An evaluation of Art Synthesis on modern GAN models

Kah Han Chia

Bachelor of Science in Computer Science and Artificial Intelligence
The University of Bath
2022/2023 Summer Semester

An evaluation of Art Synthesis on modern GAN models

Submitted by: Kah Han Chia

Copyright

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Abstract

Artificial Intelligence (AI) image synthesis is one of the most influential topics in recent years, and Generative Adversarial Network (GAN) is undoubtedly one of the influential contributors to the progression of image synthesis. While there are many different variations of the GAN model that can synthesize high-quality images with very different components, the overall architecture still stays true to the GAN model of having a Generator (G) and a Discriminator (D). However, existing GAN image synthesis models that produces images from a noise image have neglected Art synthesis as the amount of literature about GAN models that step into the field of art generation is very limited in terms of quantity. This paper will delve into the differences between real life such as pictures of humans, birds, and cars, and art images and utilize a modern image synthesis model, StyleGAN2-ADA PyTorch version, to evaluate the generated art images.

Contents

1 Introduction **1**

2 Background **3**
2.1 Preliminaries . . . 3
2.1.1 Generative Adversarial Network(GAN) . . . 3
2.1.2 Progressively Growing Generative Adversarial Network(ProGAN) . . . 4
2.2 Related Work . . . 5
2.2.1 ArtGAN . . . 6
2.2.2 ArtGAN-AEM . . . 7
2.2.3 StyleGAN . . . 7

3 Method **9**
3.1 Dataset . . . 9
3.2 Selected Model . . . 11
3.2.1 AdaIN Operation . . . 11
3.2.2 StyleGAN2 Modification . . . 12
3.2.3 Adaptive Discriminator Augmentation(ADA) . . . 12
3.2.4 Reason for selected model . . . 13
3.2.5 Method of evaluation . . . 13

4 Experiment **14**
4.1 Challenges . . . 14
4.2 Experiment Settings . . . 14
4.3 Results and Evaluation . . . 15

5 Conclusions and Future Work **21**
5.1 Challenges . . . 21
5.2 Conclusion . . . 21
5.3 Future Work . . . 22

Bibliography **23**

A Code **25**
A.1 File: Downsize.py . . . 25
A.2 File: L2.py . . . 25

List of Figures

2.1 GAN model architecture from (Wang et al., 2017) . . . 3
2.2 ProGAN model architecture from (Karras et al., 2017) . . . 4
2.3 ProGAN skip connection example from (Karras et al., 2017) . . . 5
2.4 ArtGAN model architecture from (Tan et al., 2017) . . . 6
2.5 DCGAN(Radford, Metz and Chintala, 2015)(top), GAN/VAE(Doersch, 2016)(middle), and ARTGAN (bottom) by (Tan et al., 2017) . . . 6
2.6 ArtGAN-AEM model architecture by (Tan et al., 2018) . . . 7
2.7 StyleGAN model architecture from (Karras, Laine and Aila, 2019) . . . 8
2.8 Water droplet examples from (Karras et al., 2019) . . . 8

3.1 Nude art images from (Wikiart, 2019) . . . 10
3.2 Ukiyo-e art images from (Pinkney, 2020) . . . 10
3.3 The comparison between StyleGAN architecture and StyleGAN2 by (Karras et al., 2019) . . . 11
3.4 Augmentation process of StyleGAN2-ADA by (Karras et al., 2019) . . . 12

4.1 Generators and Discriminators Layers . . . 15
4.2 Ukiyo-e Face results produced by our trained Model . . . 16
4.3 Augmentation Leakage . . . 16
4.4 Nude painting results produced by our trained Model . . . 17
4.5 Bad (Left) and Good (Right) results produced by our model . . . 18
4.6 Example of our model attempt to generate multiple individuals . . . 18
4.7 Example of our model attempt to generate images of the subject lying down 18
4.8 Results generated with our trained model: Images in red line is the real image, the 3 image surrounding the real image with red line from the right are the nearest neighbour generated image of the real image. . . . 19
4.9 Image from ArtGAN-AEM for nude painting, image in the between the red line is the real image and the images to right are the generated results. . . . 20

List of Tables

4.1 FID of the image generated by the selected model 15
4.2 L2 Distance measurements between real and generated images, Real image 1 is the most left image with a red box surrounding it. 19

Acknowledgements

I would like to express my deepest gratitude to my supervisors, HongPing Cai and Chen DA, for providing their guidance and support throughout my research journey.

Chapter 1

Introduction

Art has been a more significant asset to humans than many expect. It was a creative medium for humans to express emotions, convey a message, tell a story, and many more. Modern image synthesis models mainly focus on generating facial. They are typically evaluated using real-life objects like birds and cars, but art is in a lot of ways different from real-life objects. The objective of this project is to analyse modern image synthesis Generative Adversarial Network (GAN)(Goodfellow et al., 2020) models, For this particular literature, we will test and evaluate StyleGAN2-ADA (Karras et al., 2020a) to generate art images.

GAN is one of the most influential works in the field of computer vision and AI art generation. However, there are other applications that utilise GAN as well, including Image-to-image translation (Isola et al., 2016; Zhu et al., 2017), Super-resolution (Ledig et al., 2017) and image synthesis. Note that there is a clear distinction between Image-to-Image translation, super-resolution and image synthesis. Image synthesis outputs images from a noise image, which is different from the other two applications that output images from a non-noise image. As one of the more modern models, StyleGAN2-ADA PyTorch has proven its ability to generate high-quality and diverse images. By evaluating the performance and results, we can observe the improvement made by the current state of image synthesis models in comparison to the older model and discuss potential applications in the art world.

The book "Ways of Seeing"(Berger, 1972) explores how art is relatively unique to other image representations. According to the author, art requires engagement from the observer, and that meaning can be found by just observing the art. A more recent literature(Hertzmann, 2020) get into insights of visual indeterminacy which a term originally introduced by another literature (Pepperell, 2006) in both real-life images and art. Visual indeterminacy refers to an object or image that originally seemed to be realistic and logical, but by closer inspection, flaws in the image will start to appear. The image will appear more confusing, and the sense of depth and space will become blurry, causing the relationships between elements in the image to become more illogical. The literature goes into depth about how visual indeterminacy naturally exist in some art form and some GAN models tend to generate visual indeterminacy naturally. Therefore art that naturally inherits the indeterminacy attribute is said to be easily produced by GAN models but other forms of art that inherit less visual indeterminacy can introduce some difficulties. However, not all flows in a generated image are considered indeterminacy. Properties such as blurriness lean more towards the direction of ambiguity. The literature also discusses that the concept of visual indeterminacy does not only exist in the field of GAN image synthesis but any other existing image synthesis models

There are some challenges when it comes to generating art that typical real-life images do not suffer from. The development of art requires attributes beyond technical skills; it also requires creativity and expression. Art varies depending on individual personal preference and can sometimes be illogical. Another challenge when it comes to art generation is the difficulty of dataset collection. Given an art dataset that consists of only paintings, many of these paintings would have been painted by different individuals. Different from real-life images, different painters will have different preferences when it comes to their paintings. The colour tone, level of detail and realism may differ by a large margin. On the other hand, if we were to construct an art dataset filtered by artist names. It will not be easy to find an artist that has created over 10,000 works, which was the recommended size(Karras et al., 2020a) of the dataset to train a GAN model. Therefore, alternative models such as our selected model need to be utilised in order to train and produce high-quality art images. In contrast, the older models struggle due to the lack of a consistent dataset.

Chapter 2

Background

This section will explain the preliminaries necessary to understand the image synthesis model used in this literature. We will introduce basic concepts of GANs and progressively growing GANs (ProGAN) (Karras et al., 2017), as well as related work in the field of art image synthesis. Having an understanding of related work will provide insight into the performance of previous models, which can be used to compare with our selected model. This will help identify the direction of progression for GAN-based art image synthesis.

2.1 Preliminaries

2.1.1 Generative Adversarial Network(GAN)

The GAN(Goodfellow et al., 2020) model mainly consists of a Generator(G) and a Discriminator(D). In the case of image synthesis, the discriminator will receive both real images and generated images as input, and the job of a discriminator is to distinguish between the two of the images. The job of the generator is to generate fake images from random noise images for the discriminator and try to convince the discriminator that the generated image is real. As shown in figure 2.1 below.

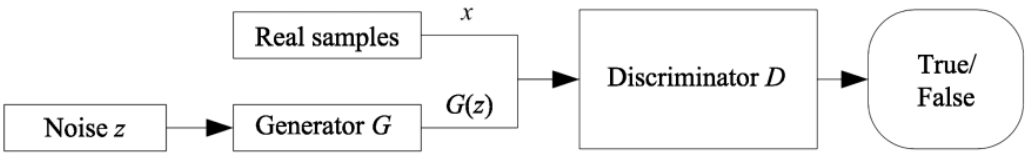

Figure 2.1: GAN model architecture from (Wang et al., 2017)

The generator improves its ability to generate convincing images by changing its parameter according to the feedback provided by the discriminator. The two neural networks will train each other adversarial in a zero-sum game-like situation; the generator will get better at generating images similar to the training set, and the discriminator will increase its strictness in differentiating real from fake. The main value function of the GAN model is given by the equation 2.1 below.

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

The objective of the Generator is to minimise the value function by maximising $D(G(z))$ in the second half of the 2.1 while the discriminator tries to maximise the value function by minimising $logD(x)$ and $log(1 - D(G(z)))$ hence the zero-sum game. For the base GAN model, the Generator will take a point from the latent space to generate an output image. Latent space provides a constructed representation of the Generator's output space. Think of the latent space as a 2D grid. By selecting a point in the latent space, the Generator will output an image. Points that are closer to each other will result in more similar images, while points that are further away will have more distinct features and appearances. However, there are a few problems regarding GAN models (Weng, 2019; Salimans et al., 2016):

- **Mode Collapse**: Where the Generator successfully produces a single or a small set of output that manages to fool the discriminator, and it tries to learn and produce only that output. It fails to learn and represent the complex training set data distribution and gets stuck in a small space with extremely low variety.
- **Vanishing gradients**: A well-trained discriminator becomes better at recognising fake images causing the change of loss function changes with respect to the discriminator's parameters to approach zero. This is a problem as the Generator will become ineffective when learning from the discriminator feedback. The process of learning will then slow down or stop. But on the other hand, if the discriminator is not able to improve to a sufficient level, then the feedback it provides to the Generator will be useless.
- **Dataset requirements**: Large datasets are required to improve the variation and quality of generated images. Training a GAN model with a small dataset introduces the risk of overfitting the generated images.
- **Training instability**: Large and complex architecture required to generate high-resolution images causes instability in the GAN model, making it difficult for the discriminator to distinguish between authentic and generated images effectively.

2.1.2 Progressively Growing Generative Adversarial Network(ProGAN)

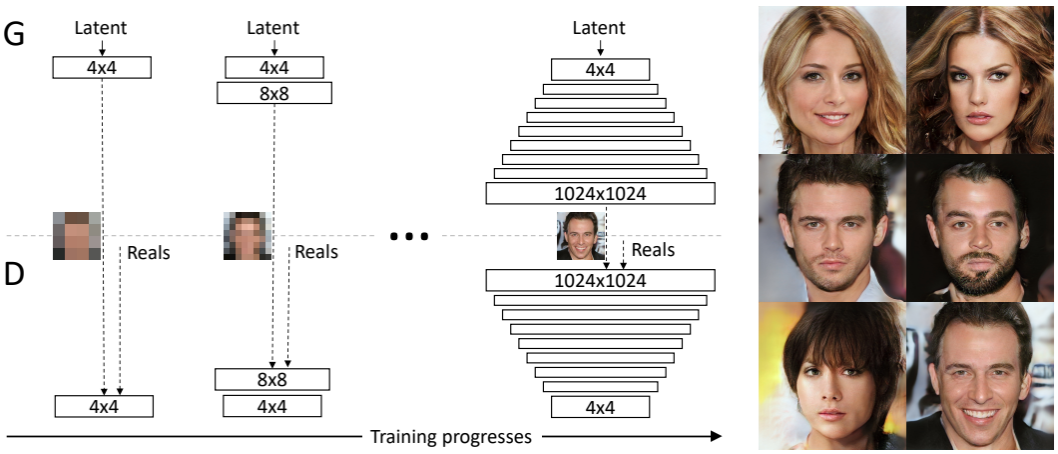

Figure 2.2: ProGAN model architecture from (Karras et al., 2017)

Progressive GAN (Karras et al., 2017) is an extension of the GAN network. The model trains images incrementally from a spatial resolution of 4x4 to a desired image size. The incremental training is done by adding an additional block of layers for each spatial resolution increase. For

each additional block of layers added, the spatial resolution doubles, as shown in Figure 2.2. This enables the model to learn the larger scale characteristics of the image, such as the overall composition, colours, and shape of the object first. Then as the spatial resolution increases over time, the model will start to learn to generate small-scale details of the images, such as the eyes, nose, and lips of a face. This progressive training allows the model to produce much higher-quality images. The overall architecture still stays true to the original GAN model in the sense that there is a Generator and Discriminator.

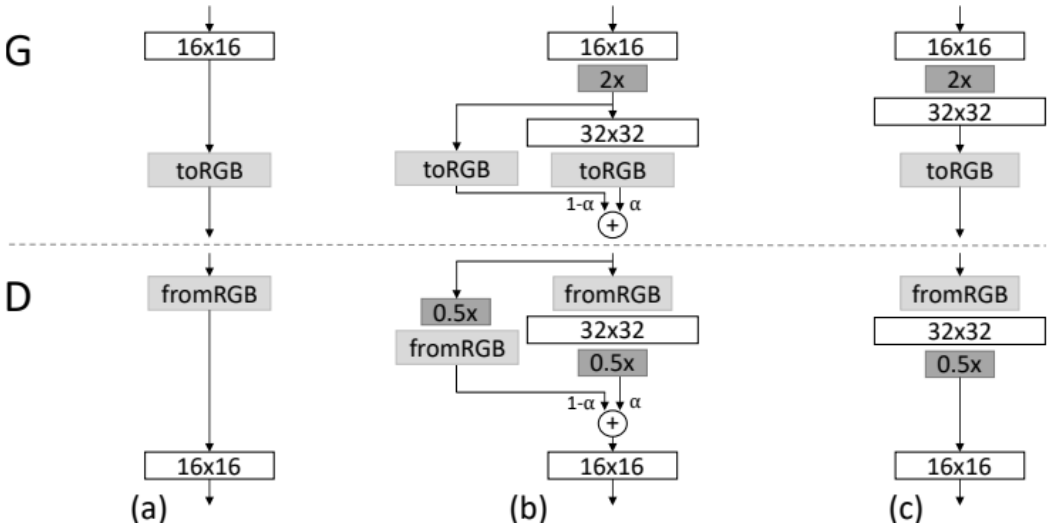

Figure 2.3: ProGAN skip connection example from (Karras et al., 2017)

The output of the current layer will be faded into the newly constructed layers instead of being added directly. This is achieved by using a skip connection, as shown in Figure 2.3. For the generator, the skip connection connects the new block to the output of the generator and adds it to the current output layer with a weighting alpha α. α starts at a small number and slowly increases to 1 over training iterations. The discriminator, on the other hand, connects the new block to the input of the current layer with the input of the discriminator with α. Incremental training also allows the training process to be faster and allows the training process of the model to be more stable. ProGAN also implemented Pixelwise Feature Vector Normalisation (Pixel Norm) in the generator instead of batch normalisation, which was used in the original GAN model, to reduce the risk of mode collapse without causing too much change in the output of the generator.

2.2 Related Work

This section will review prior studies that investigated Art Image synthesis models capable of generating images from noise images. These studies will provide illustrations of how art synthesis models were developed and evaluated. Additionally, the outcomes obtained from these studies will be used as a reference point to compare our results.

2.2.1 ArtGAN

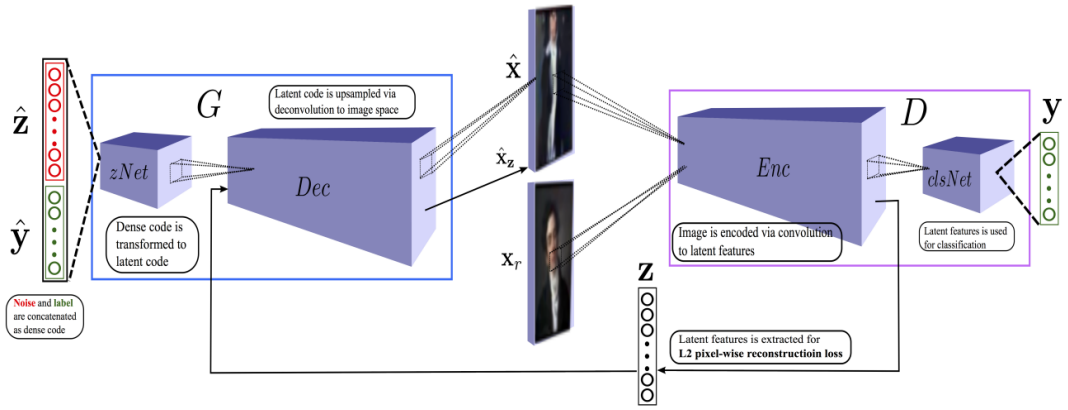

Figure 2.4: ArtGAN model architecture from (Tan et al., 2017)

ArtGAN (Tan et al., 2017) is considered an early GAN model that explored the field of art image synthesis and has strongly influenced current literature. It is an extension of the GAN model where the generator is fed with an additional randomly selected input vector, $\hat{y}$, and a noise vector, $\hat{Z}$, as illustrated in Figure 2.4.

The $\hat{y}$ vector will contain extra useful information that will contribute to generating higher-quality images, such as the attributes and classes of images. This feeding of $\hat{y}$ vector is an attempt to mimic the learning process of developing a specific type of skill at a time, where $\hat{y}$ in this context represents the type of skill being developed. Additionally, the generated images inputted into the Discriminator are labelled based on the $\hat{y}$ vector, and cross-entropy is used to provide feedback to the generator. This improves the quality of the feedback for the Generator network, resulting in the ability to synthesize better-quality images.

Another noteworthy advantage of using the extra input label is that it enabled the model processing time to be lower in comparison to the original GAN model, which is beneficial for time efficiency. However, ArtGAN still requires a large dataset to produce good results.

Figure 2.5: DCGAN(Radford, Metz and Chintala, 2015)(top), GAN/VAE(Doersch, 2016)(middle), and ARTGAN (bottom) by (Tan et al., 2017)

Figure 2.5 above showcases the overall improvement of ArtGAN in comparison to DCGAN(Radford, Metz and Chintala, 2015), Variational Autoencoders VAE(Doersch, 2016), and GAN. Please refer to the original literature for a better clarity of the image.

2.2.2 ArtGAN-AEM

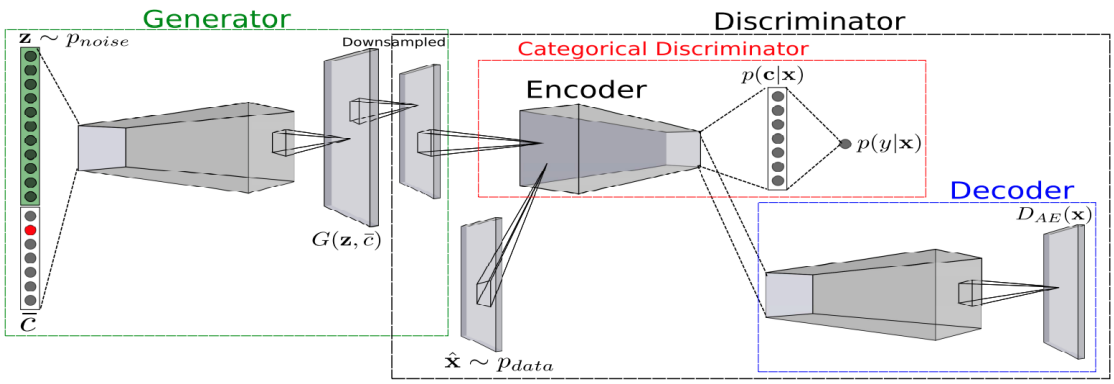

Figure 2.6: ArtGAN-AEM model architecture by (Tan et al., 2018)

ArtGAN-AEM (Tan et al., 2018) is a modified version of ArtGAN. The model includes an autoencoder-based discriminator (Berthelot, Schumm and Metz, 2017) with the IQ strategy, which trains the generator with higher-resolution images and downsizes them using average pooling to select an average-ranked section of the image. The selection process is also referred to as a voting scheme, as the average pooling will vote for the section with better quality images. Figure 2.6 illustrates the model's architecture. It's worth noting that ArtGAN-AEM is not the only variant of ArtGAN and that different experiments may have used different variants.

AEM, also known as Adversarial Example Minimization, is an operation that aims to utilize samples that will influence the model to generate correct outputs to its advantage. These samples are also known as adversarial examples. Initially, the generator will be trained to generate these adversarial examples. Then, a new loss term will be introduced to the generator to penalize it for producing these adversarial examples. This literature's main contribution is introducing an autoencoder-based discriminator, IQ strategy, and AEM to improve the quality and diversity of the generated images. The original literature of ArtGAN-AEM also showcased that the generated images on real-life images have a much higher quality. This may be due to the lack of images in the dataset, which demonstrates a reliance on a large dataset to be able to perform well.

2.2.3 StyleGAN

StyleGAN (Karras, Laine and Aila, 2019) is indeed an evolution of the ProGAN model that incorporates progressive training but with significant modifications to the generator and discriminator networks. Figure 2.7(a)below showcases the original architecture of the Progressive GAN model, and figure 2.7(b) is the StyleGAN architecture. One of the more apparent additions is the existence of the mapping network f, a mapping network consisting of 8 fully connected layers. Instead of receiving latent code z directly into the input layer, Latent code z will be directed into the Mapping network f, and the mapping network will generate a Style vector W.

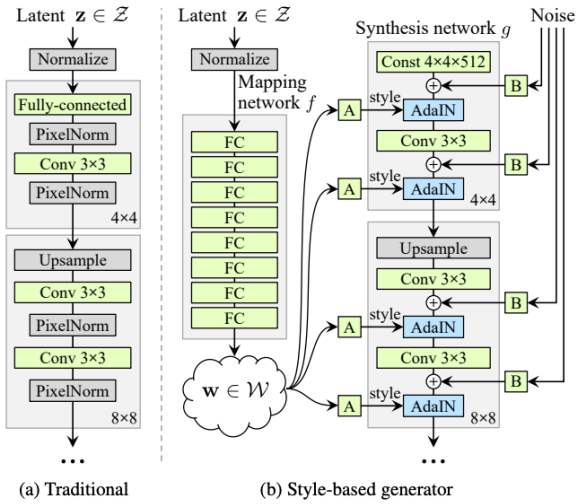

Figure 2.7: StyleGAN model architecture from (Karras, Laine and Aila, 2019)

StyleGAN replaces Pixel-wise normalisation (Pixel Norm) from ProGAN with Adaptive Instance normalisation (AdaIN) and changes the Latent point input to a constant value. Gaussian noise B, shown as a yellow box labelled "B" in figure 2.7, is added to each activation map before the AdaIN block to introduce some variation to the fake images; additionally, the image quality seem to improve. The latent point input was also replaced with a constant value to maintain consistent image quality; this, coupled with the Gaussian noise, will provide the model to generate a large variety of images with consistent image quality. Styles of images can also be controlled using mix regularisation, which is a concept that utilises multiple mapping networks to generate different style vectors w. A split point is then introduced to split the Synthesis network into separate sections, and each section is fed with a different style vector w to control various features of the generated image. The modifications introduced in StyleGAN provide the ability to control the style of specific attributes in the generated image, and significant improvements can be observed in the generated images. The images are always up-sampled before repeating the process for the next resolution block.

Figure 2.8: Water droplet examples from (Karras et al., 2019)

However, this model comes with a minor flaw when generating images; images generated by StyleGAN contain artifacts known as water droplets. Another reason for not selecting this model is due to the model's need for a large dataset, similar to the other GAN models mentioned above.

Chapter 3

Method

This section of the literature will provide an insight into the dataset and model used for the experiment. The reason for selecting the dataset and model, as well as any modifications made to the dataset and the reasons for those modifications, will also be discussed.

3.1 Dataset

The quality and quantity of an image dataset are incredibly important to train a good GAN model. According to a blog from NVIDIA, a dataset of 50,000 to 100,000 images is usually needed to train a high-quality GAN (Salian, 2021). Training a model with a low amount of images will lead to a risk of the discriminator overfitting (Karras et al., 2020b). However, training the model with a dataset too large will require a significant amount of computational power, a long processing time, and storage. The amount of resources required may not be accessible to the public. To ensure that the dataset is suitable for our training, it must satisfy a few criteria. The dataset must be square images of the same size; avoid using images with transparent features or alpha channels, colour mode of each image in the dataset must be consistent and avoid datasets with a high amount of duplicated images as it will influence the model to generate a style that is similar to the most frequently appeared image.

This literature utilizes the WikiArt(Wikiart, 2019) dataset, which contains about 80,000 images divided into genres, artists, and media. This is the same dataset that was used in the ArtGAN literature, but for this study, only the nude painting dataset will be used, which consists of around 3,000 images. Nude paintings are particularly interesting to work with due to their unique characteristics. The dataset was also processed to have a resolution of 128x128, which serves as another measure to reduce the computational power and processing time needed for the GAN training process. The colour mode of each image from our dataset was also converted to RGB (Red-Green-Blue) colours. The code for processing the dataset is available in Appendix A.1.

Figure 3.1: Nude art images from (Wikiart, 2019)

As shown in figure 3.1, Nude painting typically features a naked human subject posing in a room. Depending on the posture, setting, and characteristics of the subject, the sense of space, depth, and mood conveyed by the colours can vary significantly. Although most naked humans share similar body features such as head, body shape, and skin colour, it is interesting to observe how well a model can learn these features and potentially generate visually indeterminate outputs.

Another dataset known as Ukiyo-e faces dataset (Pinkney, 2020) was also used for our experiment to test the selected model initially. As shown in figure 3.2 below.

Figure 3.2: Ukiyo-e art images from (Pinkney, 2020)

This dataset is very different from the nude painting in the sense that the art feels more two-dimensional. This means that the sense of space and depth of this dataset is significantly different from the nude painting. The dataset was also processed differently in the mind that it was only reduced to 256x256 resolution instead of 128x128. For the testing stage, 1000 images were also extracted from the original dataset to construct the dataset used to train our model. This dataset served as a test of the performance of our model with a small dataset.

3.2 Selected Model

This literature will utilize StyleGAN2-ADA (Karras et al., 2020a) for the experiment. StyleGAN2-ADA is an improved version of StyleGAN2 (Karras et al., 2019), with minor differences so that both models will be introduced together.

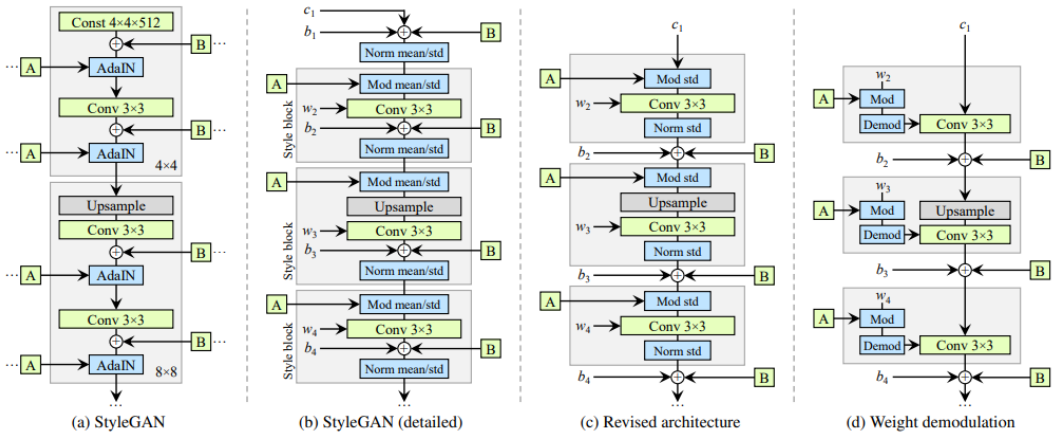

Figure 3.3: The comparison between StyleGAN architecture and StyleGAN2 by (Karras et al., 2019)

3.2.1 AdaIN Operation

This section will go into detail of AdaIN Operation originally introduced by (Huang and Belongie, 2017), each of the variable will be broken down and explained.

$$\text{AdaIN}(x_i, y) = y_{s,i} \cdot \frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i} \tag{3.1}$$

The equation 3.1 above represents the AdaIN operation, which takes the variables x_i and y as inputs. Here, x_i denotes the i-th channel of the feature map x generated by a convolutional neural network (CNN), as shown in Figure 3.3 by the yellow block labelled "Conv 3x3." On the other hand, y is generated by the yellow block labelled "A" in the same figure, and $y = (y_{s,i}, y_{b,i})$, where $y_{s,i}$ and $y_{b,i}$ represent the style parameter and bias parameter, respectively, for the feature map x with the i-th channel.

$$o(x_i) = \frac{x_i - \mu(x_i)}{\sigma(x_i)} \tag{3.2}$$

The equation 3.2 above represents the feature maps normalisation process of AdaIN operation. The equation first subtracts the mean value $\mu(x_i)$ of the i-th channel from x_i, and divides it by the standard deviation $\sigma(x_i)$ of the i-th channel. This operation is performed on all feature maps, and the resulting normalised feature map is denoted as $o(x_i)$ in the equation.

$$\text{Mod}(o_i, y) = y_{s,i} \cdot o_i + y_{b,i} \tag{3.3}$$

The equation 3.3 represents the modulation process of the AdaIN operation. The standard deviation component $y_{s,i}$ is used to scale the normalized i-th feature map o_i by multiplication, and the mean component $y_{b,i}$ is added to the scaled normalized feature map of the i-th channel. This is performed in the "Mod mean/std" block from figure 3.3.

3.2.2 StyleGAN2 Modification

Figure 3.3 showcases how StyleGAN evolved into StyleGAN2. The goal of StyleGAN2 was to eliminate the normalisation artifacts generated by the AdaIN operation. StyleGAN2 removed the constant bias b_1, noise from the yellow block B, and normalisation Norm mean/std block, as shown in Figure 3.3(b), from the input, as shown in the input of Figure 3.3(c). The addition of bias b_2 and yellow block B, which contains the noise input reallocated outside of the block, allowed the results of the model to be more manageable and predictable. StyleGAN2 also removed the mean component from both the normalisation and modulation equations, as the literature discovered that the mean component essentially does not provide any improvements to the overall model. Therefore, by removing the mean component, the processing time of the model can be reduced. From Figure 3.3(d), we can observe that the modulation is combined with the convolutional operations and normalisation is also replaced by weight demodulation. The combination of modulation and convolutional operations results in the following equation:

$$w'i, j, k = s_i \cdot wi, j, k \tag{3.4}$$

The next step is to replace normalization with demodulation, which achieves the same goal as normalization, but the equation is more direct and tailored to the modulation changes. Here, the modulated weight is equal to the product of the original weight and the scale corresponding to the i-th input feature map. Both the modulated weight and the initial weight are also associated with the spatial coordinates j, k of the i-th input feature map.

3.2.3 Adaptive Discriminator Augmentation(ADA)

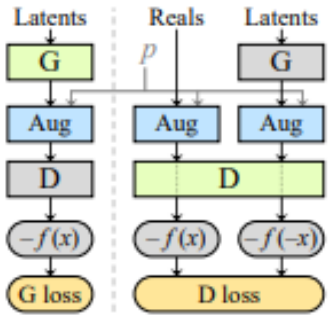

Figure 3.4: Augmentation process of StyleGAN2-ADA by (Karras et al., 2019)

The idea behind ADA is to enable the model to train on a limited amount of data. When using a small dataset, the discriminator may see the same image repeatedly, causing it to memorize the real images, which can harm the generator's ability to create new images and lead to overfitting. As shown in Figure 3.4, both the real and generated images are augmented with a probability $p \in [0, 1]$ before being fed into the discriminator. The output of the discriminator is

then processed using $-f(x)$ to remove the augmentation from the image and calculate the loss function. However, the introduction of augmentations may also introduce a problem known as leakage. Leakage occurs when the attributes of the augmentation leak into the generator, causing the generated output to appear augmented as well. The types of augmentations are split into six categories: pixel blitting, geometric transformations, colour transformations, image-space filtering, additive noise, and cutout. Therefore, ADA is introduced so that the probability of augmentation does not need to be manually tuned. Instead, the probability will be changed dynamically based on the degree of overfitting of the model.

3.2.4 Reason for selected model

The primary purpose of using the PyTorch version of StyleGAN2-ADA is to generate high-quality and diverse images with a smaller dataset. Compared to the original TensorFlow version, The PyTorch version of the model is 5-30 percent faster. The overall benefit is the ability to generate higher-quality images in a shorter amount of time with a smaller dataset, and a smaller size dataset can also be excused, as the ADA component is able to augment the dataset to generate diverse images. The tuning of the model was also made to be self-sufficient, as most of the tuning is done automatically as the training process. Therefore, no unnecessary resources are wasted on testing the model on different parameters. Do note that the code may be updated depending on the time the code was accessed. The main code used for our model can be accessed from: https://github.com/NVlabs/stylegan2-ada-pytorch. (NVlabs, n.d.)

3.2.5 Method of evaluation

Fréchet Inception Distance (FID) (Heusel et al., 2017) will be used to evaluate the generated images by our model quantitatively; this is for consistency purposes as many of the image syntheses utilise this FID to provide an evaluation and comparisons. FID is typically used to evaluate the quality of generated images; FID scoring is inversely proportionate to the quality of the generated images. Meaning a lower FID score equals a high-quality generated image. This literature also utilised Nearest Neighbour evaluation to analyse the nearest night of real images. This is an evaluation method used in the ArtGAN(Tan et al., 2017) literature to analyse the top 3 most similar generated images to a real image and make a comparison; this enables us to ensure that the model did not memories the training set and check the diversity of the images.

Chapter 4

Experiment

This section of the literature will discuss the experiment conducted for this study, the preparations made for the experiment, and the results of the experiment. Note that some of the visual evaluations made are subjective to individuals.

4.1 Challenges

The complexity of using this model comes from the process of setting the model up. Depending on the time of access, very different libraries and packages may be required for the model to be working. Even with the knowledge of what libraries and packages to have, the way of installing the packages may change and vary due to the resources available to you. Allocation of resources may also be challenging, for individuals without a good GPU or NVIDIA driver may need to source for a stronger GPU or use online Cloud GPU services. This project was done using The google Colab cloud(Google, 2018) GPU service. However, depending on the date and time of access, this service may differ or not be available for usage. Similar to the code used for the StyleGAN2-ADA PyTorch Model(NVlabs, n.d.).

The process of selecting a model can be a complicated process. If the experiment is done by individuals with very limited resources, the training process of a single image synthesis can take up to days or weeks. Therefore, problems such as augmentation leakage, mode collapse and overfitting take a much longer time to identify as it requires a long time to observe the results.

4.2 Experiment Settings

The literature uses the original code of StyleGAN2-ADA PyTorch to train with our selected dataset. The images in our dataset are resized to 128x128 resolution and converted to RGB for consistency. This is to reduce the computational power and time required to train the model. An additional benefit of this resolution is that it matches the image sizes used in ArtGAN, providing a fairer ground for comparisons between the results. When it comes to training the StyleGAN2-ADA PyTorch model, the process of setting up is much more complex, including tuning the hyper-parameters. This is due to the model's ability to tune its own hyperparameters automatically. Even parameters such as the augmentation probability are dynamically tuned using ADA.

Discriminator	Parameters	Buffers	Output shape	Datatype
b128.fromrgb	512	16	[32, 128, 128, 128]	float16
b128.skip	32768	16	[32, 256, 64, 64]	float16
b128.conv0	147584	16	[32, 128, 128, 128]	float16
b128.conv1	295168	16	[32, 256, 64, 64]	float16
b128	-	16	[32, 256, 64, 64]	float16
b64.skip	131072	16	[32, 512, 32, 32]	float16
b64.conv0	590080	16	[32, 256, 64, 64]	float16
b64.conv1	1180160	16	[32, 512, 32, 32]	float16
b64	-	16	[32, 512, 32, 32]	float16
b32.skip	262144	16	[32, 512, 16, 16]	float16
b32.conv0	2359808	16	[32, 512, 32, 32]	float16
b32.conv1	2359808	16	[32, 512, 16, 16]	float16
b32	-	16	[32, 512, 16, 16]	float16
b16.skip	262144	16	[32, 512, 8, 8]	float16
b16.conv0	2359808	16	[32, 512, 16, 16]	float16
b16.conv1	2359808	16	[32, 512, 8, 8]	float16
b16	-	16	[32, 512, 8, 8]	float16
b8.skip	262144	16	[32, 512, 4, 4]	float32
b8.conv0	2359808	16	[32, 512, 8, 8]	float32
b8.conv1	2359808	16	[32, 512, 4, 4]	float32
b8	-	16	[32, 512, 4, 4]	float32
b4.mbstd	-	-	[32, 513, 4, 4]	float32
b4.conv	2364416	16	[32, 512, 4, 4]	float32
b4.fc	4194816	-	[32, 512]	float32
b4.out	513	-	[32, 1]	float32
---	---	---	---	---
Total	23882369	352	-	-

Generator	Parameters	Buffers	Output shape	Datatype
---	---	---	---	---
mapping.fc0	262656	-	[32, 512]	float32
mapping.fc1	262656	-	[32, 512]	float32
mapping	-	512	[32, 12, 512]	float32
synthesis.b4.conv1	2622465	32	[32, 512, 4, 4]	float32
synthesis.b4.torgb	264195	-	[32, 3, 4, 4]	float32
synthesis.b4:0	8192	16	[32, 512, 4, 4]	float32
synthesis.b4:1	-	-	[32, 512, 4, 4]	float32
synthesis.b8.conv0	2622465	80	[32, 512, 8, 8]	float32
synthesis.b8.conv1	2622465	80	[32, 512, 8, 8]	float32
synthesis.b8.torgb	264195	-	[32, 3, 8, 8]	float32
synthesis.b8:0	-	16	[32, 512, 8, 8]	float32
synthesis.b8:1	-	-	[32, 512, 8, 8]	float32
synthesis.b16.conv0	2622465	272	[32, 512, 16, 16]	float16
synthesis.b16.conv1	2622465	272	[32, 512, 16, 16]	float16
synthesis.b16.torgb	264195	-	[32, 3, 16, 16]	float16
synthesis.b16:0	-	16	[32, 512, 16, 16]	float16
synthesis.b16:1	-	-	[32, 512, 16, 16]	float32
synthesis.b32.conv0	2622465	1040	[32, 512, 32, 32]	float16
synthesis.b32.conv1	2622465	1040	[32, 512, 32, 32]	float16
synthesis.b32.torgb	264195	-	[32, 3, 32, 32]	float16
synthesis.b32:0	-	16	[32, 512, 32, 32]	float16
synthesis.b32:1	-	-	[32, 512, 32, 32]	float32
synthesis.b64.conv0	1442561	4112	[32, 256, 64, 64]	float16
synthesis.b64.conv1	721409	4112	[32, 256, 64, 64]	float16
synthesis.b64.torgb	132099	-	[32, 3, 64, 64]	float16
synthesis.b64:0	-	16	[32, 256, 64, 64]	float16
synthesis.b64:1	-	-	[32, 256, 64, 64]	float32
synthesis.b128.conv0	426369	16400	[32, 128, 128, 128]	float16
synthesis.b128.conv1	213249	16400	[32, 128, 128, 128]	float16
synthesis.b128.torgb	66051	-	[32, 3, 128, 128]	float16
synthesis.b128:0	-	16	[32, 128, 128, 128]	float16
synthesis.b128:1	-	-	[32, 128, 128, 128]	float32
---	---	---	---	---
Total	22949277	44448	-	-

Figure 4.1: Generators and Discriminators Layers

The figure4.1 provide a detail of the neural network used for our specific models. The progressive training of the layers can be observed in the figure above, where the generator incrementally increases in resolution from 4x4 to our desired image size of 128x128, as shown in the output shape. The discriminator, on the other decrease from 128x128 to 4x4; this is the reason why a higher image resolution will cost a lot more computational power and resources as there will be the need to be processed through an additional layer of neural network. The processing stage through 3000 images of 256x256 compared to 128x128 is also a huge difference. The cost to process images of a higher resolution and through an additional layer of the neural network will scale up as the iterations increase.

However, training images to a higher resolution will allow the model to access to finer details of the images. The details and features within the image would have been further broken down by the model itself, resulting in a higher image quality. Therefore, if provided with the resources it is recommended to train a model at a higher resolution. Mentioned in the (NVlabs, n.d.), the model was also designed to have a 35 percent increase in inference speed which allows the continuation of any trained model to be much faster. This is incredible as the inference time of this StyleGAN2-ADA is significant, and the training continuation happens frequently enough for this to be an instrumentally welcomed benefit.

4.3 Results and Evaluation

Dataset	Number of Images	Image Resolution	FID
Wikiart Nude Painting	3000	128x128	27.51
Ukiyo-e Face	1000	256x256	71.93

Table 4.1: FID of the image generated by the selected model

The table 4.1 above showcases the information about the dataset used and the Fréchet t Inception Distance(Heusel et al., 2017) of the our generated images. FID is often used to quantitatively analyse the quality of images generated by image synthesis models. It measures

the differences between the feature vector of the real images and generated images in the feature space. Lower FID represents a higher-quality image.

Experiment 1:

Figure 4.2: Ukiyo-e Face results produced by our trained Model

Figure 4.2 showcase the results of our model trained with Ukiyo-e Faces, as shown in the results at an FID of 71.93. The result seems to have successfully generated results with varying facial features. Most of the results also successfully learn to generate all of the facial features without missing an eye or a lip. But there are some of the examples with very distorted faces. However, there seem to have some augmentation leakages when it comes to the training of this dataset, as most of the results produced have a pink theme to the images. In the early stages of training, another strange augmentation seems to have leaked into the generator, causing all the generated results to have the same augmentation.

Figure 4.3: Augmentation Leakage

As shown in 4.3 above are images of the older training results. Interestingly, these results have more diverse colours, but there seems to be some weird augmentation leakage causing the images to be rotated in a single direction. Another observation is that even when all of the images produced have varying styles of facial features, all of the results seem to have the characters facing towards a single direction. This may be due to some mode collapse issues as most of the real images seem to be facing in very diverse directions.

Experiment 2:

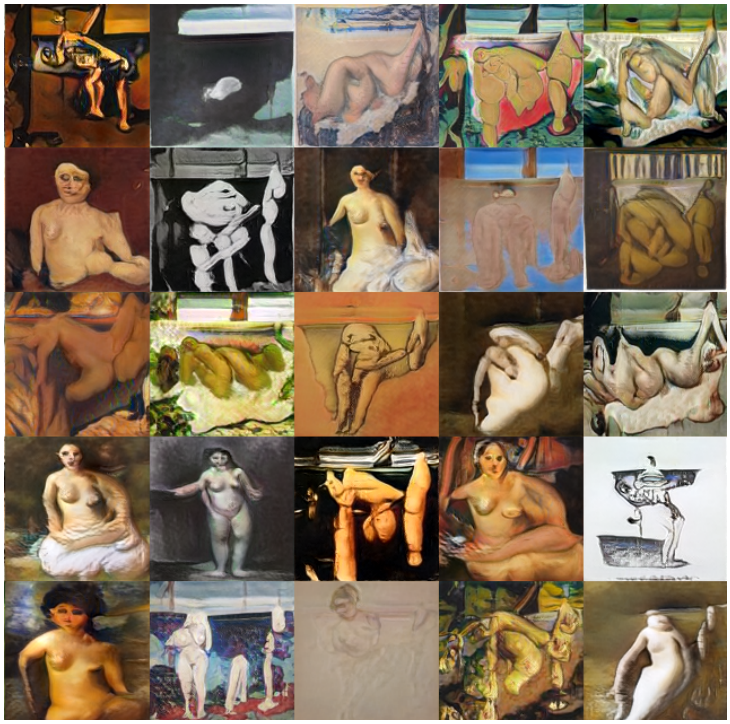

Figure 4.4: Nude painting results produced by our trained Model

The figure 4.4 above showcase the results of our model trained using the Wikiart(Wikiart, 2019) nude dataset. The results produced are significantly more diverse, and the quality of the images is also much higher. Visually the model did manage the sense of depth and space for nude paintings. By making a comparison between this and previous 4.2 results, Ukiyo-e Face has more than double the FID compared to Wikiart Nude Painting, but it appears to be easier to identify the art style of the generated results for Ukiyo-e Face dataset. This may be due to several reasons, firstly this may be due to the potential mode collapse that has occurred in the first experiment causing the generated images to be very similar as, generally, this generated style is able to fool the generator, but when the generated results are evaluated with the real images with FID the overall diversity and style of the real images did not get translated well into the generator. Another more interesting reason is that due to the overall complexity between the two art dataset, Ukiyo-e faces is significantly less complex in terms of the colour tones, sense of depth and realism. Even with a much smaller dataset and more than double the FID, the results produced by the first experiment is visually easier to identify in comparison to the second experiment. However, in terms of the training process. Nude painting does not seem to have the same augmentation leakage issue experienced by Ukiyo-e face even when trained with the exact same initial parameters.

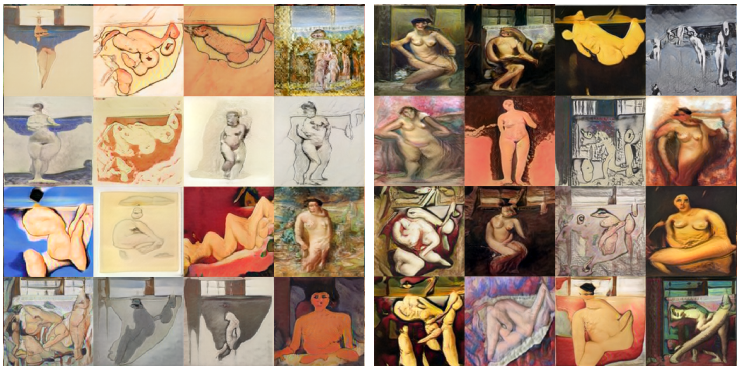

Figure 4.5: Bad (Left) and Good (Right) results produced by our model

Figure 4.5 above showcases the good and bad results of our trained model for a nude painting. The model seems to struggle when generating images with the nude artist lying down; the style of the painting also appears to be challenging for the model to generate. One of the observations found is that art with a bright colour tone seems to lack a lot of details and, in most cases, is challenging to observe. This may be due to the darker colour tone painting having more obvious details and clarity in comparison to the lighter tone paintings, causing the model to struggle when learning the features of more delicate tone paintings.

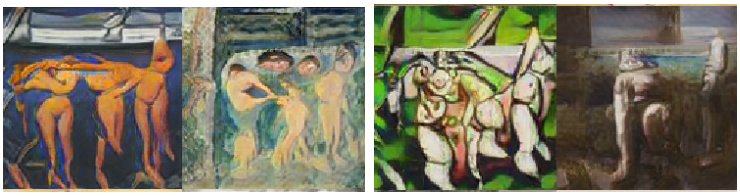

Figure 4.6: Example of our model attempt to generate multiple individuals

Another struggles the model has is the ability to generate paintings with multiple nude models; figure 4.6 below shows some examples of the model attempting to generate paintings with numerous nude artists. The levels of detail generated for a single nude model painting are much higher in comparison to paintings with multiple nude models. Their model also seems to merge various into a single subject. In most cases, there is some sort of connection in the subjects of the painting; this may be due to the model not being used to develop multiple subjects with the human body colour tone. Therefore merging the subject into one.

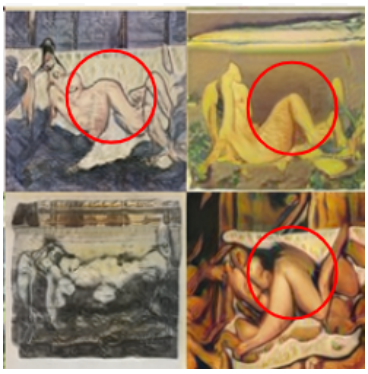

Figure 4.7: Example of our model attempt to generate images of the subject lying down

The figure4.7 above showcases some of the more obvious examples of the model struggling to generate images of the subject lying down. Interestingly, there is a trend in the generated image lying down. The feature circled red in figure 4.7 is a common feature that is present in most of the attempts to learn paintings with the subject lying down. This trend can even be observed in 4.5 and 4.4 shown above.

Nearest Neighbour Evaluation

Figure 4.8: Results generated with our trained model: Images in red line is the real image, the 3 image surrounding the real image with red line from the right are the nearest neighbour generated image of the real image.

Real Image	Generated Image 1	Generated Image 2	Generated Image 3
1	Distance = 20790.29	Distance = 20910.83	Distance = 20998.76
2	Distance = 22951.68	Distance = 23460.35	Distance = 23989.39
3	Distance = 23804.46	Distance = 24719.83	Distance = 24851.15

Table 4.2: L2 Distance measurements between real and generated images, Real image 1 is the most left image with a red box surrounding it.

The Figure 4.3 above showcases the nearest neighbours of the images. This method of evaluation was inspired by the work of ArtGAN(Tan et al., 2017). The purpose of this evaluation is to demonstrate that the model does not memorize the training set. As shown above, the generated results are only slightly similar in the higher-level details, such as background colours and body postures, but visually, the finer details are different. Therefore, the model was proven not to memorize the training set. This is also useful for visualizing various examples of the style in terms of the background colour and posture of the subject in the art. The distance is measured using L2/Euclidean distances between each generated image and each training image.

$$\text{L2 distance} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.1}$$

The code of this process is located in Appendix A.2. Essentially, the L2 distance is calculated using the equation 4.1 shown above. The calculation calculates the differences between the pixel value for our training image and generated image.

Figure 4.9: Image from ArtGAN-AEM for nude painting, image in the between the red line is the real image and the images to right are the generated results.

The figure 4.9 below can be used to compare the improvement of StyleGAN2-ADA over ArtGAN-AEM visually. When looking at Figure 4.8 from a distance, the visual indeterminacy mentioned previously can be observed. Some notable differences between the ArtGAN-AEM result and our result are visible, including visible distortions in the results produced by StyleGAN2-ADA. Some of the results seem to generate head and body to some degree, and varying postures can be observed. There are some results that appear to show subjects sitting down. Results from the ArtGAN-AEM, on the other hand, seem to fail to generated body features.

Chapter 5

Conclusions and Future Work

5.1 Challenges

The work showcased in this literature was a pivot from the original proposal; the project was modified after the phase of the building model. Due to the lack of resources, the project was modified to be more feasible after a discussion with the supervisors. When building a GAN model, resource requirements should be an essential consideration during the research phase. As some models can only operate under certain drivers, for example, StyleGAN2-ADA requires very specific CUDA-Compatible with at least 11GB of VRAM and without an NVIDIA driver, the process of setting up will be incredibly difficult, which could harm the project process. This is incredible as the inference time of this StyleGAN2-ADA is significant, and the training continuation happens frequently enough for this to be an instrumentally welcomed benefit.

5.2 Conclusion

Visually the generated results by the selected do seem to be a significant improvement over the older image synthesis models. Visual indeterminacy does seem to be more visible in results generated by our model as well, compared to the results generated by ArtGAN-AEM. Our model does seem successfully generate some body features, although only to a certain degree; the ADA component does prove to be extremely useful, although augmentation leakage did happen in the first experiment conducted. StyleGAN2-ADA was initially designed to generate high-quality faces, and models such as ArtGAN and ArtGAN-AEM are more focused on generating Art images. However, StyleGAN2-ADA is still much better at art image synthesis. This shows an outstanding level of progression in image synthesis in the field of art. Moving on to some of the key findings, the augmentation leakage that occurred was an interesting case; it is unclear if the leakage is caused by the size of the dataset or the features of the art style, as these are the only varying factors between the two trained models.

We also discovered that the model struggles to generate art with varying amounts of subjects, as most of the paintings in our dataset consist of paintings with one subject. The model seems to pivot towards the direction of generating a single subject and resulting in a painting with multiple subjects merging into one. The model also appears to be much better at generating darker-tone paintings; this is due to darker-tone paintings having more apparent features in comparison to light-tone images. Lastly, the model also seem to struggle generating images

with varying composition. We also observe that even when the model struggles to generate images of the subject lying down, there seem to be a trend in the features that spread amongst most of the pictures that tries to mimic the same postures. The key takeaway of this literature is that art image synthesis is much more complicated in comparison to the initial thought provided in the Introduction, problems such as the image having varying amount of subjects, postures, composition is an extended list of struggles for models to learn. These are problems that stretches even towards model trained with dataset of real-life images. In conclusion, there is no doubt that a significant improve can be seen for image synthesis and the reward for all these is that we get observe more and discover more problems that could potentially lead to a more significant improvement of image synthesis models. There is also an understanding now on why most models trained on dataset with a single or consistent subject.

Word Count: 6335

5.3 Future Work

Improvements can be made to the training process of the experiment. A more comprehensive range of art styles can be used for comparison, and the training process can be extended to improve results. The quality of the dataset can be improved by using a larger dataset with a higher resolution. StyleGAN2-ADA trains better at higher resolutions, but due to resource and time constraints, a high resolution was not utilized. Furthermore, it could also be interesting to train similar datasets on different modern image synthesis models to make additional comparisons. Identifying the correlation of models to art style characteristics could be helpful too. There may be a potential where a specific model is better at generating certain characteristics while other models are better at generating other characteristics.

Training datasets with varying amounts can be an exciting topic to explore as well. By training a model with images of one subject to a certain satisfactory level of image quality will and then we prepare the same model with the same trained parameters with a duplicate of the subject or multiple subjects, will the model be able to recognise or learn to just duplicate the subject with similar features into multiple or relearn all the features? If provided with more time and resources, training the images with a much higher resolution would also be interesting to evaluate the models at different resolution levels.

Bibliography

Berger, J., 1972. *Ways of seeing*. BBC and Penguin.

Berthelot, D., Schumm, T. and Metz, L., 2017. Began: Boundary equilibrium generative adversarial networks. *arxiv preprint arxiv:1703.10717*.

Doersch, C., 2016. Tutorial on variational autoencoders. *arxiv preprint arxiv:1606.05908*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2020. Generative adversarial networks. *Communications of the acm*, 63(11), pp.139–144.

Google, 2018. Google colaboratory [Online]. Accessed: May 4, 2023. Available from: `https://colab.research.google.com/`.

Hertzmann, A., 2020. Visual indeterminacy in gan art. *Acm siggraph 2020 art gallery*. Association for Computing Machinery, pp.424–428.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S., 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Huang, X. and Belongie, S.J., 2017. Arbitrary style transfer in real-time with adaptive instance normalization. *Corr* [Online], abs/1703.06868. `1703.06868`, Available from: `http://arxiv.org/abs/1703.06868`.

Isola, P., Zhu, J., Zhou, T. and Efros, A.A., 2016. Image-to-image translation with conditional adversarial networks. *Corr* [Online], abs/1611.07004. `1611.07004`, Available from: `http://arxiv.org/abs/1611.07004`.

Karras, T., Aila, T., Laine, S. and Lehtinen, J., 2017. Progressive growing of gans for improved quality, stability, and variation. *Corr* [Online], abs/1710.10196. `1710.10196`, Available from: `http://arxiv.org/abs/1710.10196`.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. and Aila, T., 2020a. Training generative adversarial networks with limited data. *Corr* [Online], abs/2006.06676. `2006.06676`, Available from: `https://arxiv.org/abs/2006.06676`.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. and Aila, T., 2020b. Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33, pp.12104–12114.

Karras, T., Laine, S. and Aila, T., 2019. A style-based generator architecture for generative adversarial networks. *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*. pp.4401–4410.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J. and Aila, T., 2019. Analyzing and improving the image quality of stylegan. *Corr* [Online], abs/1912.04958. 1912.04958, Available from: http://arxiv.org/abs/1912.04958.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. et al., 2017. Photo-realistic single image super-resolution using a generative adversarial network. *Proceedings of the ieee conference on computer vision and pattern recognition*. pp.4681–4690.

NVlabs, n.d. Nvlabs/stylegan2-ada-pytorch: Stylegan2-ada - official pytorch implementation. Available from: https://github.com/NVlabs/stylegan2-ada-pytorch.

Pepperell, R., 2006. Seeing without objects: Visual indeterminacy and art. *Leonardo*, 39(5), pp.394–400.

Pinkney, J.N.M., 2020. Aligned ukiyo-e faces dataset. https://www.justinpinkney.com/ukiyoe-dataset.

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

Salian, I., 2021. Nvidia research achieves ai training breakthrough. Available from: https://blogs.nvidia.com/blog/2020/12/07/neurips-research-limited-data-gan/.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. and Chen, X., 2016. Improved techniques for training gans. *Advances in neural information processing systems*, 29.

Tan, W.R., Chan, C.S., Aguirre, H. and Tanaka, K., 2017. Artgan: Artwork synthesis with conditional categorical gans. 1702.03410.

Tan, W.R., Chan, C.S., Aguirre, H.E. and Tanaka, K., 2018. Improved artgan for conditional synthesis of natural image and artwork. *Ieee transactions on image processing*, 28(1), pp.394–409.

Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X. and Wang, F.Y., 2017. Generative adversarial networks: introduction and outlook. *Ieee/caa journal of automatica sinica*, 4(4), pp.588–598.

Weng, L., 2019. From gan to wgan. *arxiv preprint arxiv:1904.08994*.

Wikiart, 2019. Wikiart.org - visual art encyclopedia. Available from: https://www.wikiart.org/.

Zhu, J.Y., Park, T., Isola, P. and Efros, A.A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the ieee international conference on computer vision*. pp.2223–2232.

Appendix A

Code

A.1 File: Downsize.py

```
from PIL import Image
import os

def resize_images(rawImg, trainSet, targetsize):
    for filename in os.listdir(rawImg):
        if filename.endswith('.jpg'):
            #Open the image
            with Image.open(rawImg + '/' + filename) as img:
                try:
                    #Resize the image
                    with img.resize(targetsize):

                        try:
                            img = img.convert("RGB")
                            resized_img = img.resize(targetsize)
                            #Save the resized image to the output directory
                            resized_img.save(trainSet + '/' + filename)
                        except:
                            print("file corrupted")

                except:
                    print("error")

#Change the rawImg to the unprocessed images path
#Change the trainSet to the images path that you wished to store the processed images
rawImg = 'Other Dataset/nude-painting-nu'
trainSet = 'nu128(2)'

#Change this to the size of the desired size of the image, ensure the image is square
    shape.
#Recommended sizes: (128,128), (512,512) and (1080,1080)
imgSize = 128

recSize = [128,256,512,1024]
# Check if the size is right
if (imgSize not in recSize):
    print("Size error: Please consider a new image size")
targetsize = (imgSize, imgSize)

# Call the resize_images function with the input directory, output directory, and new
    size
resize_images(rawImg, trainSet, targetsize)
```

A.2 File: L2.py

```
import os
```

```
import cv2
import numpy as np


# Variables to change
trainingImg = "nyest"#Path to folder containing training images
genArt = cv2.imread("khc73_Code\sample_result.png")#Path to folder containing generated
    result
imgSize = 128 #Size of each generated image, make sure the sizes are 128,256,512,1024


# Get a list of all image files in the folder
imgFiles = [f for f in os.listdir(trainingImg) if f.endswith(".jpg")]

rec_sizes = [128,256,512,1024]
# Check if the size is right
if (imgSize not in rec_sizes):
    print("Size error: Please consider a new image size")


# Load each image file into a numpy array
def trainNp(imgFiles):
    trainDataset = []
    for imgFile in imgFiles:
        imagePath = os.path.join(trainingImg, imgFile)
        image = cv2.imread(imagePath)
        trainDataset.append(image)
    trainDataset = np.array(trainDataset)
    return trainDataset

#Split the generated artwork into individual images
def splitGen(imgSize, genArt):
    row = genArt.shape[1] // imgSize #Get the number of row
    col = genArt.shape[0] // imgSize #Get the number of column
    genImgs = []
    for i in range(col):
        for j in range(row):
            x1 = j * imgSize
            x2 = x1 + imgSize
            y1 = i * imgSize
            y2 = y1 + imgSize
            genImgs.append(genArt[y1:y2, x1:x2]) #Store up the split image
    return genImgs


print("test0")



def nearestN(imgSize,genArt,imgFiles):
    genImgs = splitGen(imgSize, genArt)
    trainDataset = trainNp(imgFiles)
    distances = np.zeros((len(trainDataset), len(genImgs)))
    for i, training_image in enumerate(trainDataset):
        for j, generated_image in enumerate(genImgs):
            distances[i][j] = np.linalg.norm(training_image-generated_image)
    # Find the index of the top 3 generated images that are most similar to each
        training image
    top = []
    for i in range(distances.shape[0]):
        closestIn = np.argsort(distances[i])[:2]
        top.append(closestIn)
    # Display the top 3 nearest neighbor images for each training image
    for i, genIm in enumerate(top):
        #open each image from the training set
        training_image = cv2.imread(os.path.join(trainingImg, imgFiles[i]))
        for j, o in enumerate(genIm):
            print("Distance = " + str(distances[j][o]))
            cv2.imshow("Generated Image", genImgs[o])
        cv2.imshow("Training Image", training_image)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

```
nearestN(imgSize,genArt,imgFiles)
```