Galaxy Imagery Synthesis using GANs

Submitted May 2021, in partial fulfillment of
the conditions for the award of the degree **BSc Computer Science.**

20029662

School of Computer Science
University of Nottingham

Abstract

Augment data has always been a significant element in computer vision. With the development of deep learning, the need for expansion of training set is required more, compared to the past decade. Unlike traditional ways like cropping and rotating the image, generative adversarial networks give us an alternative way to enlarge the training set by producing fake images. This project aims to study the performance of the generative adversarial network in the expansion of the galaxy image collection, compare two types of StyleGAN and explore the latent space. My experiments reveal that using higher resolution images as the training set has a longer convergence time and higher FID and KID scores. Through the adaptive discriminator augmentation, the overfitting problem in GAN training can be effectively reduced.

Acknowledgements

I would like to thank my supervisor, Dr Mercedes Torres Torres, for her guidance when I was confused about the project and the help when I was struggling. I would like to thank Manuel Jiménez for generously providing the data set of their project. I would also like to thank Dr Aaron Jackson for his patience and guidance on using the Nottingham GPU cluster. Additionally, I would like to express my gratitude to Matthew Yates, for his help on GPU cluster memory usage. Besides, I want to express my gratitude to Coursera, an online platform that allowed me to gain start-of-the-art GAN's knowledge. Finally, I would like to thank my roommates Ta-Liang Lu and Yujie Xie, for spending this difficult time with me in the same apartment under the panic of Covid-19.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Generative Adversarial Networks, known as GANs, is considered one of the most interesting developments in deep learning and artificial intelligence in the past decade. The most common way of applying GANs is to generate fake images, especially in domains like art and advertising. Besides, it can also be used to provide better astronomical images and then helps scientific (dark matter) research [1]. As the technology matures, the images generated by GANs have also been converted from grey-scale, low resolution to RGB, high resolution. For example, the images generated by BigGAN [2] and COCO-GAN [3] are almost indistinguishable from real images. In this way, GANs are now facing moral issues. It can be used to form fake faces to achieve personal sinister purposes. In 2019, a website 'This Person Does Not Exist' using StyleGAN was established [4]. Whenever the website is reloaded, a synthesis human face image is displayed. Although GAN was originally proposed for unsupervised learning, it has also been proved to be useful for semi-supervised learning [5], fully supervised learning [6], and reinforcement learning [7]. In general, Generative Adversarial Networks have a bright prospect in the future and are of great research value.

1.1 Motivation

With the continuous development of the central processing unit (CPU) and graphics processing unit (GPU), the ability of computers to process image data has become more and more powerful, and the ability to use deep learning to solve computer vision problems has become more and more popular. Under this circumstance, it is vital to use data augmentation for the training set to meet the natural need of deep learning. Traditional ways like geometric transformation, flipping, cropping, rotation are classical methods to achieve this [8]. However, GANs provide us with an alternative way to accomplish the goal.

With the popularity of GANs, hundreds of related models have been derived, of which nearly fifty are related to computer vision. In the data augmentation area, GANs could perform an excellent job. Specifically, in an unbalanced dataset [9], the training set can be significantly increased by generating new data that similar to the original ones [10]. Moreover, data augmentation could play a significant role in handling sensitive information, such as medical data [11].

The project is about using GANs (StyleGAN2 specifically) to synthesise merged galaxy images [12]. Previous studies on generative adversarial networks mainly chose natural creatures or human faces as training sets, such as ImageNet [13] and CelebA [14]. When it comes to generating galaxy images, previously published articles only used a specific GAN to generate images of individual celestial bodies [15][16]. In my project, very few merged galaxy images are chosen as my training set. I will present a comparative study with StylenGANs to determine the performance using different metrics and images with different resolutions as the training set. To the best of my knowledge, this field has never been explored before. Hopefully, in this way of galaxy image research, I could enhance some understanding of the field of image data augmentation, also obtain some inspiration for medical image data augmentation.

1.2 Aims and Objectives

The goal of this project is to synthesise more images of merging galaxy, as a way to augment data in the imbalanced dataset. The key objectives of the research are:

1. To explore the architecture behind these two GANs, especially the details of how to build a specific GAN, such as the method used to stabilise the training process.

2. To learn the metrics used to evaluate a GAN's performance.

3. To synthesise images of merged galaxies and measure their quality by calculating the metrics from time to time.

4. To compare the result using different resolutions of images.

5. To compare the performance between two different GANs by output.

6. To explore the latent space by interpolating between generated galaxy images.

7. To explore the latent space by performing vector arithmetic with generated galaxy images.

1.3 Description of the work

The project initially aimed to compare the results of two GANs using 64x64, 128x128, 256x256, 512x512 and 1024x1024 resolution training images. Due to the GPU memory problem, the project is unavailable of training 1024x1024 images. Since the training of a single GAN takes much time until its convergence, 256x256 image training is abandoned due to time limitation. As a result, the comparison between the two GANs is under 64x64, 128x128, and 512x512 images. The metrics used to compare two GANs are provided by the framework. However, the code for latent space exploration and vector arithmetic is implemented by myself.

The training set of this project is provided by one of the authors of the published paper "Galaxy Image Classification Based on Citizen Science Data: A Comparative Study [17]". All the experiments are accomplished in the University of Nottingham's GPU cluster, with cuda-10.0 and cudnn-v7.6.5.32-forcuda 10.0.

Two frameworks are used in the entire project. The first semester, since I was new to the computer vision field, I decided to use a simple framework, which helped me experience using GAN to create fake images. However, this framework only allows FID for evaluation. As a result, in the second semester, the official StyleGAN2-ADA framework is used, and a comparison study between StyleGAN2 with and without ADA is proposed.

PyTorch version of StyleGAN2 is used in the first semester, which can be available from `https://github.com/lucidrains/stylegan2-pytorch#multi-gpu-training`. By default, it uses the truncation trick [18] and sets the truncation value to 0.75 to balance fidelity and diversity. It allows users to set network capacity and allows multi-GPU training. Besides, it can also apply Differentiable Augmentation (DiffAugment) [19] so that a small training set (1-2k) can also generate high-quality pictures.

In the second semester, NVDIA's StyleGAN2-ADA TensorFlow implementation is used, which can be available from `https://github.com/NVlabs/stylegan2-ada`. Based on the original StyleGAN2 framework, this implementation made much improvement. One of the most significant achievements is introducing an adaptive discriminator augmentation mechanism [20], which makes the limited training set produce a much better result for GAN's training. Apart from that, it provides a more concise command-line tool and better hyper-parameter defaults. The framework allows more quality metrics calculations, including Fréchet Inception Distance, Kernel Inception Distance, Precision and Recall, and Inception Score, in which Fréchet Inception Distance and Kernel Inception Distance are used in my project.

Chapter 2

Background and Related Work

In this section, information about previously presented papers is provided. First, I will introduce the background of the dataset, which is a part of the citizen science projects. Secondly, I will introduce GAN, with some illustrious architectures developed in recent years. Then, I will present StyleGAN together with StyleGAN2, the problem of StyleGAN and how StyleGAN2 solved it. Next, data augmentation in the computer vision field will be reviewed, and a new way of augmenting data will be introduced. Finally, a summary of how GAN has used latent space for research in recent years is illustrated.

2.1 Citizen Science And Galaxy Zoo 1

Since the disparate galaxy images vary from different brightness, size, and distance, the task of labelling hundred of thousands galaxy images could be very complicated. Citizen science projects [21] alleviate the pressure of the researchers, with the participation of amateur volunteers, labels could be assigned to the training set in a short time instead of experts' tedious long term work. As a result, many studies used that information, including my project, which requires the Galaxy Zoo 1 (GZ1) [22], the first edition of Galaxy Zoo project [23], which is a part of the Zooniverse [24], a set of citizen science projects that study the universe. GZ1 focused on classification between spiral and elliptical galaxy images, providing 900k amateur labelled images and a subset of approximately

41K expert labelled images. Although the classification of spiral and elliptical images is the main task of the project, there are very few merged galaxies in the dataset. There is an imbalance between merged and unmerged galaxy pictures.

In this work, since the images labelled by amateurs has high uncertainty to be assigned with wrong labels, only images labelled by experts are chosen as my dataset. Within this small subset, images of merged galaxies account for a very small percentage, which meet our need for augment the data.

2.2 Generative Adversarial Network

Generative Adversarial Networks (GAN) [25] was first formally proposed by Ian Goodfellow and his colleagues in 2014. Two years later, a tutorial of GANs [26] was presented at Neural Information Processing Systems (NIPS) 2016, concretely described the motivation of using GAN, its usage, the comparison with Variational Auto-Encoder (VAE) [27], and some tricks to improve the training process. In the next few years, the GAN derivatives continued to produce, among which Deep Convolutional Generative Adversarial Network (DCGAN) [28] is the most famous one, and it is still used in physical, medical, and more research areas.

GAN consists of a pair of models: generator (G) and discriminator (D), whose role is to sample an image from a latent variable and judge whether the generated image is fake or real, respectively. Formally, the purpose of GAN is to reach a Nash Equilibrium by finding a state within a min-max problem following Equation (2.1):

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{2.1}$$

where D denotes the discriminator and G stands for the generator, z is a latent variable derived from a Gaussian distribution. D(x) denotes the probability that the discriminator estimates real data x is real, D(G(z)) denotes that the probability that the discriminator estimates that fake data G(z) is real. E_x is the expected value over all real data instances, E_z is the expected value over all random inputs to the generator. The formula derives from the cross-entropy between the real and generated distributions.

Due to the instability and sensitivity to hyper-parameters, people are finding ways to stabilise the training process by mostly designing or modifying the network architecture, learning objectives, regularisation methods, and heuristic tricks [29] through the years.

Applications of GANs show its ability to handle image manipulation, analysis, or characterization [30]. Classification and regression are certainly two tasks that most deep learning networks can cope with, including GAN. For instance, the output of convolutional layers of the discriminator is an ideal feature extractor. Besides, image analysis can be achieved by the recent Laplacian Generative Adversarial Network (LAPGAN) [31] with a coarse-to-fine fasion sample drawing. Furthermore, image-to-image translation [32] and text-to-image-to-text translation [33] are solved as challenging problems decades ago.

2.3 StyleGAN and StyleGAN2

StyleGAN [34] is considered one of the most state-of-the-art GANs right now and was proposed by generating extreme realistic human faces. The goal of StyleGAN can be concluded by generating high fidelity images, increasing the diversity of images, while having more control over the features. StyleGAN adds feature mapping, which makes the features more disentangled by using an intermediate noise. Besides, an extra noise is also passed into the generator model to add a stochastic variation on the generating images, and Adaptive Instance Normalization (AdaIN) is added after each convolutional layer [35] in the generator. StyleGAN trains the model uses progressive growing, as a way to generate high-resolution images.

By switching the intermediate noise at some specific layers, StyleGAN accomplished generating an image that is a mixture of two picture styles. StyleGAN also allows the perturbation of a single generated image, which means it has the ability to control super subtle features of the output image.

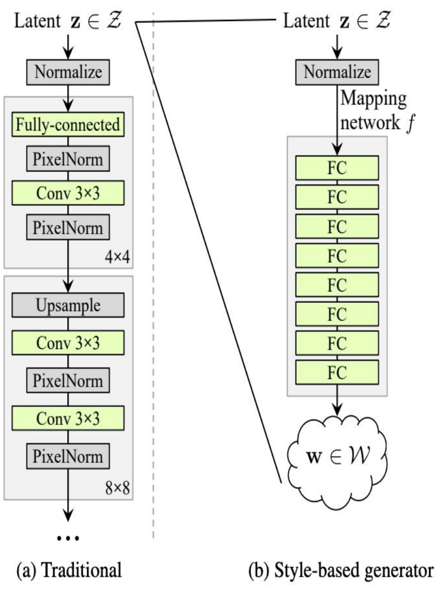


Figure 2.1: Comparison between GAN and StyleGAN

StyleGAN2 [36] was proposed by Tero Karras and his NVIDIA team in 2020 to remedy defects after generating high-resolution images. Blob-like artefacts can be observed in images generated by StyleGAN, which are more apparent when examining the intermediate feature map within the generator network. According to their paper, all feature maps with 64*64 resolution seem to be affected by this problem, and the situation gets worse if the resolution of the generated image is higher.

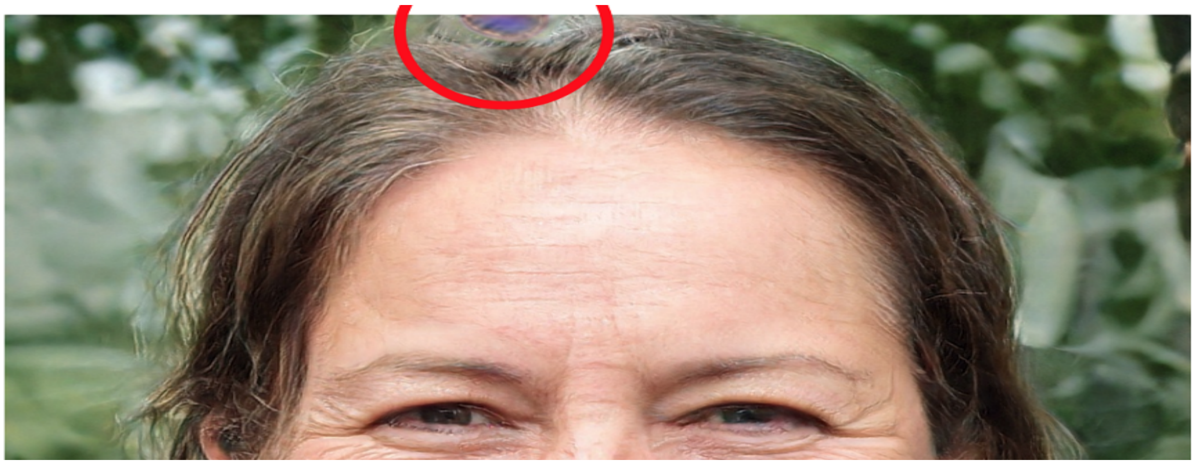


Figure 2.2: StyleGAN blob-like artefact

Due to this, the paper tracked instance normalisation in AdaIN layer and discovered that by normalising the mean and variance, disentangled information found in the magnitudes of the features could be potentially destroyed. They improved the original StyleGAN by replacing weight demodulation. In addition, StyleGAN2 proposed an alternative design in progress growing to stabilise high-resolution image training.

2.4 Data Augmentation and Adaptive Discriminator Augmentation

Previous studies indicate that data augmentation is a way to prevent overfitting in deep learning models [37]. More than that, it has been proved that data augmentation in feature space improves the performance of classification effectively [38]. VAE, an artificial neural network to learn efficient data codings, has been used to data augmentation [39]. GANs have also been discovered for data augmentation in recent years. Data Augmentation Generative Adversarial Networks (DAGAN) [40] shows that with augmentation, accuracy of classification on MNIST [41], Omniglot [42], and VGG-Face [43] datasets are improved.

GAN is a model whose goal is to maximise the similarity between synthesis and real distributions. However, due to this, if the real distribution has over or under representations of some classes, it would be a problem that the discriminator may have biased weight of specific features [44]. Traditional methods of augmenting data, for example, mirroring the image, improves the classification accuracy on specific dataset while not so effective on others [45]. Due to this, an algorithm brought out to adaptively utilise the generative model to augment the training set [46]. More than that, StyleGAN has already been used to detect brain CT motion artefacts [47]. Similarly, the StyleGAN2-ADA [20] framework proposes an adaptive discriminator augmentation mechanism to augment the dataset while stabilising the training process with limited data.

2.5 Latent Space

Latent space is a concept in deep learning. It simply means a representation of compressed data. It is important since people can learn patterns and structural similarities by analysing them. With the popularity of GANs, latent space has become more interesting for people to interpret [48], optimise [49] and explore. As the state-of-the-art StyleGAN proposed, manipulation of latent space has become unprecedently prevalent [50][51] since a small vector change in latent space can affect the feature in a high-grained generated picture.

Chapter 3

Methodologies

This section discusses how the raw image data transfers to train the generator and the discriminator and how the experiment set up. For data training, first, data pre-processing is achieved by filtering out non-merged galaxy images and adjusting the size. Then, pre-processed image data will go through adaptive discriminator augmentation. The result will be the input of the discriminator with the fake images generated by the generator. The generator and the discriminator both use progressive growing, in which AdaIN layers are applied. The generator uses a noise mapping network to transfer the noise vector to a more disentangled representation. The overall pipeline is as follows:

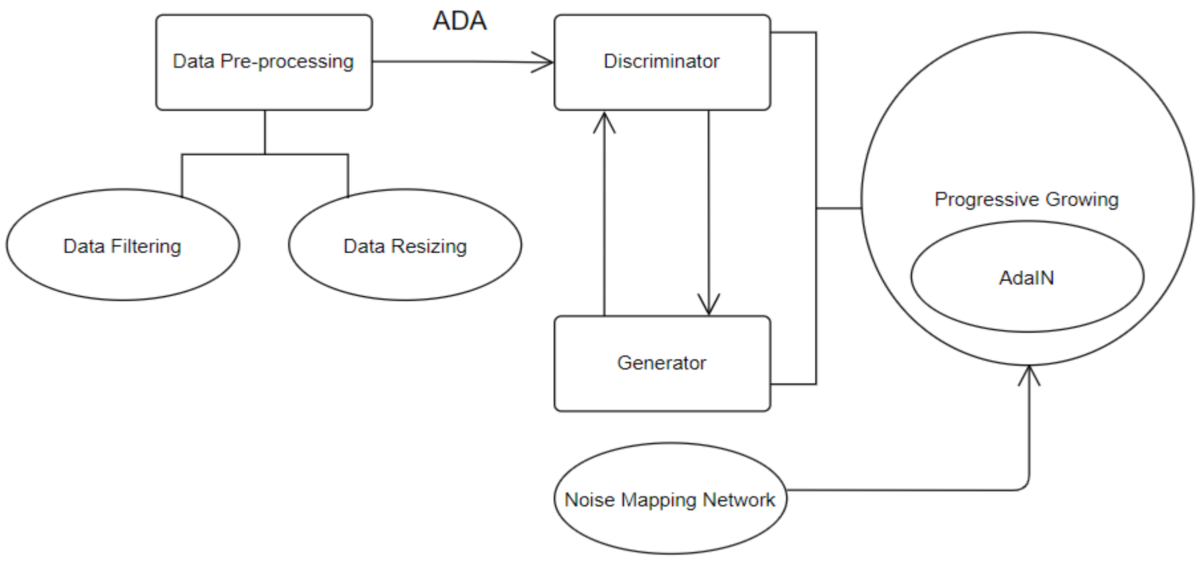


Figure 3.1: Implementation pipeline

3.1 Data Pre-processing

In the training dataset, 43,780 images are classified and labelled by experts rather than amateurs due to certainty problem, all with a resolution of 64x64. Then, 2851 images of the merged galaxy are filtered out. Those images are chosen as the training set for my project to simulate limited data problem in deep learning. PNG format conversion is performed to suit the framework's need.

The pseudo code is as follows:

Algorithm 1 Select Merged Images

```
data = load(”GZ1Expert-MergerMaster.csv”)
data = data[2 :, :]
mergerIdList = [ ]
for i in range data do
    if data[i, 20] == ”1” then
        mergerIdList.append(data[i, 1])
    end if
end for
path = imageLocation
newPath = imageDestination
for m in range lengthof(mergerIdList) do
    for file in all files in path do
        if mergerIdList[m]+”.tiff” == file.name then
            copy the file to the newpath
            Break
        end if
    end for
end for
```

Firstly, a CSV file that contains all the image information is loaded. Then, the data from the second row to the last is selected because the first row is the label. After that, a for loop is used to extract all the image data whose ”Merger” label is 1 and form a merger list. Finally, for each image in this merger list, the name of the image is compared with all the images from the path directory. If the image is found, copy the image from the path to the new path, then break into the next loop.

Here, after calculation, a conclusion can be drawn that merged galaxies only account

	elliptical and spiral images	**merged galaxy images**	**total images**
Image number	41199	2581	43780
Proportion	94.1%	5.9%	100%

Table 3.1: Merged galaxy images data proportion

for 5.9% of the total images labelled by experts. However, according to the first edition of GZ1, there are 900k amateur-labelled images, and I only calculate the proportion of merged galaxy images in a subset of these images. It is not difficult to infer that in the entire galaxy data set, according to this ratio, there will be about 54,000 pictures of merged galaxies.

In 2015, the ImageNet data set used by ResNet [52] contained about 100,000 images. In 2017, Google used 300M images to study deep learning problem in computer vision [53]. With the optimization of GPU, CPU and other hardware, the size of the training set is increasing, which proves the necessity of using GANs for data augmentation.

Image resizing is accomplished after filtering out the merged galaxy images by first copying the origin training set folder several times, then manipulating each folder. The manipulation pseudo code is as follows:

Algorithm 2 Image Resolution Resize

```
1: $path$ = new folder
2: $m = 0$
3: $size$ = imageNewSize
4: **for** file in folder **do**
5:     $image = Image.open(file)$
6:     $resizedImage = image.resize$
7:     save $resizedImage$ with PNG suffix
8:     $m = m + 1$
9: **end for**
```

First, rename the folder. Second, create two variables, m for counting and $size$ for resolution of the new images. For example, 512,512 is assigned to $size$ if I manage to create a training set with all images with 512x512 resolution. In the for loop, an image resizing operation is carried out. Then, resized images are saved with PNG suffix, as the framework requires. The counting variable is also added by 1 in the loop, which is helpful in error checking.

3.2 Progressive Growing

Progressive growing is one of the reasons that StyleGAN2 can generate super realistic images—starting with a task of generating a 4x4 image with 16 pixels to 1024x1024 in the end. The generator will generate 4x4 images, and the discriminator needs to evaluate whether they are real or fake. Since the input size of fake images is 4x4, the real images need to be down-sampled to 4x4. The next step of progressive growing is to double the size of the generated image and the size of the image that feeds into the discriminator. Convolutional layers are added to both generator and discriminator between those 4x4, 8x8 blocks to help to do so. The real images are now down-sampled to 8x8, as the input of the discriminator along with the 8x8 image created by the generator. Repeat this process several steps. Eventually, the network can generate and discriminate 1024x1024 images.

However, there are learned parameters that constantly change along the process. The way of generating 8x8 images above is not taking a 4x4 image, using nearest neighbour interpolation [54] to up-sample the image, but a combination of up-sampling and relying on learned parameters. At the beginning of a step, for example, 4x4 to 8x8. StyleGAN2 does 99% up-sampling and 1% of taking this up-sampled image into a convolutional layer that produces an 8x8 resolution image so that some learned parameters can be used along the process. Over time, this alpha decreases from 99% to 50% then to 0, so that the image looks more realistic rather than just up-sampling from nearest neighbours. Also, the network now relies on these learned parameters entirely. Similarly, but opposite direction, the discriminator down-samples the image 100% at the start of a step and then relies more and more on learned parameters contained in the convolutional layers.

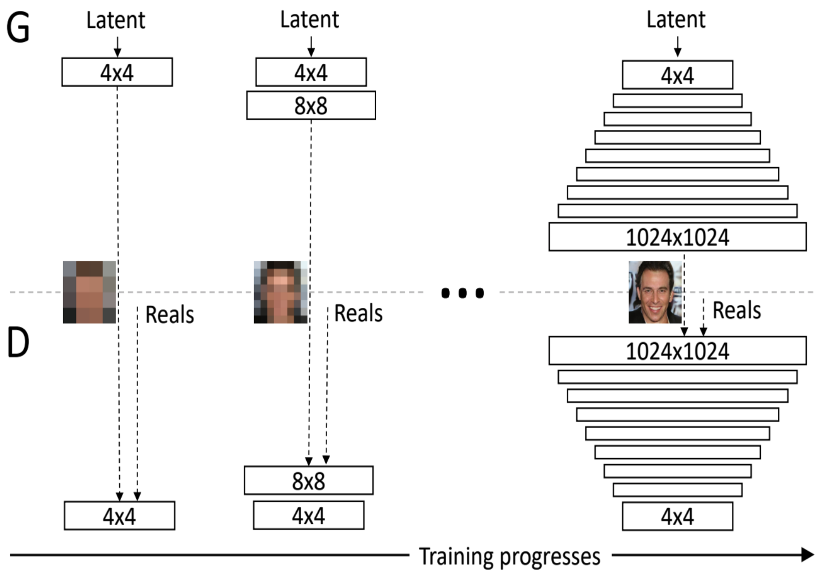


Figure 3.2: Progressive Growing

3.3 Noise Mapping Network

A noise mapping network is simply a network containing eight fully connect layers with activation layers in between, aiming to transfer the input noise z to an intermediate noise vector w, with a more disentangled representation. The motivation for doing this is that the first noise vector z has a normal prior, which means that all the values are drawn from a normal distribution. It is challenging for this vector to match the density distribution of output features. As a result, it will twist itself to map onto those desired features. StyleGAN reduces the chance of styles being too closely correlated by doing so. StyleGAN also changes the position of the noise vector input. Instead of having a noise vector z before the first 4x4 block like classic GAN, the intermediate noise vector w that comes from the noise mapping network is regarded as input into all of those 4x4, 8x8 ... blocks, as a part of the progressive growing. The authors of StyleGAN have found no noticeable difference between them, so rather than having noise at the very beginning, the model starts with a constant value, with 512 channels.

3.4 AdaIN

Adaptive instance normalisation is added in each block in progressive growing after each convolutional layer. Unlike batch normalisation, instance normalisation normalises the convolution outputs for one example in one channel. Every value in that instance will be normalised to a mean of 0 and a standard deviation of 1. The adaptive part then comes in to apply adaptive styles to these normalised values. The intermediate noise w, as mentioned earlier, results from an initial noise vector going through a multi-layer perceptron. Then, this intermediate noise will affect every block as long as there is an AdaIN layer. The way it controls the style is not directly input. However, it goes through learned parameters, two fully connected layers are used in StyleGAN2. By doing so, a scale parameter y_s and a bias/shift parameter y_b are produced and as inputs to the AdaIN layer, as Equation (3.1):

$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}, \tag{3.1}$$

where x_i stands for a single instance, $\sigma(x_i)$ is the standard deviation of the instance, $\mu(x_i)$ denotes the mean value of the instance. Adaptive styles is achieved because w can change. In other words, these scale and bias parameters extracted can change. The style can be boiled down to just re-scaling and re-shifting values to a certain range, mean, and standard deviation from this perspective.

The generator point of view is made up of lots of blocks where the earlier blocks like conv4x4 or conv8x8, aligned with coarser features and latter blocks with finer details. Since w is added into all of these blocks, it will be coarser details that are changed by this intermediate noise if w is added into those earlier blocks, and it will be finer details that are affected by w if it is added into later blocks. With instance normalisation occurs in every block, the output statistics of each block will always be re-normalised to a mean of 0, standard deviation of 1. This allows each block control the style only at that block because statistics will be overwritten by the next AdaIN that normalises those previous outputs. By using multiple initial noise vectors, z_1, z_2, z_3 ..., multiple corresponding intermediate noise vectors w_1, w_2, w_3 ..., can be created, thus different styles can displayed in one image if different noise vectors interpolate different blocks.

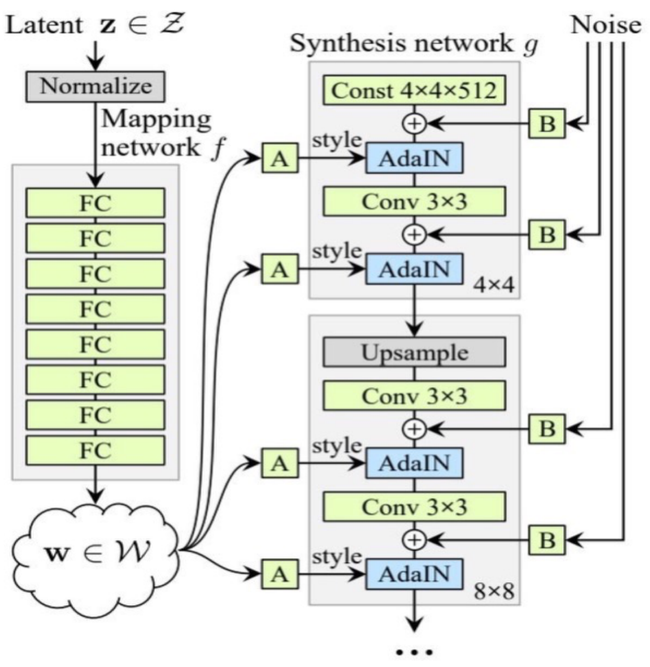


Figure 3.3: StyleGAN2 structure

3.5 ADA

For almost all situations in deep learning, data augmentation is the best way to solve the limited data problem. However, this method is undesirable in GAN since the generator will learn to produce the augmented distribution rather than real ones, which is denoted as ”leaking” issues in the paper [20].

Before introducing adaptive discriminator augmentation mechanism, two definitions should be presented: invertible and non-invertible transformation. In terms of the most common way in data augmentation: rotation. Random rotations chosen uniformly from 0, 90, 180, 270 degrees are not invertible since it is impossible for the discriminator to distinguish the orientation of the augmented image. As shown in Figure 3.4, generated and real images all have 25% chance to be either rotated 0, 90, 180, or 270 degrees after augmentation, even if all images generated by the generator are upside-down.

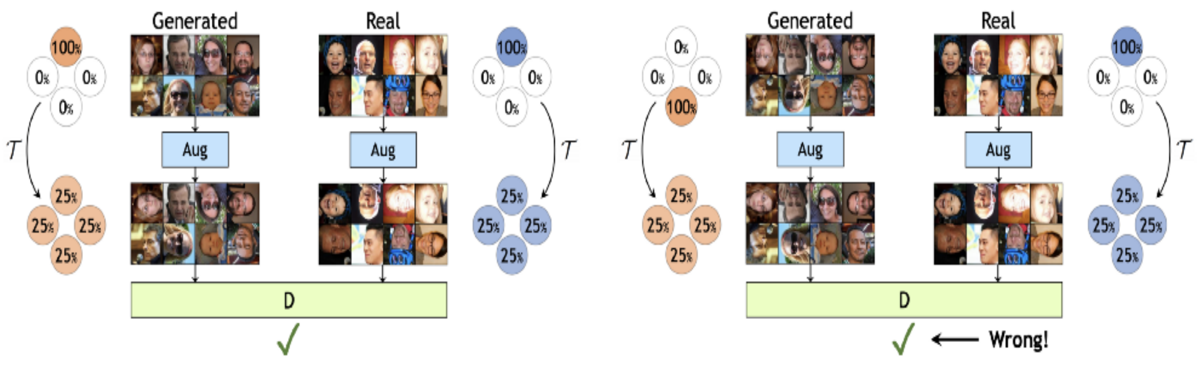


Figure 3.4: Non-invertible example: 90 degree rotations

However, suppose this rotation operation happens under a probability p as Figure 3.5. Things will change because now, when the generator tries to augment fake images, the augmented distributions do not match the real augmented distribution. For example, suppose p is set to 0.8, and all the images generated by the generator are upside-down. The images also perform a uniform rotation. As a result, augmented generated images have 40 percent upside down. However, the real augmented images only have 20 percent upside down. In this way, the generator is forced to match the distribution to the real distribution before augmentation.

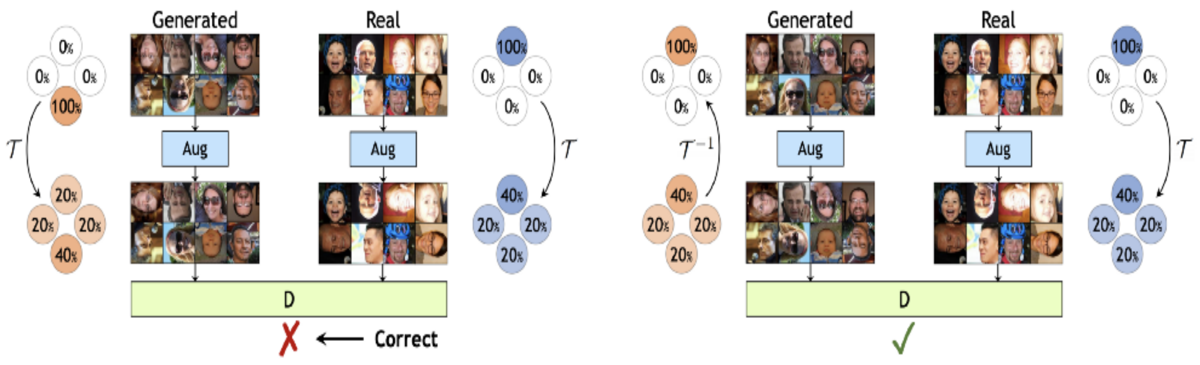


Figure 3.5: Invertible example: 90 degree rotations with uneven probability (p=0.8)

Theoretically, as the paper points out, if the augmentation operation T is invertible, there exists one and only one initial distribution x for the augmented distribution T_x, without "leaking" issues in x. However, in practice, due to numerous reasons such as limitations of finite sampling, very high p leads to leaking of augmentations in the generated images. The paper also shows that the optimal value of p is susceptible to the size of the dataset.

The standard way of quantifying overfitting is to use a validation set and compare its behaviour to the training set. The paper discovers that the validation set behaves increasingly like the generated samples after overfitting. For smaller datasets, this happens even earlier. The authors propose two heuristics to measure overfitting, as follows:

$$r_v = \frac{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{validation}}]}{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{generated}}]} \qquad r_t = \mathbb{E}[\text{sign}(D_{\text{train}})] \tag{3.2}$$

In equation 3.2, D_{train}, $D_{validation}$, and $D_{generated}$ stand for the discriminator outputs for training set, validation set, and generated images respectively. E is the mean value over four mini-batches, as the experiment took in the paper. The first heuristic r_v measures the similarity between the validation set and training set and the similarity between the validation set and generated set. If the training set and validation set are the same, the numerator will be 0, so as the r_v, which means no overfitting. Contrarily, if the validation set behaves the same as the generated set, the numerator and denominator will be the same. Under this circumstance, r_v is 1, which means completely overfitting.

Since the paper assumes that there is already a small validation dataset, it is impossible to calculate r_v. Hence, the authors propose a r_t heuristic to estimate the portion of the training set that gets positive discriminator outputs. This has been proved to be less sensitive to other hyperparameters and the chosen target value than directly looking at $E(D_{train})$. The adaptive part is reflected here: if r_t is very high, then the model needs to augment more to increase p. If r_t is too low, then the model needs to augment less so that the p will decrease. In such a way, augmentation is achieved by transforming non-invertible image augmentations into invertible transformations with a probability p, and p can be altered by the heuristic.

3.6 Experiment Setup

Given the training set, different resolution images can be used to train the network. Since the resolution of the image I received is 64x64, which is very low, the convergence speed will be very fast, according to empiricism. In this case, increasing the resolution of the same image by nearest filter is an ideal way to achieve the comparison by using different resolution images as the training set. This is done in data pre-processing. With different resolution image data, different termination conditions can be set to train the network. Due to time limitation, I chose 5,000 steps for semester 1 and 15,000 kimg for semester 2. For resolution, 64x, 128x, 256x, 512x images are compared in semester 1, and 64x64, 128x, 512x images are compared in semester 2.

The training of PyTorch version of StyleGAN2 will take 150,000 steps, and the FID score will be calculated every 5,000 steps by default. However, the FID score dropped rapidly, so another experiment to calculate the FID every 200 steps is set up. The calculation method is to create two folders every 200 or 5000 training steps to store real and fake images, and each folder contains four batches, a total of 12800 real or fake pictures. Due to the insufficient real image data set, the program will use the cycle function to fill in real data distribution. By comparing the two folder data distribution, the score can be calculated.

For StyleGAN2-ADA, a comparative study is proposed by whether ADA is used. After carefully reading the readme file of the framework and precise calculation, I set the total number of training steps to 15,000 kimg since an undergraduate student can use only 1 GPU. Due to the GPU memory problem, the parameters used for training cannot be adjusted to the best state. Here a unified configuration is used to reproduce the results paper for FFHQ and LSUN Cat at 256x256. Parameter **mirror** is set to 1 to amplify the dataset with x-flips, which is often beneficial with and without ADA, according to the framework. As a result, the only variable for the two models is whether ADA is used. The probability p is set at 0.7 by default.

Chapter 4

Implementation

4.1 Overall Implementation

My implementation includes interpolation between fake images and performing vector arithmetic with fake images. Both the tasks are finished using Python. I write scripts on the remote desktop of the University of Nottingham and send the scripts to the school GPU cluster by Slurm [55] jobs. Code is written with Python version of 3.6.8, importing standard deep learning modules like TensorFlow, NumPy, dnnlib, and tflib, image modules like Image, imageio, and system modules os, sys, Path. I also allow bar progress checking and the command line argument for script running, as I import tqdm and argparse modules.

4.2 Interpolation

Interpolation is implemented by creating a path between two points generated with random seeds, then generating fake images along this path. In this project, I use a linear or uniform interpolation. However, more sophisticated interpolation function can be taken into account, such as spherical linear interpolation [56].

The code for interpolation is based on **projector.py** provided by StyleGAN2-ADA framework, which helps users find a matching vector for a given image file. Inspired by that, interpolation is achieved by first loading the GAN model that has already been trained. Then, random seeds are selected as inputs to generate two fake images. Finally, the output can be produced by performing an animation using linear interpolation and save the animation using the 'gif' suffix.

The pseudo-code for interpolation is as follows:

Algorithm 3 Interpolation

```
1: Create a interpolator object
2: $Gs = Loadmodel$
3: $results\_size = 512$
4: set model
5: $seed1 = \text{randomSeed1}$
6: $seed2 = \text{randomSeed2}$
7: $image, latent\_vector1 = \text{generate\_image\_random}(seed1)$
8: $image1 = image.\text{resize}(results\_size)$
9: $image, latent\_vector2 = \text{generate\_image\_random}(seed2)$
10: $image2 = image.\text{resize}(results\_size)$
11: $\text{make\_latent\_interpolation\_animation}(image1, latent\_vector1, image2, latent\_vector2)$
```

The user can alter the random seeds of two fake images. Here I use 42 and 1234 to create fake images and randomly cut a frame in the middle of the animation. The results are shown below. The model used here was trained using 512 resolution images, and ADA was used.

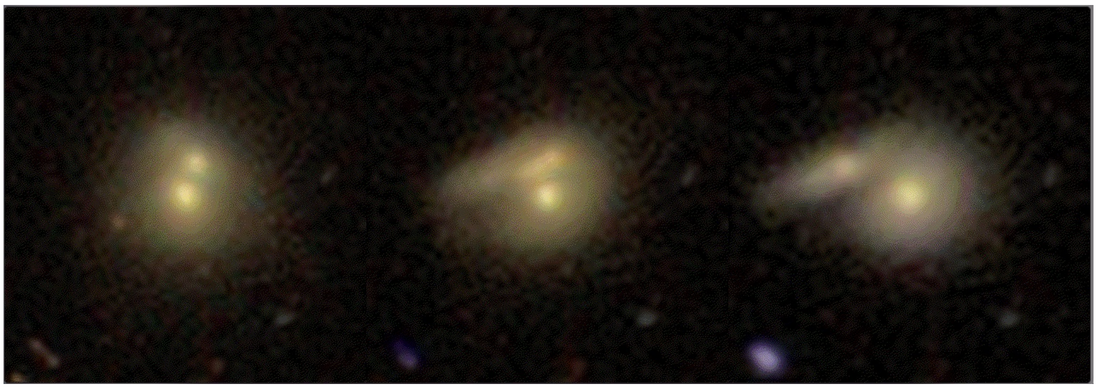


Figure 4.1: Interpolation

In terms of animation, the number of interpolations is preset as 200 and also can be changed by the user. Linear interpolation is carried out by using a weighted alpha.

For each frame in the animation, the current vector is calculated by latent vector one multiply by alpha plus latent vector two multiply by one minus alpha. Since the number of interpolations is 200, the step size is set to 1/200 (one over the number of interpolations). So for the first frame, alpha is 1/200, second frame 2/200 and so on. The image on the top left is generated by seed 42, the image on the top right is generated by seed 1234. The middle part is an animation that transforms from the left to the right image, but I can only take a screenshot here. The animation will be displayed in the demo video.

4.3 Vector Arithmetic

In a similar way as interpolation, vector arithmetic is also a way to manipulate latent vectors. Ideally, the vector arithmetic should initially let the generator randomly generate many fake images and then allow users to select the images they are interested in, eventually performing latent vector arithmetic. However, for convenience, I randomly generate three images with random seeds 42, 1234, 321. The latent vector arithmetic is calculated as the latent vector one minus the latent vector two, then plus the latent vector three by default, yet user can alter this formula in **vectorArithmetic.py**.

The pseudo-code for vector arithmetic is as Algorithm 4, where **vector_arithmetic()** includes simply vector adding/subtracting, and sub-plotting the origin images and result image after vector arithmetic.

The output is shown in Figure 4.2. The model is used the same as interpolation. The vector arithmetic is used to operate human face images in the StyleGAN2 paper. However, my experiment proves that the calculation of latent vector does not map well to the image of galaxies as the characteristics of the image cannot be well recognized.

Algorithm 4 Interpolation

```
1: Create a vector arithmetic object
2: $Gs = Loadmodel$
3: $results\_size = 512$
4: set model
5: $seed1$ = randomSeed1
6: $seed2$ = randomSeed2
7: $seed3$ = randomSeed3
8: $image, latent\_vector1$ = generate_image_random($seed1$)
9: $image1 = image$.resize($results\_size$)
10: $image, latent\_vector2$ = generate_image_random($seed2$)
11: $image2 = image$.resize($results\_size$)
12: $image, latent\_vector3$ = generate_image_random($seed3$)
13: $image3 = image$.resize($results\_size$)
14: vector_arithmetic($image1$,$latent\_vector1$,$image2$,$latent\_vector2$,$image3$,$latent\_vector3$)
```

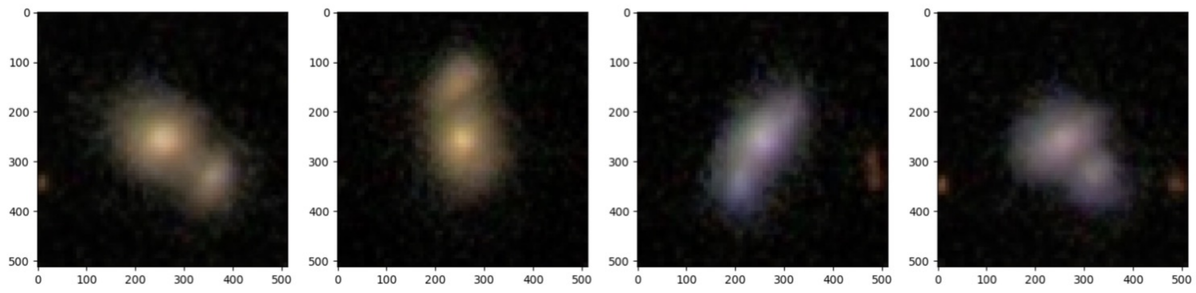


Figure 4.2: Vector Arithmetic

Chapter 5

Evaluation Design

5.1 Evaluation Metrics

In this session, two mainstream metrics (Inception Score and Fréchet Inception Distance) and a newly proposed metric (Kernel Inception Distance) of evaluating the generated images will be introduced by providing and explaining the formula. Then, the result of the first semester will be presented and discussed, followed by the second semester's experiments results.

5.1.1 Inception Score (IS)

Inception Score [5] measures the quality of generated images by their fidelity and diversity. Using the Inception V3 model [57], the desire is to produce various outputs, which means that the classes of images are evenly distributed. This can be measured by calculating marginal distribution P(y), where y is the label or the class. In addition, the other expectation is that given a certain picture, the class can be determined easily. This is measured by calculating the conditional distribution $P(y \mid X)$, where X is the generated data or the generated image, in this case. As a result, a KL divergence is used to compute

IS following Equation (5.1):

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right), \tag{5.1}$$

where E denotes the summation of all the images and the average of all the classes, exp is just an exponent, so that the score does not look that big. However, the Inception Score does have some limitations. Due to Barratt et al. [58], the weakness of IS is manifested in the following aspects. This measurement only focuses on generating images; there is no comparison to the real data. If the goal is to generate a specific type of images that do not present in the classifier's training set before, the IS will remain low. Poor quality images can sometimes get high IS, and so on. Due to this, the IS has been used less and less nowadays. Since merged galaxy images do not involve multi-class image variation, and the training set is not ImageNet, IS will not be considered as a metric in this project.

5.1.2 Fréchet Inception Distance (FID)

To overcome one of the disadvantages of the Inception Score that it does not compare the fake and real images, Fréchet Inception Distance is proposed [59]. FID is calculated by comparing real and fake data distribution at some intermediate layers when using the Inception network. Real and fake data distributions are modelled using multivariate Gaussian distribution with corresponding mean μ and covariance Σ. The FID score between real images and fake images could be calculated as Equation (5.2):

$$\mathtt{FID}(x,g) = ||\mu_x - \mu_g||_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x\Sigma_g)^{\frac{1}{2}}), \tag{5.2}$$

where Tr sums up all the diagonal elements of the matrix. μ_x is the mean of the real distribution, μ_g is the mean of the generated distribution, Σ_x and Σ_g are the covariance matrix of real and generated data distributions, respectively.

Comparing to IS, FID score is more robust. First, the training set can be different from Inception V3. Second, FID is calculated more reasonably. By observing how IS is calculated, we can infer that the generated data distribution is compared to the ImageNet training set. However, FID makes a comparison by training data and sampled data. As a result, FID is suitable for being used in this project.

5.1.3 Kernel Inception Distance (KID)

Proposed by Kynkäänniem and his colleagues in 2018 [60], Kernel Inception Distance works very similar to FID. Instead of computing the Wasserstein-2 distance between two Gaussian distributions, it calculates the squared maximum mean discrepancy (MMD) between the features of real and generated images. The following formula calculates the KID:

$$\begin{aligned}\mathrm{KID}(p_\mathrm{r}, p_\mathrm{g}) =& \mathbb{E}_{\mathbf{x}_\mathrm{r}, \mathbf{x}'_\mathrm{r} \sim p_\mathrm{r}}[k(\mathbf{x}_\mathrm{r}, \mathbf{x}'_\mathrm{r})] \\ &+ \mathbb{E}_{\mathbf{x}_\mathrm{g}, \mathbf{x}'_\mathrm{g} \sim p_\mathrm{g}}[k(\mathbf{x}_\mathrm{g}, \mathbf{x}'_\mathrm{g})] \\ &- 2\mathbb{E}_{\mathbf{x}_\mathrm{r} \sim p_\mathrm{r}, \mathbf{x}_\mathrm{g} \sim p_\mathrm{g}}[k(\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{g})]\end{aligned} \tag{5.3}$$

where k denotes the polynomial kernel function $k(x, x^{'}) = (\frac{1}{k}x^{'}x^{T} + 1)^{3}$, x_r and x_g represent real and generated data distribution respectively. E as mention earlier in the IS part, is the average value of all classes.

Since FID and KID both calculate the distance between real and fake data distributions, KID can also be used as a metric in this project. Also, because of the above reason, the lower scores of FID and KID, the higher the quality of the fake pictures generated.

In general, in the three measurement methods mentioned above, IS does not adapt to models other than the Inception V3 model. Therefore, this project will use the other two metrics (FID and KID) to test the quality of generated images.

5.2 Experiment Result

5.2.1 First Semester Result

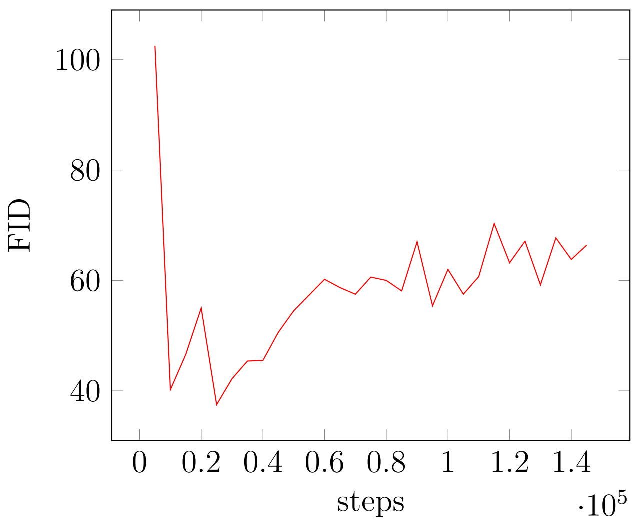
In the first semester, using StyleGAN2 to generate fake images and using only FID to measure different models performance is achieved.

Experiments show that if the FID score is calculated every 5000 steps by default, after 10000 steps, the FID score will fluctuate between 40 and 65, except for the 25000 step when the score drops to 37.5. Although the contingency of the experiment cannot be excluded, a conclusion can be drawn that the convergence has started at step 10000 in terms of this experiment according to the statistic. In order to more accurately determine when the convergence begins, calculation of FID every 200 steps is needed, as mentioned in Chapter 3. The experiment data is shown in Table 5.1 and Figure 5.1.

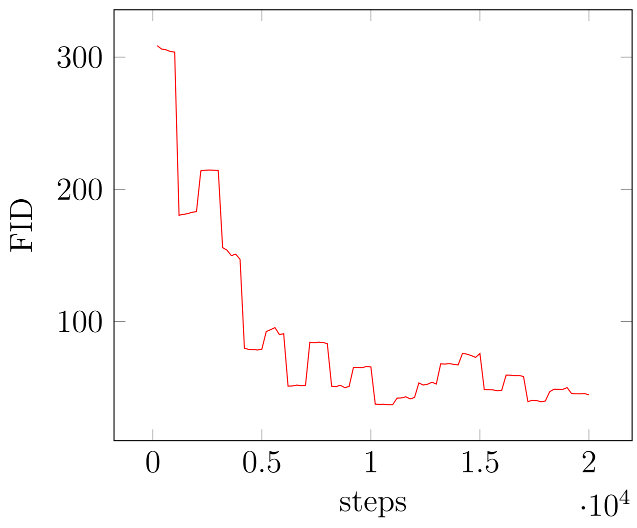
Steps Count	**FID score**
5000	102.5
10000	40.2
15000	46.6
20000	55.0
25000	37.5
30000	42.2
35000	45.4
40000	45.5
45000	50.6
50000	54.5

Table 5.1: The FID score of using 64 x 64 images as the training set until step 50000

The experiment shows that the FID of the 64x64 resolution merged galaxy image is already less than 65 at step 6200, and has been fluctuating afterwards, which indicates that it has indeed begun to converge. The FID reached its lowest point between 10000 and 25000 steps during training and then started to overfit. Therefore, the model is saved when it trained for 10000 steps so that it can be utilised to generate more fake merged galaxy pictures with 64x64 for further use.



(a) FID is calculated every 5000 steps



(b) FID is calculated every 200 steps

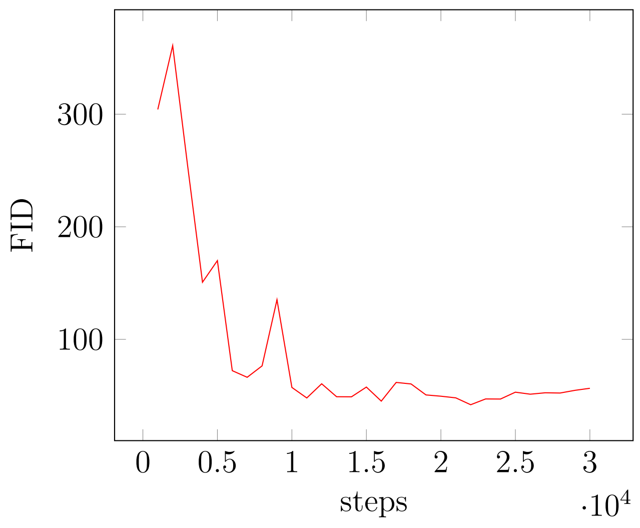
Figure 5.1: FID score with 64x64 resolution images

The experimental results of using different resolution images are shown in Figure 5.2. Experiments show that when three different resolution pictures are used as training sets for training, the FID scores at the final convergence are approximately the same, all between 40-65. If the final convergence score is set between 40-65 points, according to the chart, the convergence of 128x128 resolution image starts after 7000-8000 steps, the 256x256 resolution image starts after 8000-9000 steps, and 512x512 starts between 10000-11000 steps, which verifies the empiricism, that is, the higher the resolution of the image, the more convergence time it requires.

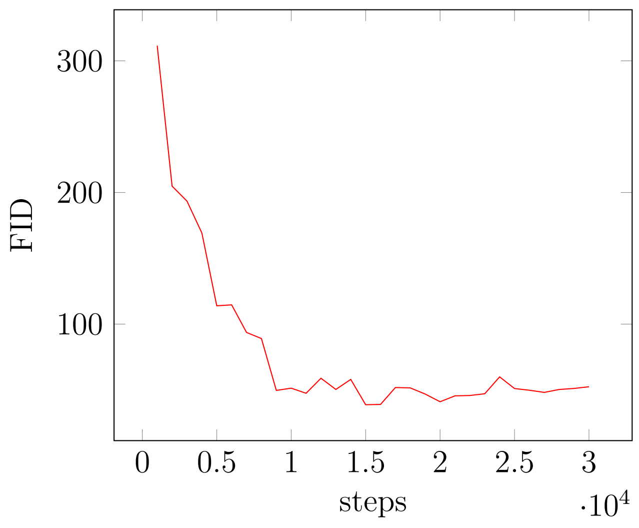
5.2.2 Second Semester Result

The experiments results are shown in Figure 5.4.

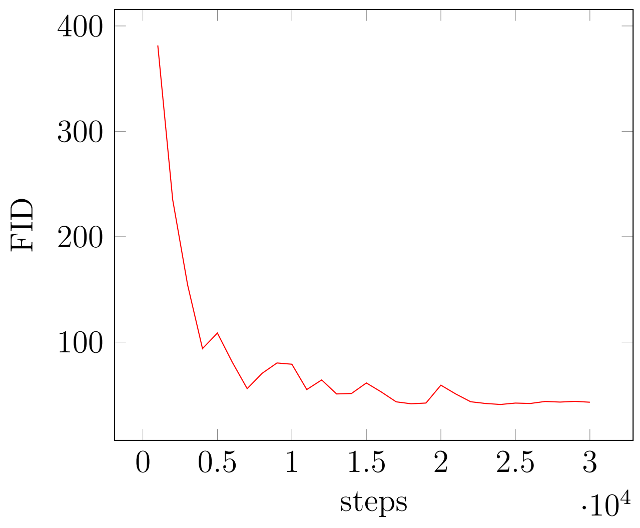
The figures show that using ADA can indeed avoid overfitting to the training set, as the paper states. With 64× and 128× images, FID and KID drop vertically in the beginning, then there is no change until the end, and the curve presents a 90-degree turn. Without ADA, overfitting is shown, but the magnitude of overfitting is not significant with 64× and 128× images, and the curve is almost consistent with the model using ADA. Nevertheless, with 512× images, the convergence speed is slower than other resolutions.



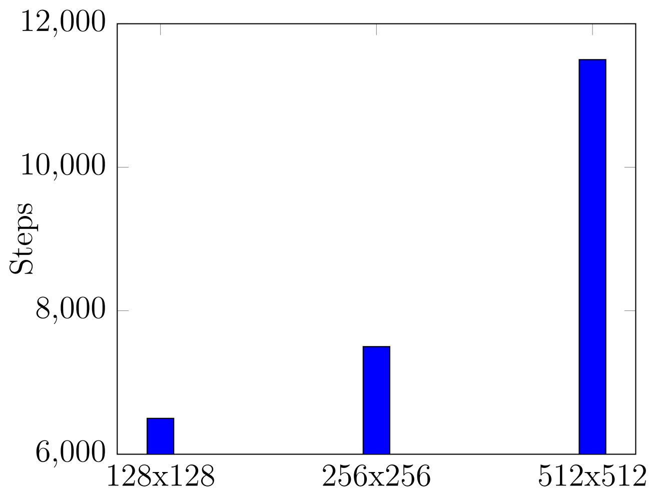
(a) FID with 128x128 resolution images



(b) FFID with 256x256 resolution images



(c) FID with 512x512 resolution images



(d) Comparison of the final convergence time of the training set with different resolution pictures

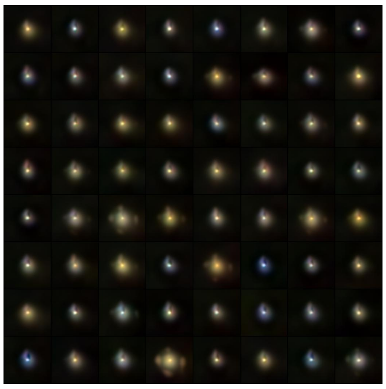
Figure 5.2: Using different resolution images as the training set, their FID scores change over time

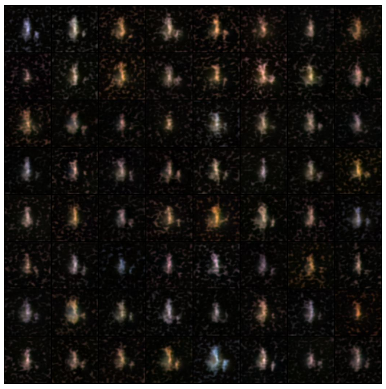
Although the final FID and KID scores are similar to other resolutions when using ADA, the figures show that they will not converge until about 10,000 kimg. Without ADA, the situation is much worse than other resolutions. Starting from 5000 kimg, the fluctuations begin and continue to the end. The degree of fluctuations is more significant than other resolutions, and overfitting impacts both FID and KID. According to the results of my experiment, without using ADA, the higher the resolution of the training set, the more serious the overfitting, as can be seen from Figure 5.4 (c) and (d). The final FID and KID scores increase as the resolution of the training set image increases.

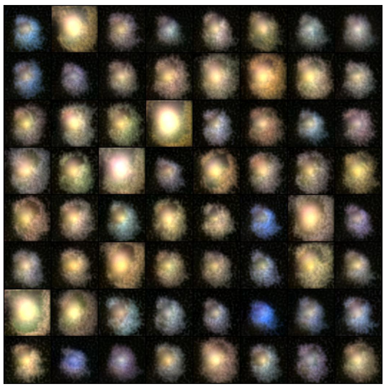
Training time does not increase linearly with the increase in resolution. With ADA, training time is increased by nearly 25 percent compared to without using ADA.

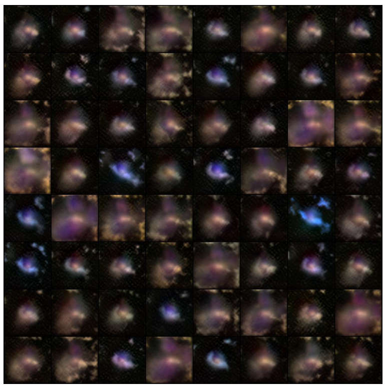
Name of models	**Training time**
64×	4 days 22 hours
128×	5 days 19 hours
512×	13 days 1 hours
64× with ADA	5 days 19 hours
128× with ADA	6 days 9 hours
512× with ADA	16 days 0 hours

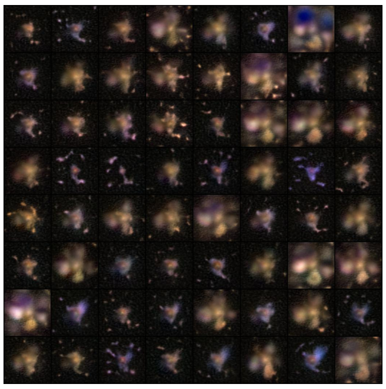
Table 5.2: Training time for each model, all with steps 15000 kimg

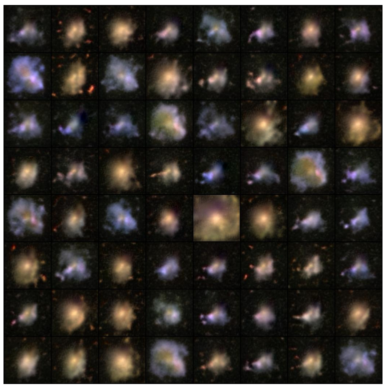

(a) 1000steps.

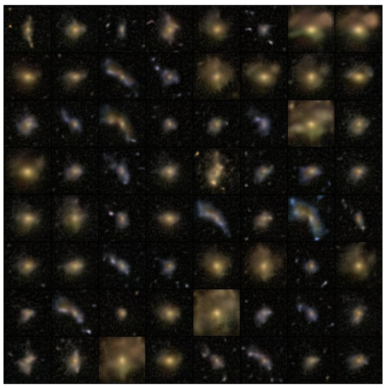

(b) 2000steps.

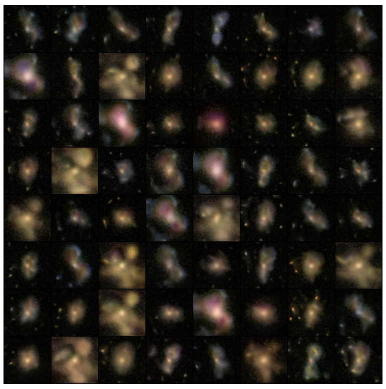

(c) 3000steps.

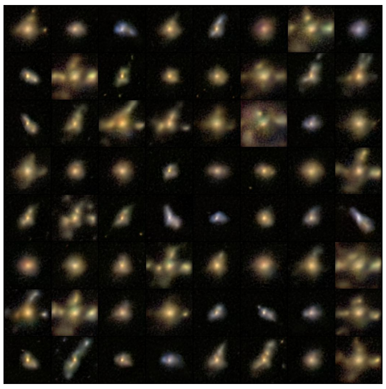

(d) 4000steps.

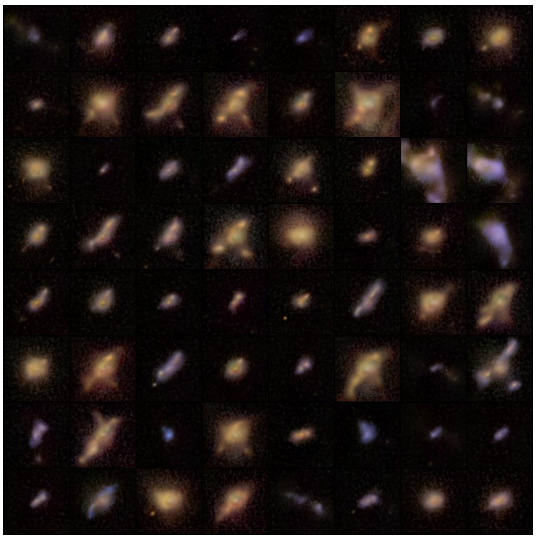

(e) 5000steps.


(f) 6000steps.


(g) 7000steps.


(h) 8000steps.


(i) 9000steps.


(j) 10000steps.

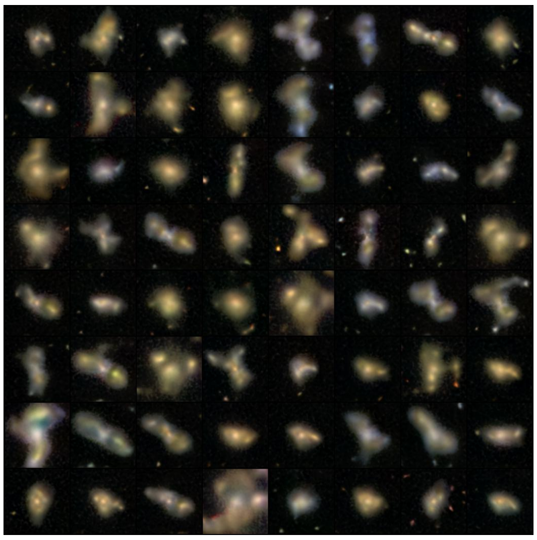
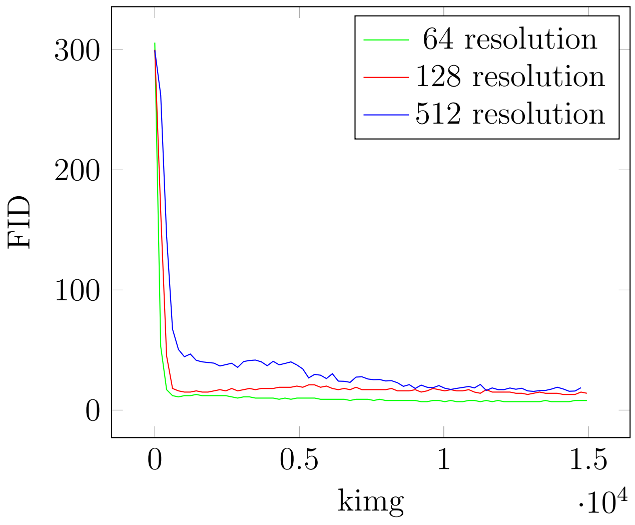
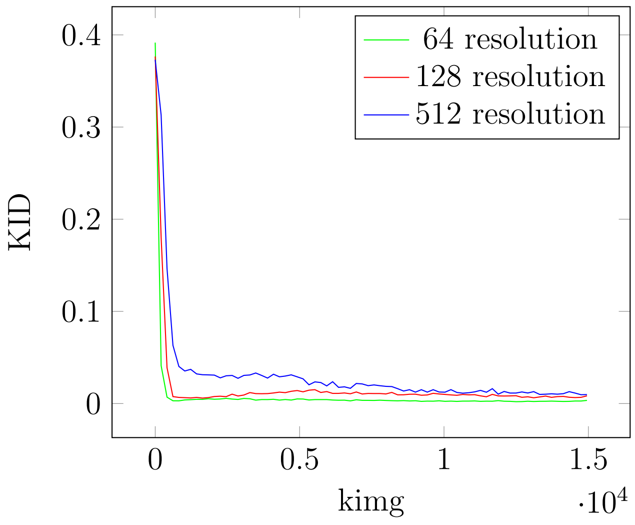

(k) 11000steps.

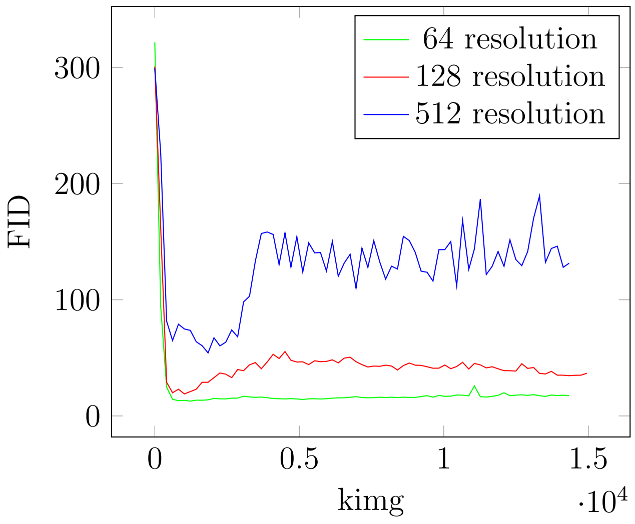
Figure 5.3: Images generated during model training by first framework, using 64x64 image as training set



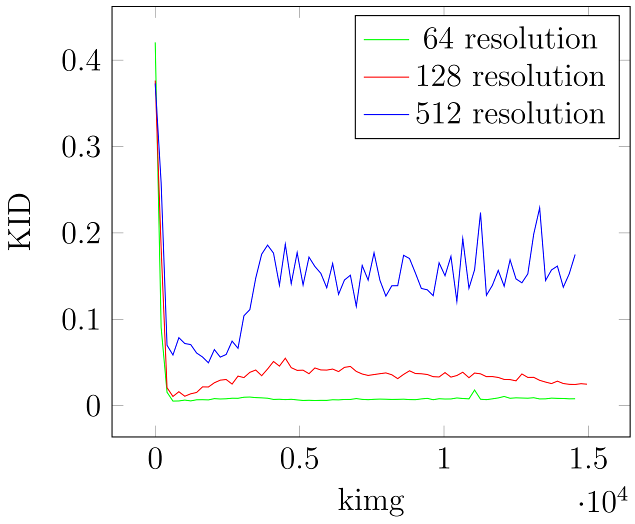
(a) model with ADA FID scores



(b) model with ADA KID scores



(c) model without ADA FID scores



(d) model without ADA KID scores

Figure 5.4: Comparison with different models using different resolution images as training set

Chapter 6

Summary and Reflections

6.1 Project management

The project is designed to run on a weekly basis, with easier tasks taking a week, more challenging tasks taking two weeks or more to complete. The reasons for adopting this schedule style are first, to facilitate schedule management, to compare the schedule and plan at the end of the week, and second, to ensure that enough time is saved to make final adjustments.

At the beginning of the project, the progress was designed as Figure 7.1. However, due to my inadequate literature review, this project's aim and objective were misunderstood, and the workload was significantly underestimated. The project should focus more on image generation rather than classification, and the discriminator should determine the authenticity of the generated image rather than classifying elliptical or spiral galaxies.

Therefore, a new plan was formulated in the middle of the first semester, as shown in Figure 7.2. The revised project plan, generally speaking, with Christmas as the boundary, the image generation and data evaluation of two different GANs need to be completed, respectively. In the first semester, and easier GAN framework is used to get familiar with the process, while in the second semester, a more difficult GAN framework will be applied to generate fake images. Then a comparison of two GANs will be proposed.

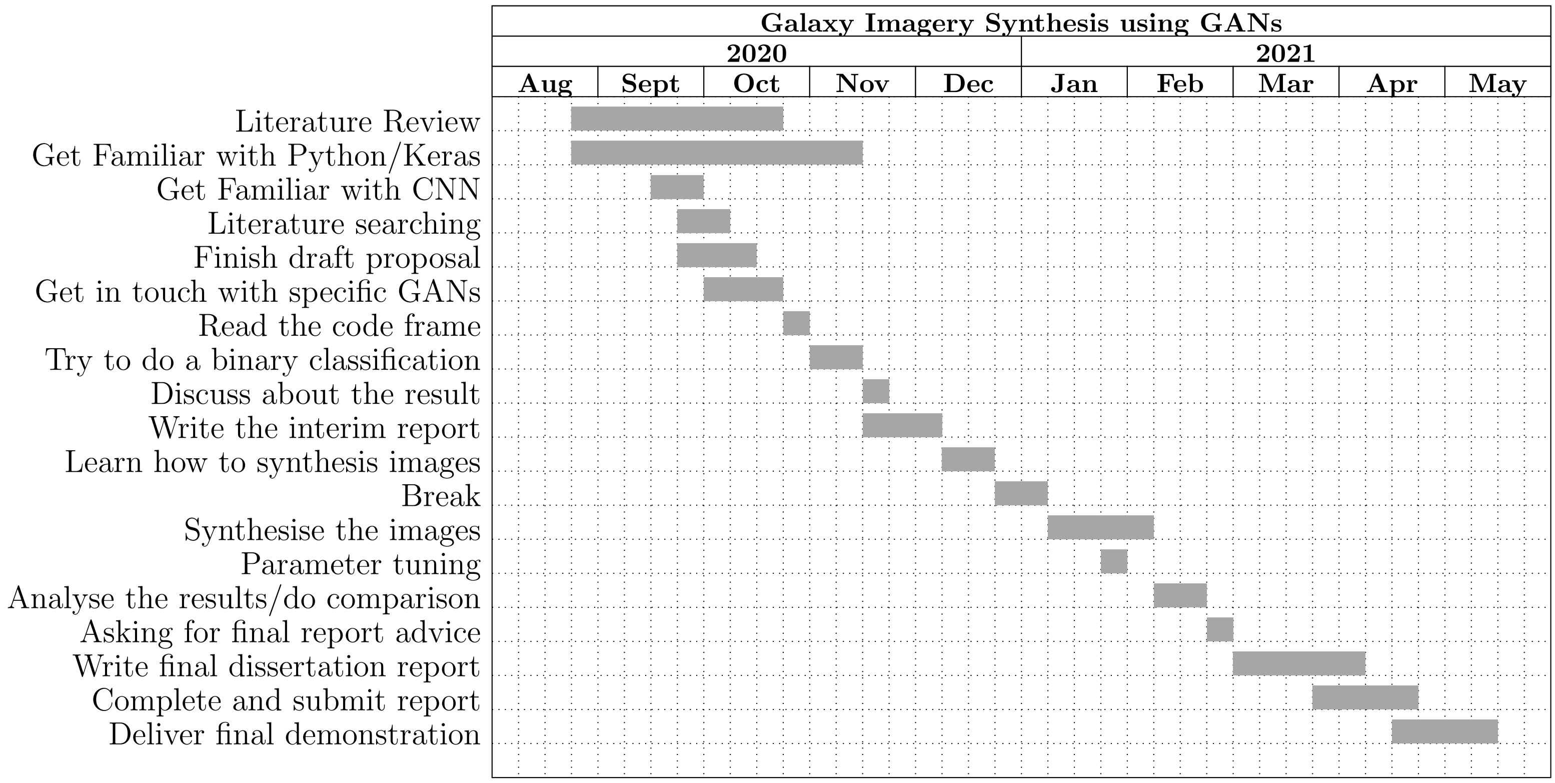


Figure 6.1: Original Time Plan

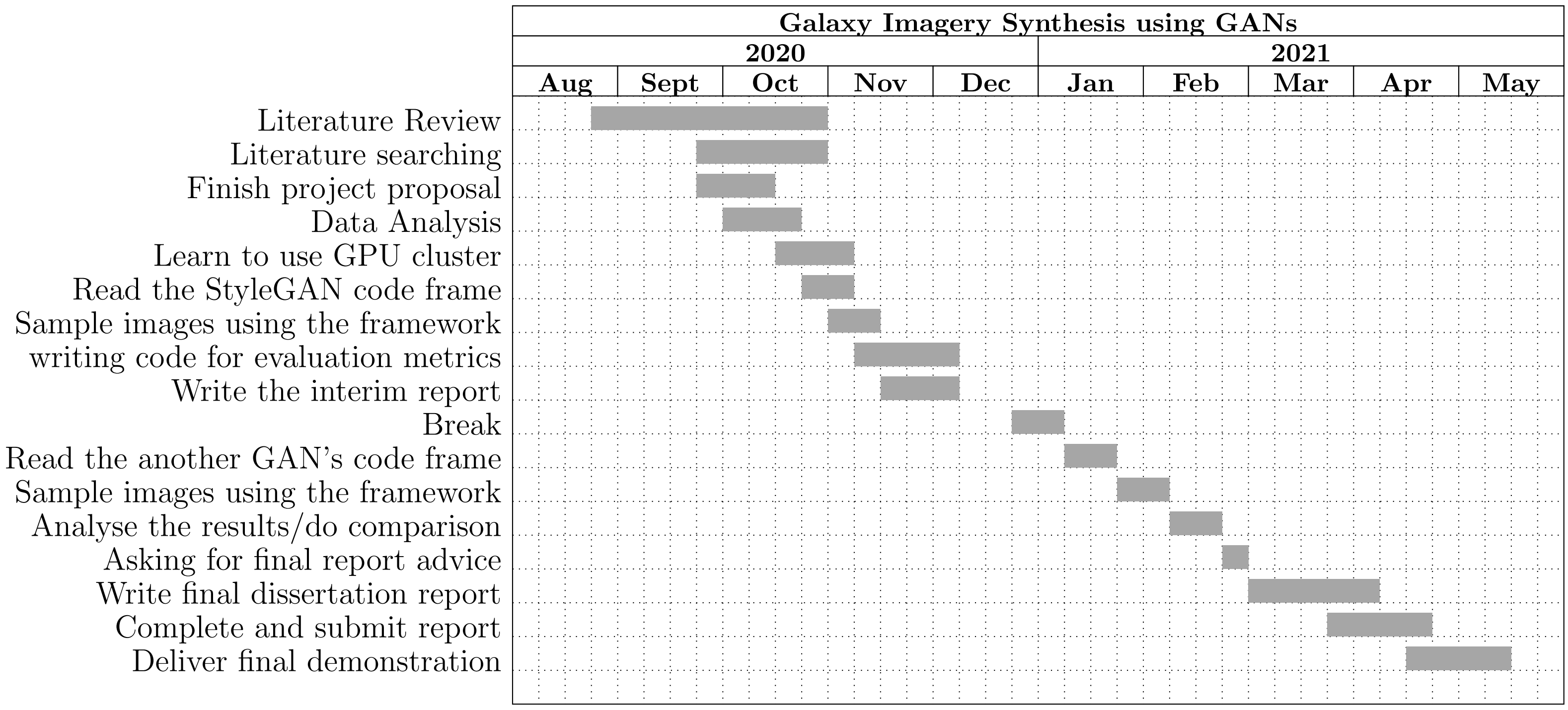


Figure 6.2: Revised Time Plan

I strictly fulfilled the requirements in accordance with the newly formulated plan, but with the following changes. First, I did not complete a new GAN research in the second semester. I cannot let COCO-GAN complete the training of custom pictures. BigGAN needs a large number of training sets, I knew in advance that the results must be abysmal. Second, since I chose the official framework for model training in the second semester, I cannot compare the results with the results in the first semester, so I made the first semester's progress again and then analyzed and compared the data. Third, I did not expect that a model's training would take two weeks in the second semester, which resulted in me not having time to debug with the interpolation and vector arithmetic code. Fortunately, the Easter holiday gave me plenty of time to complete the project smoothly.

6.2 Future Work

I do not think it is a good way to start with an image with a resolution of 64 and use the nearest neighbour to upsample for comparison experiments. Using upsampling will make the picture blurry, even if the resolution is high. Thus, it will not show the advantages of StyleGAN. In this way, the image generated by the model will also be blurred. On the contrary, using higher-resolution images at the beginning and comparing the downsampled images will make the experimental data more convincing. Therefore, if high-resolution galaxy images are available, using downsampling to do comparative experiments is a more reasonable way.

Furthermore, the comparison is under a configuration of reproducing results for FFHQ and LSUN Cat at 256x256. If more advanced equipment is available, I can use a configuration of reproducing results for MetFaces at 1024x1024. With 2, 4, or even 8 GPUs, training time can be hugely decreased. The comparison set probability p at 0.7 by default. However, p can be changed by command line to present another comparison.

6.3 Contributions and reflections

Although I have encountered many unexpected technical difficulties in the process, such as accidentally deleting the school Linux system's personal account and having difficulty writing batch scripts for Slurm, the project has progressed smoothly. One regret is that I did not complete the exploration of other GANs but only explored a small part of StyleGAN. This also made me realize that I lack knowledge and need to make more efforts in computer vision. Secondly, I regret the lack of 128$\times$ images as the result of the training set, because I did not expect that the training of a model would last for two weeks. If this is the case, I should have reduced the number of training steps for each model, such as from 15,000 kimg to 13,000 kimg. Nevertheless, before I set the number of steps, I did not know when the FID and KID scores will converge, so it is excusable to think about it now. But in short, I have completed the exploration of the StyleGAN model, including the structure and methodology of the model. I also explored the unique features of StyleGAN2 (interpolation and vector arithmetic). In addition, I also compared the results of using the ADA model and not using it with a limited training set.

I am delighted with the image processing knowledge gained in the project because I take image processing as a career plan. However, I am not satisfied with the project's progress because I did not expect so many technical problems to occur. I initially thought that running these codes would be like C language. This experience made me feel that I lack fundamental knowledge in this field, even after taking a machine learning course last semester. This project made me deeply feel that the development of computer vision is not that easy, and I will continue to explore this field at the graduate level.

Bibliography

[1] K. Kincade and L. B. N. Laboratory, "CosmoGAN: Training a neural network to study dark matter." [Online]. Available: https://phys.org/news/2019-05-cosmogan-neural-network-dark.html

[2] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," *arXiv:1809.11096 [cs, stat]*, Feb. 2019, arXiv: 1809.11096. [Online]. Available: http://arxiv.org/abs/1809.11096

[3] C. H. Lin, C.-C. Chang, Y.-S. Chen, D.-C. Juan, W. Wei, and H.-T. Chen, "COCO-GAN: Generation by Parts via Conditional Coordinating," *arXiv:1904.00284 [cs, stat]*, Jan. 2020, arXiv: 1904.00284. [Online]. Available: http://arxiv.org/abs/1904.00284

[4] "This Person Does Not Exist." [Online]. Available: https://thispersondoesnotexist.com/

[5] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," *arXiv:1606.03498 [cs]*, Jun. 2016, arXiv: 1606.03498. [Online]. Available: http://arxiv.org/abs/1606.03498

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, Jul. 2017, pp. 5967–5976. [Online]. Available: http://ieeexplore.ieee.org/document/8100115/

[7] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," *arXiv:1606.03476 [cs]*, Jun. 2016, arXiv: 1606.03476. [Online]. Available: http://arxiv.org/abs/1606.03476

[8] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul. 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[9] C. Veni, "On the Classification of Imbalanced Data Sets," Tech. Rep., Jul. 2018.

[10] F. H. K. d. S. Tanaka and C. Aranha, "Data Augmentation Using GANs," *arXiv:1904.09135 [cs, stat]*, Apr. 2019, arXiv: 1904.09135. [Online]. Available: http://arxiv.org/abs/1904.09135

[11] Z. Hussain, F. Gimenez, D. Yi, and D. Rubin, "Differential Data Augmentation Techniques for Medical Imaging Classification Tasks," *AMIA Annual Symposium Proceedings*, vol. 2017, pp. 979–984, Apr. 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5977656/

[12] W. J. Pearson, L. Wang, J. W. Trayford, C. E. Petrillo, and F. F. S. v. d. Tak, "Identifying galaxy mergers in observations and simulations with deep learning," *Astronomy & Astrophysics*, vol. 626, p. A49, Jun. 2019, publisher: EDP Sciences. [Online]. Available: https://www.aanda.org/articles/aa/abs/2019/06/aa35355-19/aa35355-19.html

[13] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255, iSSN: 1063-6919.

[14] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep Learning Face Attributes in the Wild," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 3730–3738, iSSN: 2380-7504.

[15] L. Fussell and B. Moews, "Forging new worlds: high-resolution synthetic galaxies with chained generative adversarial networks," *Monthly Notices of the Royal*

Astronomical Society, vol. 485, no. 3, pp. 3203–3214, May 2019, arXiv: 1811.03081. [Online]. Available: http://arxiv.org/abs/1811.03081

[16] M. Dia, E. Savary, M. Melchior, and F. Courbin, “Galaxy Image Simulation Using Progressive GANs,” *arXiv:1909.12160 [astro-ph, stat]*, Sep. 2019, arXiv: 1909.12160. [Online]. Available: http://arxiv.org/abs/1909.12160

[17] M. Jiménez, M. T. Torres, R. John, and I. Triguero, “Galaxy Image Classification Based on Citizen Science Data: A Comparative Study,” *IEEE Access*, vol. 8, pp. 47 232–47 246, 2020, conference Name: IEEE Access.

[18] M. Esmaeilpour, R. A. Sallo, O. St-Georges, P. Cardinal, and A. L. Koerich, “Conditioning Trick for Training Stable GANs,” *arXiv:2010.05844 [cs, eess]*, Oct. 2020, arXiv: 2010.05844. [Online]. Available: http://arxiv.org/abs/2010.05844

[19] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han, “Differentiable Augmentation for Data-Efficient GAN Training,” *arXiv:2006.10738 [cs]*, Dec. 2020, arXiv: 2006.10738. [Online]. Available: http://arxiv.org/abs/2006.10738

[20] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, “Training Generative Adversarial Networks with Limited Data,” *arXiv:2006.06676 [cs, stat]*, Oct. 2020, arXiv: 2006.06676. [Online]. Available: http://arxiv.org/abs/2006.06676

[21] R. Bonney, J. L. Shirk, T. B. Phillips, A. Wiggins, H. L. Ballard, A. J. Miller-Rushing, and J. K. Parrish, “Next Steps for Citizen Science,” *Science*, vol. 343, no. 6178, pp. 1436–1437, Mar. 2014, publisher: American Association for the Advancement of Science Section: Policy Forum. [Online]. Available: https://science.sciencemag.org/content/343/6178/1436

[22] C. Lintott, K. Schawinski, S. Bamford, A. Slosar, K. Land, D. Thomas, E. Edmondson, K. Masters, R. C. Nichol, M. J. Raddick, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg, “Galaxy Zoo 1: data release of morphological classifications for nearly 900 000 galaxies*,” *Monthly Notices of the Royal*

Astronomical Society, vol. 410, no. 1, pp. 166–178, Jan. 2011. [Online]. Available: https://doi.org/10.1111/j.1365-2966.2010.17432.x

[23] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg, "Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey*," *Monthly Notices of the Royal Astronomical Society*, vol. 389, no. 3, pp. 1179–1189, Sep. 2008. [Online]. Available: https://doi.org/10.1111/j.1365-2966.2008.13689.x

[24] R. Simpson, K. R. Page, and D. De Roure, "Zooniverse: observing the world's largest citizen science platform," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14 Companion. New York, NY, USA: Association for Computing Machinery, Apr. 2014, pp. 1049–1054. [Online]. Available: https://doi.org/10.1145/2567948.2579215

[25] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *arXiv:1406.2661 [cs, stat]*, Jun. 2014, arXiv: 1406.2661. [Online]. Available: http://arxiv.org/abs/1406.2661

[26] I. J. Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," *ArXiv*, 2017.

[27] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114 [cs, stat]*, May 2014, arXiv: 1312.6114. [Online]. Available: http://arxiv.org/abs/1312.6114

[28] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv:1511.06434 [cs]*, Jan. 2016, arXiv: 1511.06434. [Online]. Available: http://arxiv.org/abs/1511.06434

[29] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," *arXiv:1805.08318 [cs, stat]*, Jun. 2019, arXiv: 1805.08318. [Online]. Available: http://arxiv.org/abs/1805.08318

[30] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An Overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, Jan. 2018, conference Name: IEEE Signal Processing Magazine.

[31] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution," 2017, pp. 624–632. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Lai_Deep_Laplacian_Pyramid_CVPR_2017_paper.html

[32] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 700–708.

[33] S. K. Gorti and J. Ma, "Text-to-Image-to-Text Translation using Cycle Consistent Adversarial Networks," *arXiv:1808.04538 [cs, stat]*, Aug. 2018, arXiv: 1808.04538. [Online]. Available: http://arxiv.org/abs/1808.04538

[34] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *arXiv:1812.04948 [cs, stat]*, Mar. 2019, arXiv: 1812.04948. [Online]. Available: http://arxiv.org/abs/1812.04948

[35] Y. Bengio and Y. Lecun, "Convolutional Networks for Images, Speech, and Time-Series," Nov. 1997.

[36] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," *arXiv:1912.04958 [cs, eess, stat]*, Mar. 2020, arXiv: 1912.04958. [Online]. Available: http://arxiv.org/abs/1912.04958

[37] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep, Big, Simple Neural Nets for Handwritten Digit Recognition,” *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, Dec. 2010. [Online]. Available: https://doi.org/10.1162/NECO_a_00052

[38] T. DeVries and G. W. Taylor, “Dataset Augmentation in Feature Space,” Feb. 2017. [Online]. Available: https://openreview.net/forum?id=HyaF53XYx

[39] Z. Wan, Y. Zhang, and H. He, “Variational autoencoder based synthetic data generation for imbalanced learning,” in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Nov. 2017, pp. 1–7.

[40] A. Antoniou, A. Storkey, and H. Edwards, “Data Augmentation Generative Adversarial Networks,” *arXiv:1711.04340 [cs, stat]*, Mar. 2018, arXiv: 1711.04340. [Online]. Available: http://arxiv.org/abs/1711.04340

[41] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, conference Name: Proceedings of the IEEE.

[42] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, “Human-level concept learning through probabilistic program induction,” *Science*, vol. 350, no. 6266, pp. 1332–1338, Dec. 2015, publisher: American Association for the Advancement of Science Section: Research Article. [Online]. Available: https://science.sciencemag.org/content/350/6266/1332

[43] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” 2015, publisher: British Machine Vision Association. [Online]. Available: https://ora.ox.ac.uk/objects/uuid:a5f2e93f-2768-45bb-8508-74747f85cad1

[44] J. P. Cohen, M. Luck, and S. Honari, “Distribution Matching Losses Can Hallucinate Features in Medical Image Translation,” *arXiv:1805.08841 [cs]*, Oct. 2018, arXiv: 1805.08841. [Online]. Available: http://arxiv.org/abs/1805.08841

[45] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.

[46] M. Pesteie, P. Abolmaesumi, and R. N. Rohling, “Adaptive Augmentation of Medical Data Using Independently Conditional Variational Auto-Encoders,” *IEEE Transactions on Medical Imaging*, vol. 38, no. 12, pp. 2807–2820, Dec. 2019, conference Name: IEEE Transactions on Medical Imaging.

[47] K. Su, E. Zhou, X. Sun, C. Wang, D. Yu, and X. Luo, “Pre-trained StyleGAN Based Data Augmentation for Small Sample Brain CT Motion Artifacts Detection,” in *Advanced Data Mining and Applications*, ser. Lecture Notes in Computer Science, X. Yang, C.-D. Wang, M. S. Islam, and Z. Zhang, Eds. Cham: Springer International Publishing, 2020, pp. 339–346.

[48] P. Toan, L. Doan, N. Tran, N. Tam, N. Hoai, D. Huy, and T. Thanh, “Interpreting the Latent Space of Generative Adversarial Networks using Supervised Learning,” Nov. 2020.

[49] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, “Optimizing the Latent Space of Generative Networks,” *arXiv:1707.05776 [cs, stat]*, May 2019, arXiv: 1707.05776. [Online]. Available: http://arxiv.org/abs/1707.05776

[50] R. Abdal, Y. Qin, and P. Wonka, “Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?” 2019, pp. 4432–4441. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/html/Abdal_Image2StyleGAN_How_to_Embed_Images_Into_the_StyleGAN_Latent_Space_ICCV_2019_paper.html

[51] L. Fetty, M. Bylund, P. Kuess, G. Heilemann, T. Nyholm, D. Georg, and T. Löfstedt, “Latent space manipulation for high-resolution medical image synthesis via the StyleGAN,” *Zeitschrift für Medizinische Physik*, vol. 30, no. 4, pp. 305–314, Nov. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0939388920300544

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385

[53] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, Jun. 2015, pp. 1–9. [Online]. Available: http://ieeexplore.ieee.org/document/7298594/

[54] N. Jiang and L. Wang, "Quantum image scaling using nearest neighbor interpolation," *Quantum Information Processing*, vol. 14, no. 5, pp. 1559–1571, May 2015. [Online]. Available: https://doi.org/10.1007/s11128-014-0841-8

[55] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux Utility for Resource Management," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer, 2003, pp. 44–60.

[56] M. Jafari and H. Molaei, "Spherical Linear Interpolation and Bézier Curves," *General Scientific Researches*, vol. 2, Jan. 2014.

[57] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *arXiv:1512.00567 [cs]*, Dec. 2015, arXiv: 1512.00567 version: 3. [Online]. Available: http://arxiv.org/abs/1512.00567

[58] S. Barratt and R. Sharma, "A Note on the Inception Score," *arXiv:1801.01973 [cs, stat]*, Jun. 2018, arXiv: 1801.01973. [Online]. Available: http://arxiv.org/abs/1801.01973

[59] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," *arXiv:1706.08500 [cs, stat]*, Jan. 2018, arXiv: 1706.08500. [Online]. Available: http://arxiv.org/abs/1706.08500

[60] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," *arXiv:1801.01401 [cs, stat]*, Jan. 2021, arXiv: 1801.01401. [Online]. Available: http://arxiv.org/abs/1801.01401

Appendix A

User Manuals

A.1 stylegan2-pytorch

You will need a machine with a GPU and CUDA installed. Then pip install the package like this:

```
$ pip install stylegan2_pytorch
```

To allow FID score calculation, your will additionally pip install pytorch_fid package.

```
$ pip install pytorch-fid
```

To train a model from a given image file path, and saving the result to a user-defined folder, you can use:

```
$ stylegan2_pytorch --data /path/to/images --name my-project-name
--results_dir /path/to/results/dir
```

To allow calculating FID, before the training, you can add following command in the end and define how many steps FID is calculated

```
$ --calculate-fid-every steps
```

To custom define training steps, you can add

```
$ --num-train-steps final_steps
```

A.2 stylegan2-ada

First thing you need to do is to change the format of the training set to TFRecords.

```
$ python dataset_tool.py create_from_images destination image_path
```

To train the model, you can use

```
$ python train.py --outdir=result_folder --gpu=1 --kimg 15000
--aug=noaug --mirror=1 --cfg=paper256
--data=/cs/home/scyhx2/datasets/custom --metrics=kid50k,fid50k
```

where you can define your result path, GPU usage, final training steps, training with or without ADA, mirror, training configuration, source image data, metrics respectively.

Training is often interrupted by timer, to resume training, you can use

```
$ python train.py --outdir=result_folder --gpus=1 --kimg resume_steps
--resume=model_path
```

Interpolation can be achieved by providing out directory and network.

```
$ python interpolate.py --outdir=out_directory --network=network
```

Vector arithmetic can be accomplished in a similar way.

```
$ python vectorArithmetic.py --outdir=out_directory --network=network
```

To generate fake images, you can refer to **interpolate.py** or **vectorArithmetic.py**. In both scripts, **generate_image_random()** will return a fake image and its corresponding vector given a random seed, you can write a for loop according to this to create as many fake images as you want. Then do vector operations using the vectors that corresponds to these images.