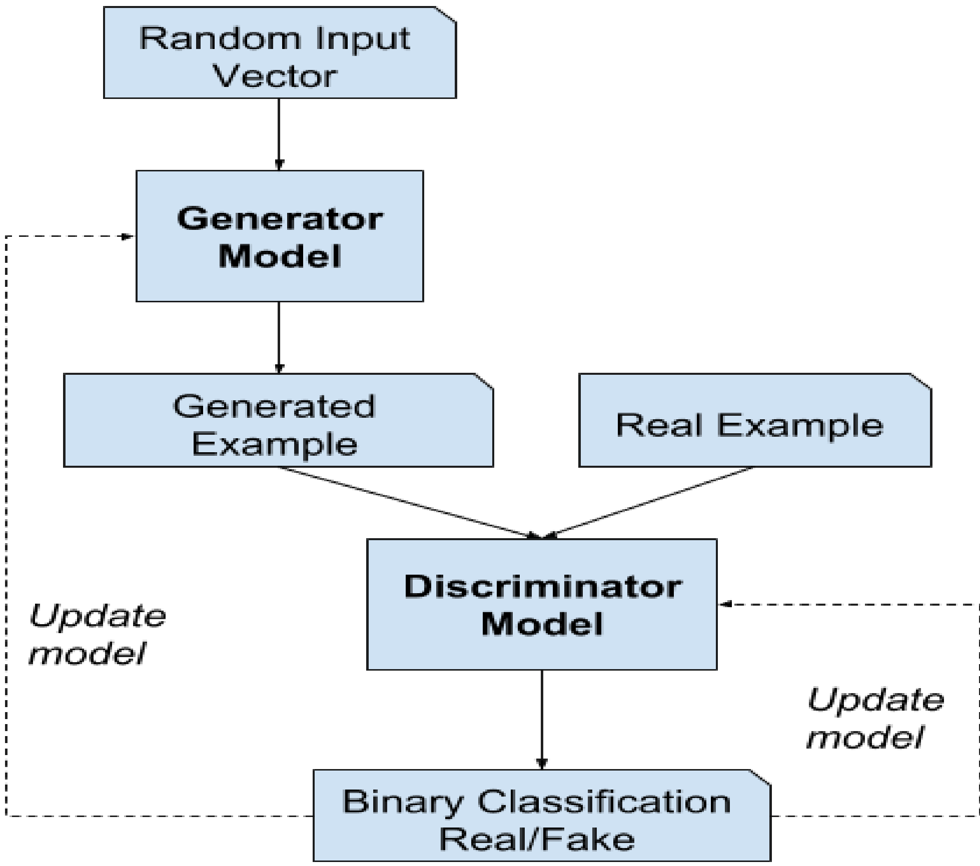

Figure 5: GAN Training Loop

4. Results

Epoch	Generator Loss	Discriminator Loss	Real Score	Fake Score
25	4.9199	0.0623	0.9683	0.0269
20	4.1046	0.1163	0.9609	0.0673
15	3.9303	0.3790	0.8527	0.1580
10	5.7568	0.2232	0.9855	0.1575
5	5.8114	0.3049	0.9141	0.1636
1	3.3059	0.7899	0.5771	0.0475

We give the model 63,632 anime face images dataset. Here's a image from the data set

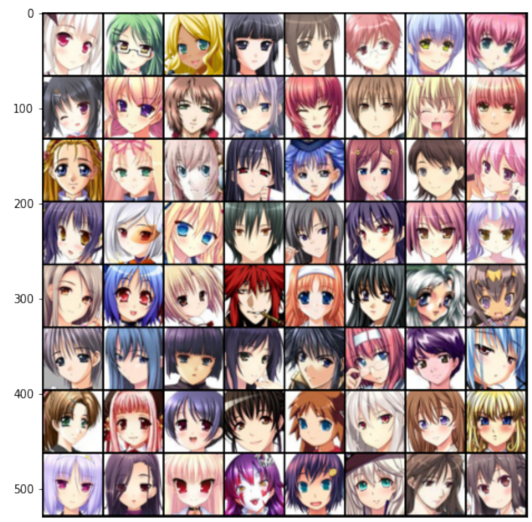

Figure 6: Train Data Image

Then we use the generator to generate fake image. Here's a fake image

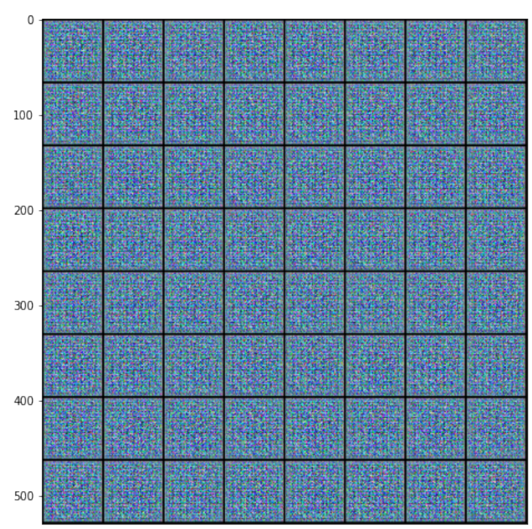

Figure 7: Generated Fake Image

After training the model for 25 epochs we get the final image shown bellow

Figure 8: Generated Final Image

From figure 8 we can we the generated image is closer the real dataset. The Discriminator and Generator loss and the Real and Fake Score graph is given bellow

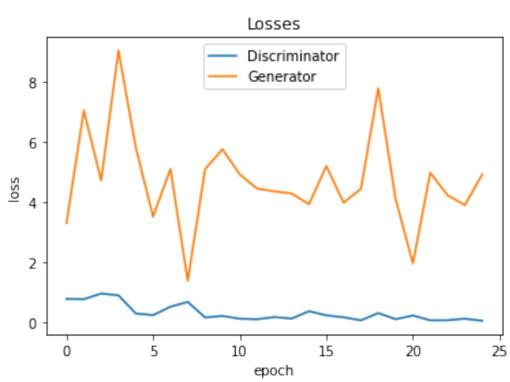

Figure 9: Discriminator and Generator loss

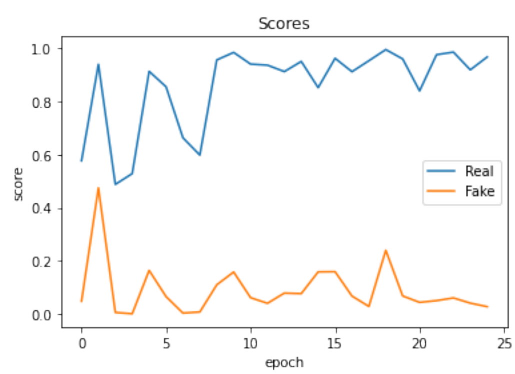

Figure 10: Real and fake score

5. Discussion

The main content of this article will present how discovering and learning the patterns in input data that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.

6. Conclusion

In this paper, we look into the automatic production of anime characters. We successfully built a model that can create realistic facial images of anime characters by combining a clean dataset and multiple viable GAN training methodologies. When class labels in the training data are not evenly distributed, one path to go is to improve the GAN model. FID only gives measurement when the prior distribution of sampled labels equals the empirical labels distribution in the training dataset, hence quantitative assessing methods should be investigated in this circumstance.

References

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. Generative adversarial nets. Advances in neural information processing systems 2014;27.

Mirza, M., Osindero, S.. Conditional generative adversarial nets. arXiv preprint arXiv:14111784 2014;.

Liu, M.Y., Tuzel, O.. Coupled generative adversarial networks. Advances in neural information processing systems 2016;29:469–477.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A.A.. Generative adversarial networks: An overview. IEEE Signal Processing Magazine 2018;35(1):53–65.

Li, C., Xu, K., Zhu, J., Zhang, B.. Triple generative adversarial nets. arXiv preprint arXiv:170302291 2017;.

Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J.. Unrolled generative adversarial networks. arXiv preprint arXiv:161102163 2016;.

Nair, V., Hinton, G.E.. Rectified linear units improve restricted boltzmann machines. In: Icml. 2010,.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., et al. Show, attend and tell: Neural image caption generation with visual attention. In: International conference on machine learning. PMLR; 2015, p. 2048–2057.