Furry Image Generation with Generative Adversarial Networks

Zehua Guo, B.Sc

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science

(Augmented and Virtual Reality)

Supervisor: Michael Manzke

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Zehua Guo

August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Zehua Guo

August 19, 2022

Furry Image Generation with Generative Adversarial Networks

Zehua Guo, Master of Science in Computer Science
University of Dublin, Trinity College, 2022

Supervisor: Michael Manzke

In this dissertation, a generative adversarial network is trained for furry image generation through transfer learning. The base model is StyleGAN2 pre-trained on the Flickr-Face-HQ dataset at the resolution of 256×256. Structure loss and similarity loss are employed to help training the new model based on pre-trained weights. Several configurations are tested for training, from which the best one is selected and used for layer swapping to capture more human facial features. Results show that furry features from the training data are successfully transferred to the generated images.

Acknowledgments

The sincerest thanks to my parents, who have been supporting me even when in the darkest times, and to the Final Fantasy series, which continue to bring me surprise and a world full of magic and imagination.

Zehua Guo

University of Dublin, Trinity College

August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Furry fandom refers to a group of people who like anthropomorphic animal characters with furry features. They are active in animation, comics and games (ACG) areas as a group with distinct characteristics. Typical furry figures include Judy Hopps and Nick Wilde, the main characters in the well-known Disney movie Zootopia, whose prototypes are a rabbit and a red fox, respectively. Furry fans tend to discuss on their hobby through painting, games and cosplays, many of whom have little skills in painting and must pay commercial artists to create their own furry figures. The fact suggests the potential need for developing a tool to help such people design their unique figures in an easier way. Since generative methods have been popular in recent years, generative adversarial networks (GANs), as a representative of such methods, would be suitable for this goal. The aim of this project is to train a GAN model to generate stylised furry-like portrait images from real human portraits. For any given input image, the output image should preserve the overall structure of the input image with added furry features.

1.2 Dissertation Structure

The dissertation consists of five chapters apart from the introduction: Chapter 2 reviews some of the literature related to the method and techniques used in this paper. Chapter 3 describes the design of the entire model, followed by Chapter 4 which discusses on the implementation details. Chapter 5 discusses the experiments and some results. Finally, Chapter 6 concludes the whole dissertation and identifies shortcomings and possible future works.

Chapter 2

Literature Review

2.1 Generative Adversarial Networks (GANs)

GAN was first introduced by (Goodfellow et al. (2014)). The framework involves two independent networks trained simultaneously but against each other, namely the generator and the discriminator. The goal for generators is to produce fake images that look like real world images, whereas the discriminators try to distinguish fake images apart from real images. The process of training can be expressed as

$$\max_{\mathcal{D}} \min_{\mathcal{G}} V(\mathcal{G}, \mathcal{D})$$

where

$$V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{p_{data}(\mathbf{x})} \log \mathcal{D}(\mathbf{x}) + \mathbb{E}_{p_g(\mathbf{x})} \log \left(1 - \mathcal{D}(\mathbf{x})\right) \tag{2.1}$$

Under the ideal case, the discriminator should use a probability of 0.5 to mark an input as either fake or real images, and the generator should be able to learn a distribution $p_g(\mathbf{x})$ that equals to the real distribution $p_{data}(\mathbf{x})$.

Because of the amazing generative capability, GANs have been used widely in many applications and have many variants. (Karras et al. (2017)) uses layers with multiple resolutions in both the generator and the discriminator to progressively generate high-

resolution images in an unsupervised setting. Training stability also benefits from this new architecture. Further, inspired by style transfer (Huang and Belongie (2017)), (Karras et al. (2019)) proposed a new generator architecture that improves image quality. The authors also introduce an intermediate, disentangled latent space $\mathcal{W}$ in additional to the original latent space $\mathcal{Z}$ for better style controlling, along with a mapping network converting latent codes in $\mathcal{Z}$ space to $\mathcal{W}$ space. StyleGAN2, as the successor of StyleGAN, takes one more step by using perceptual path length regularizer. To alleviate the problem of discriminator overfitting due to shortage of training data, adaptive discriminator augmentation (ADA) mechanism is introduced into StyleGAN2, making it more stable with limited data.

Transfer learning has been demonstrated to work well with StyleGAN. Instead of starting from scratch, transfer learning uses tricks to leverage existing models to significantly reduce training time and improve results. Pinkney and Adler (2020) applied transfer learning based on StyleGAN and performed interpolation between different generative models. Song et al. (2021) trained their model initialised with the weights of a pre-trained StyleGAN2 and achieved fast training speed as well as satisfactory results. **?** used a pre-trained StyleGAN2 based on which the parameters were fine-tuned for cartoon face generation.

2.2 Stylised Image Generation

Stylised image generation is a popular area in non-photorealistic rendering. Early works involve low-level histogram matching to generate textures (Heeger and Bergen (1995)). Gatys et al. (2016) explored neural style transfer based on convolutional neural networks (CNNs) to synthesize images and achieved satisfactory consequences. However, these methods fail to perform well on stylised portrait.

Isola et al. (2017) proposed a general purpose image generator *pix2pix*, leveraging conditional adversarial networks to realise image-to-image translation. Pix2pix not only

learns a mapping from the input image to the output image, but also learns a loss function for training the mapping, which has achieved great succuss. Similarly, Sangkloy et al. (2017) also used conditional adversarial networks to implement a feed-forward network that can generate photorealistic images from sketches. Despite the effect, training such networks require paired data which can be hard to acquire. Zhu et al. (2017) introduces a cycle consistency loss that helps preserving the original image among translations, enabling training in an unsupervised manner.

A number of studies have demonstrated the potential of StyleGAN in generating stylised images. (Pinkney and Adler (2020)) presented a new method that can interpolate between models based on the StyleGAN architecture to enable cross-domain control of images. (Song et al. (2021)) introduced a multi-path structure embedded in the low levels of a standard StyleGAN2 architecture for stylised portrait generation. The training speed is fast with no compromise on image quality. Richardson et al. (2021) employed a fixed pre-trained StyleGAN2 generator for training an encoder that can extract features at different level of details for numerous tasks, including GAN inversion, sketch-to-image, etc.

Chapter 3

Design

3.1 Overview

For transfer learning, the source generator $\mathcal{G}_s$ is initialised from a pre-trained StyleGAN2 generator on the Flickr-Faces-HQ dataset (Karras et al. (2019)), which contains approximately 70,000 high quality human face images. The target generator $\mathcal{G}_t$ is also initialised with the same weights as $\mathcal{G}_s$, but is trained on a new furry portrait image dataset.

A standard StyleGAN2 structure is composed of three networks: the mapping network, the generator network and the discriminator network. The mapping network is responsible for converting samples in $\mathcal{Z}$ space into a less entangled intermediate space $\mathcal{W}$. Latent codes in $\mathcal{W}$ space are subsequently fed into the generator to produce the fake or generated images. Lastly, the fake images, along with real training images, are fed into the discriminator whose responsibility is to distinguish real images from fake ones.

StyleGAN2 is targeted at resolutions up to 1024×1024. However, to reduce computational resource use and training time, target resolution is set to 256×256 in this project, which means a reduced number of style blocks from 9 to 7.

Figure (3.1) illustrates the entire structure of the model.

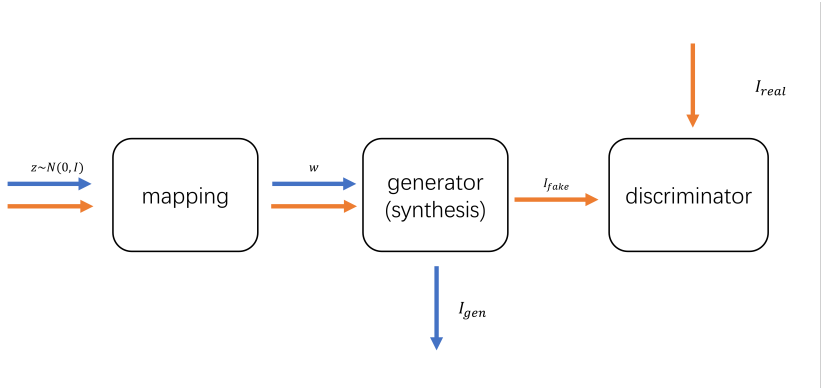


Figure 3.1: An overview of the model. Inputs from $\mathcal{Z}$ space are first mapped to an intermediate space $\mathcal{W}$ and then fed into the generator or the synthesis network. In this project, the term "generator" usually refers to the mapping network and the synthesis network as a whole, unless explicitly stated. The generator subsequently produces fake images as outputs or fed in to the discriminator for training. I_{fake} and I_{gen} are actually the same thing but under different contexts. The orange arrows above indicate data flow in the training process while the blue arrows indicate data flow in the inference process.

3.2 Mapping Network

The mapping network is composed of an 8-layer multi-layer perceptron (MLP). All layers in an MLP are fully-connected layers. Karras et al. (2019) discovered that the original $\mathcal{Z}$ space from which inputs to generators are typically sampled can be entangled in terms of linearity, thus preventing reasonable results from interpolation in $\mathcal{Z}$ space. Besides, $\mathcal{Z}$ space is supposed to obey a fixed distribution, usually a normal distribution, which may not best fit the actual distribution of the inputs. Based on these facts, an intermediate latent space $\mathcal{W}$ is proposed that complies with a learned distribution through training. $\mathcal{W}$ space is more linearly separable, meaning a reasonably interpolated image is expected with corresponding interpolation of latent codes in this space. For example, an interpolation between the latent codes standing for round and rectangle is supposed to generate a shape like a rounded rectangle instead of a triangle in the generated image.

The weights from a pre-trained StyleGAN2 mapping network on the FFHQ (Karras et al. (2019)) at resolution 256×256 are loaded as a starting point. The model is

subsequently trained on the new dataset.

Figure (3.2) shows the structure of the mapping network.

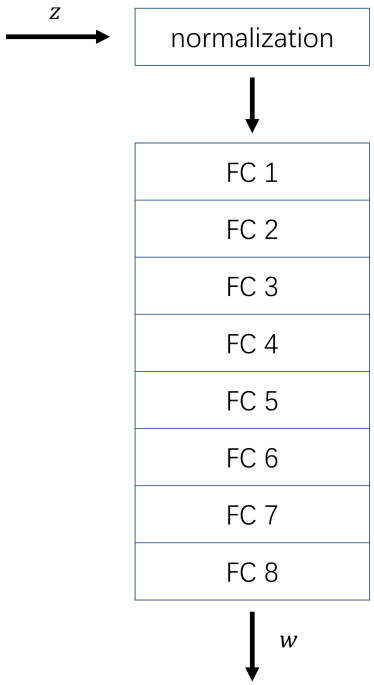


Figure 3.2: The structure of the mapping network. FC denotes for fully-connected layer. Normalised latent codes in $\mathcal{Z}$ space are forwarded through 8 fully-connected layers, mapped to the less entangled $\mathcal{W}$ space. Both $\mathcal{Z}$ space and $\mathcal{W}$ space are typically 512-dimensional.

3.3 Generator

3.3.1 Structure

The generator mainly consists of several style blocks. Depending on the target resolution, there can be 9 style blocks at most, from 4×4 to 1024×1024. Each of the block layer contains 2 3×3 convolutional layers, except for the first style block at 4×4 resolution, where a constant layer is used instead. This is because it has almost the same effect as using a traditional input layer but is simpler. (Karras et al. (2019))

Latent codes in $\mathcal{W}$ are not directly fed into the generator as the inputs. Instead, they are involved in image generation through a mechanism called adaptive instance

normalization (AdaIN) (Huang and Belongie (2017), Karras et al. (2019))

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

where i refers to the i-th channel of the input $\mathbf{x}$, $\mathbf{y}_s$ and $\mathbf{y}_b$ refer to the factors controlling scale and bias, respectively. $\mathbf{y}$ is also referred to as *styles* (Karras et al. (2019)). Each channel, or feature map in $\mathbf{x}$ is adjusted independently on a per channel, per instance basis, which is conducted by a learnable affine module consisting of fully-connected layers. By this adjustment, the styles are "transferred" to the feature maps (Huang and Belongie (2017)) to achieve control over the generated images.

Gaussian noises are added before fed into the AdaIN layer to increase random features to the generated images, making them appear more realistic.

To accelerate training and make it more affordable, the target resolution is set to 256×256, meaning that there are 7 style blocks in the generator. (Karras et al. (2020b)) introduced 3 different architectures of StyleGAN2. In this project, the default architecture "skip" is used as in the original paper. This architecture employs an additional toRGB layer to the end of each style block, taking the RGB outputs into consideration to solve the problem of shift invariance occuring with the previous architecture (Karras et al. (2017)).

The complete structure of the generator is shown in Figure 3.3

3.3.2 Loss function

This project uses the non-saturating (Goodfellow et al. (2014)) loss function $\mathcal{L}_G$, which is defined as

$$\mathcal{L}_G = \log \mathcal{D}(\mathcal{G}_t(\mathbf{z})) \tag{3.2}$$

where $\mathbf{z}$ is the encoded latent vectors in ‡ space and $\mathcal{G}_t$ stands for the current generator being trained, as opposed to the loaded pre-trained generator $\mathcal{G}_s$.

Back (2021) proposed a simple structure loss $\mathcal{L}_{struct}$ based on mean squared error

(MSE) to help transfer learning from existing weights. The idea is to sum up the MSE values between the outputs of tRGB layers in both the source generator and the target generator.

$$\mathcal{L}_{struct} = \sum_{k=1}^{n} \text{MSE}\left[\mathcal{G}_s^k(\mathbf{z}) - \mathcal{G}_t^k(\mathbf{z})\right] \tag{3.3}$$

where n is the maximal target style block, $\mathcal{G}_s^k(\mathbf{z})$ and $\mathcal{G}_t^k(\mathbf{z})$ are the RGB outputs of the k-th style block in the source generator and the target generator, respectively.

As furry figures tend to have different facial structures from human-beings, the similarity loss $\mathcal{L}_{sim}$ introduced by Song et al. (2021) is employed in this project. $\mathcal{L}_{sim}$ is essentially a VGG16-based Learned Perceptual Image Patch Similarity (LPIPS) (Simonyan and Zisserman (2014), Zhang et al. (2018)) with the first 9 layers being discarded to encourage the generator to capture more facial structural information while ignoring other details (Song et al. (2021)).

$$\mathcal{L}_{sim} = \sum_{i=9}^{30} (\mathcal{L}_{lpips}^i(\mathcal{G}_t(\mathbf{z}), \mathcal{G}_s(\mathbf{z}))) \tag{3.4}$$

where i refers to the i-th layer in LPIPS calculation.

In addition, this project employs the path length regularization (Karras et al. (2020b)) $\mathcal{L}_{path}$ to encourage reliable generative models, which is a default configuration of the StyleGAN2 training framework.

The overall training objective for the generator can be expressed as

$$\max_{\mathcal{G}_t} (\mathcal{L}_G - w_{path}\mathcal{L}_{path} - w_{struct}\mathcal{L}_{struct} - w_{sim}\mathcal{L}_{sim}) \tag{3.5}$$

where non-negative w_{path}, $w_{struct}and$ w_{sim} stand for the relative weights for path length regularization, structure loss and similarity loss, respectively.

3.4 Discriminator

3.4.1 Structure

The discriminator contains the same number of blocks as the generator, but in a reversed order i.e. the spatial resolutions gradually shrink from 1024×1024 at most to 4×4. Each block, except for the last block at spatial resolution 4×4, contains two 3×3 convolutional layers. The last block consists of one 3×3 convolutional layer, a minibatch stddev layer to encourage the model to capture more variation from the training data (Karras et al. (2017), Salimans and Kingma (2016)) and an output layer.

Freezing the high-resolution layer of the discriminator, also known as Freeze-D, has been demonstrated to perform well in transfer learning (Mo et al. (2020)). The parameters in the affected layers are not updated as the training process proceeds, hence are "frozen". According to the tests from (Karras et al. (2020a)), freezing layers with the 3 or 4 highest resolutions gives the best results when the target resolution is at 1024×1024. As 256×256 is the target resolution in this project, the number of blocks to apply Freeze-D is set to 2. These parameters remain unchanged throughout the whole training process once initialised.

Karras et al. (2020b) also introduced the new architecture in discriminator. In this project, the default configuration "resnet" (He et al. (2016)) is used as in the original paper. The weights of the discriminator are loaded from the same savepoint file as the generator.

The complete structure of the discriminator is shown in Figure 3.4.

3.4.2 Loss function

The adversarial loss $\mathcal{L}_D$ for the discriminator is defined as

$$\mathcal{L}_D = \log \mathcal{D}(\mathbf{y}) + \log 1 - \mathcal{D}(\mathcal{G}_t(\mathbf{z})) \tag{3.6}$$

where $\mathbf{y}$ refers to real images (Goodfellow et al. (2014)). As discriminator training is prone to instability (Mescheder et al. (2018)), this project uses R1 regularization (Mescheder et al. (2018), Ross and Doshi-Velez (2018)) to stabilise the training

$$\mathcal{L}_{R1} = \frac{\gamma}{2}\mathbb{E}[\|\nabla\mathcal{D}(\mathbf{y})\|^2] \tag{3.7}$$

where γ is a constant controlling the strength of the regularization.

The overall training objective for the discriminator can be written as

$$\max_{\mathcal{D}} (\mathcal{L}_D - \mathcal{L}_{R1}) \tag{3.8}$$

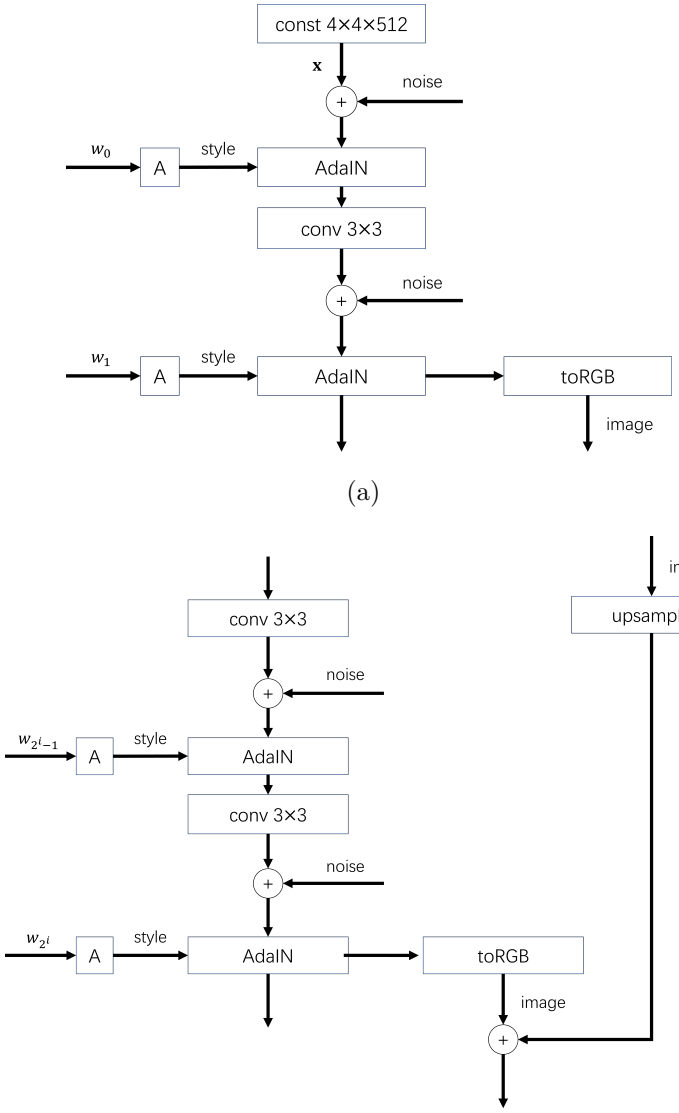


Figure 3.3: (a) The structure of the first style block, whose spatial resolution is 4×4. The constant layer serves as the input layer. Latent codes in $\mathcal{W}$ space are translated into styles through an affine transformation module "A". These styles are then involved in AdaIN to scale and bias each channel of the input $\mathbf{x}$, trying to "transfer" styles from w to $\mathbf{x}$. Noises are introduced to add more random details in the generated images. Before entering the next style block, inputs are converted to RGB images to form the so-called "skip" architecture proposed by Karras et al. (2020b). (b) The structure of the other style blocks. There are totally 7 style blocks in this project as the target resolution is 256×256. Compared to the first style block, the only difference is that two 3×3 convolutional layers are included.

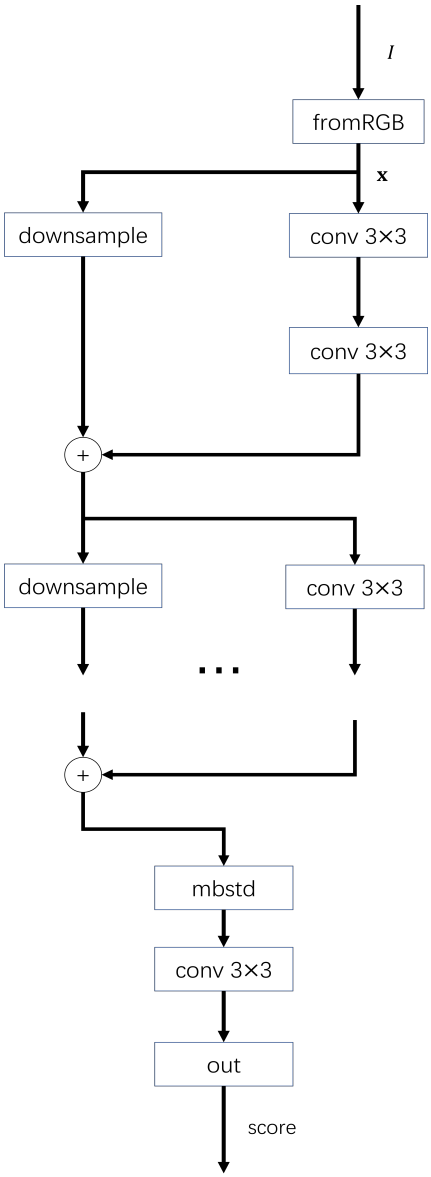


Figure 3.4: The structure of the discriminator. The number of blocks is the same as the generator. Each block contains two 3×3 convolutional layers, except for the last block. ”mbstd” stands for minibatch standard deviation layer, which helps to capture more variation from the training data Salimans and Kingma (2016). The output of each block is accumulated through the left part in the above figure, which is similar to a residual network structure He et al. (2016).

Chapter 4

Implementation

4.1 Data Preparation

The training data were obtained from the E621 Faces Dataset (arfafax (2020)), which contains 186k cropped furry face images collected from https://e621.net/. However, downloading all these images can take up to 106 gigabytes of storage, which is unaffordable. Besides, there are some inappropriate contents which are recorded in the faces_q.csv and features_q.csv, where the letter q means questionable. The author suggests that use a confidence score of greater than 0.99 for training. Considering the above factors, 3,550 images were finally obtained using the script that comes with this repository and setting -min-score=50, –csv=faces_s.csv, –min-confidence=0.99, and –crop-size=256.

Horizontal flip is used as the only means of data augmentation in this project. The amount of data is doubled to 7100 after this operation.

Figure 4.1 shows some of the data.

4.2 Environment Setup

The whole training process took place on a rented GPU cloud server provided by Alibaba Cloud. The machine is equipped with an NVIDIA A10 24G GPU. After completing the

Figure 4.1: Samples of the training data.

basic configuration according to the instructions on the provider's website, Docker Engine was downloaded and installed. Docker provides a simple way to recreate a specific runtime, much like a lightweight virtual machine. In this project, the Dockerfile from (NVlabs (2021a)) was used as the base image, upon which a specific container was created containing all needed runtime dependencies. By using the docker container, the chances for dependency errors due to manual configuration can be significantly reduced. The training data were then uploaded to the server and mounted from the physical machine to the container as a volume, enabling access to the data on the physical machine from within the container and avoiding data loss due to container shutdown or restart.

4.3 Model Configurations

4.3.1 General Settings

All the code is implemented in PyTorch. The whole training framework uses the official StyleGAN2 implementation (NVlabs (2021b)), with two new loss functions $\mathcal{L}_{struct}$ and $\mathcal{L}_{sim}$ added.

The pre-trained weights are downloaded from (NVlabs (2021c)), which contains the weights for the generator, the discriminator and a special generator "G_ema". "G_ema" is a moving average of the generator weights over the training steps and is smoother than snapshots directly taken from the training process (Karras et al. (2020b)).

The Adam optimiser (Kingma and Ba (2014)) is used for both the generator and the discriminator in this project, with the learning rate being 0.002 and ϵ being 1×10^{-8}, respectively. β is set to [0, 0.99] as the base value. However, it is influenced by lazy regularization discussed in the following subsection. The batch size is set to 32 and the total number of training images is set to 300,000 (300 kimg for short) as a starting point whose value is adjusted according to the pre-experiment discussed in the next chapter.

To reduce memory use, 16-bit floating point are used for the blocks with the 4 highest spatial resolutions for both the generator and the discriminator. All other blocks still use 32-bit floating point. This is also the default settings of the source pre-trained StyleGAN2 model.

γ and w_{path} remain the same as in the source model settings, where $\gamma = 1$ and $w_{path} = 2$. The choise of w_{struct} and w_{sim} is discussed in the next chapter for convenience.

4.3.2 Lazy Regularization

Karras et al. (2020b) have found that the regularization terms i.e. the path length regularization for the generator and the R1 regularization for the discriminator can be computed at an interval without compromising their effects. The path length regular-

ization is performed every 4 iterations and the R1 regularization is performed every 16 iterations, which are consistent with the settings of the source models. To cancel the influence due to an additional iteration of regularization, the hyperparameters for Adam optimiser (Kingma and Ba (2014)) are slightly adjusted accordingly, with $\lambda' = c \cdot \lambda$, $\beta_1' = (\beta_1)^c$ and $\beta_2' = (\beta_2)^c$, where $c = k/(k+1)$, k is the total number of iterations.

4.3.3 Freeze-D

Freeze-D (Mo et al. (2020)) is conducted on the blocks with the 2 highest resolutions i.e. setting the "freeze_layers" argument to 7. The reason why the number is 7 is because the first block in the discriminator has 4 layers: "from_rgb", "conv0", "conv1", "skip" and the second block has 3 layers: "conv0", "conv1", "skip". Specifically, the parameters are frozen by using torch.nn.Module.register_buffer() to store these parameters in buffers instead of using torch.nn.Parameters. Parameters stored in buffers are not influenced by optimisers, therefore excluded from the training process.

Chapter 5

Experiments

5.1 Pre-experiments

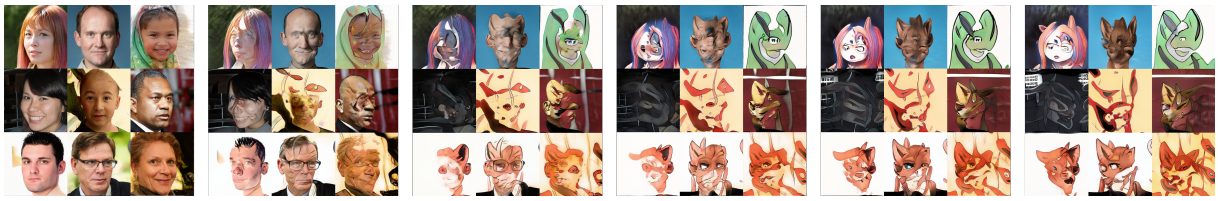
Pre-experiments was conducted to find the appropriate number of iterations as well as the possible combination of w_{struct} and w_{sim}. The configurations of these pre-experiments are detailed in Table 5.1.

	kimg	γ	w_{path}	w_{struct}	w_{sim}
config A	100	1	2	0	0
config B	100	1	2	1	0
config C	100	1	2	0	1

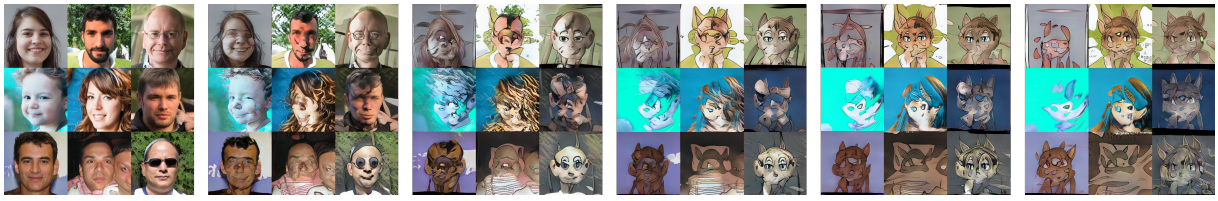
Table 5.1: Pre-experiment configurations

With a batch size of 32, each configuration took approximately 3 hours to train on an NVIDIA A10 24G GPU. Sample images are generated every 20 kimg. Some of the sample images are shown in Figure 5.1

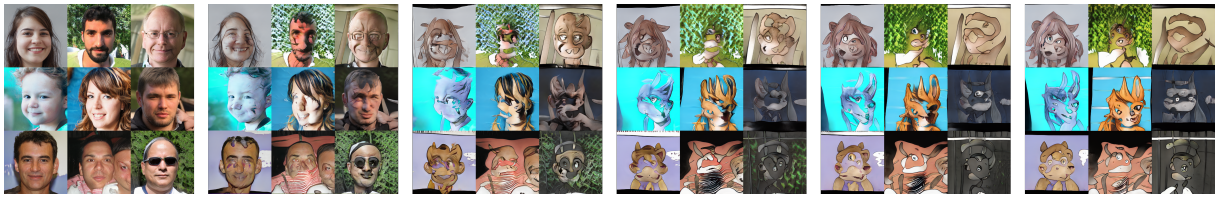
It can be seen from Figure 5.1 that config A has relatively better image quality compared with other two configurations. Config B with structure loss being applied can roughly maintain the facial structure, especially areas around the eyes. However, the image generated with config B still have fragmented parts compared with config A. Config C with similarity loss being applied achieved the poorest image quality among all three



(a)



(b)



(c)

Figure 5.1: (a)-(c) Results of pre-experiments with config A to config C. In each subfigure, from left to right, the number of images trained increases from 0 kimg to 100 kimg at an interval of 20 kimg.

configurations. Based on these results, config A and config B may be better for the task, which still needs more experiments to ensure. This also means that similarity loss may not help to improve image quality. Possible reasons are discussed in 5.3.

5.2 Formal Experiments

5.2.1 Model Training

The defects in pre-experiments may be because of short training time. Some new configurations are tested with a larger number of kimg as listed in Table 5.2

With the same batch size as in pre-experiments, each configuration took approximately 5 hours to train.

Figure 5.2 shows some sample images generated with the trained model. Despite some

	kimg	γ	w_{path}	w_{struct}
config D	300	1	2	0
config E	300	1	2	0.5
config F	300	1	2	1

Table 5.2: Formal experiment configurations

distortions, config E has achieved the best image quality. Thus, the model trained with config E is chosen as the transferred model to perform layer swapping on.

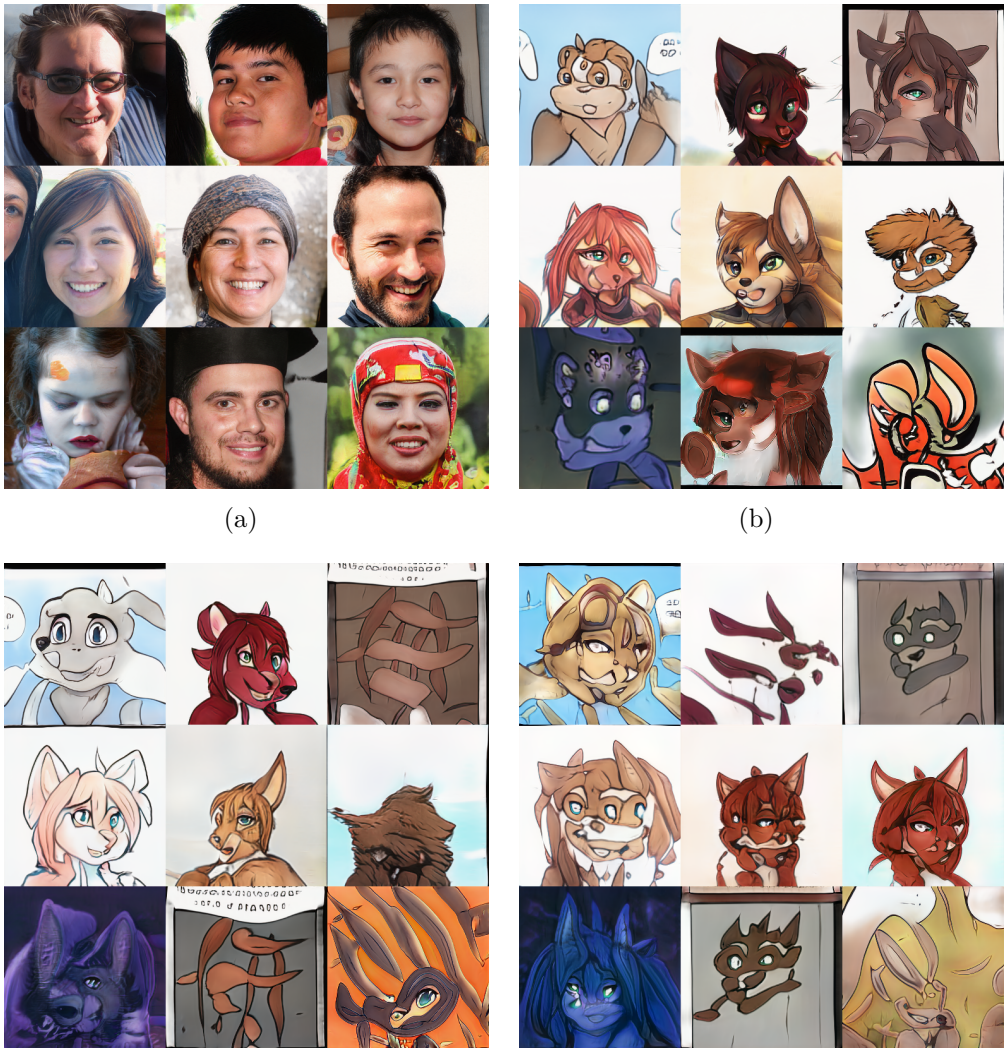
5.2.2 Layer Swapping

Pinkney and Adler (2020) introduced an interesting mechanism called layer swapping that can be used for transfer learned models. Given the source generator $\mathcal{G}_s$ and the target generator $\mathcal{G}_t$, the parameters of style blocks can be interpolated to generate images with more structural features of the source domain.

$$
\begin{aligned}
p_{interp} &= (1-\alpha)p_{source} + \alpha p_{target} \\
\alpha &= \begin{cases} 1, & \text{where } r \leq r_{swap} \\ 0, & \text{where } r > r_{swap} \end{cases}
\end{aligned} \tag{5.1}
$$

where r_{swap} is the boundary resolution. Note that the role of source model and target model can also be reversed to determine which model the generated images will be closer to. Figure 5.3 intuitively illustrates the mechanism.

Using the transferred generator and the source generator, experiments were conducted for each possible resolution in this project. The results are shown in Figure 5.4. Based on the results, $r_{swap} = 32$ with the swapped position of generators achieved the best result. The human facial structure is well preserved with some furry-like features added, albeit there are still some unsatisfactory defects.



(a) (b)

(c) (d)

Figure 5.2: (a) Initial images (b) Config D (c) Config E (d) Config F

5.3 Discussion

No quantitative evaluation is performed in this project. This is because furry image generation is a rather subjective process and can be hardly evaluated with traditional lower-means-better or higher-means-better metrics.

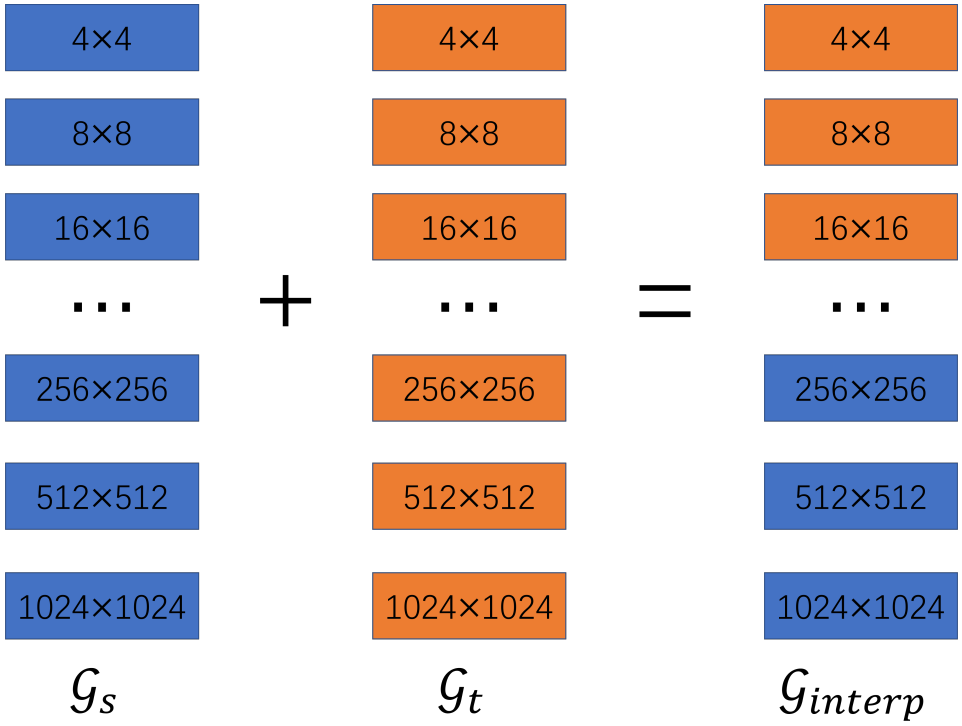
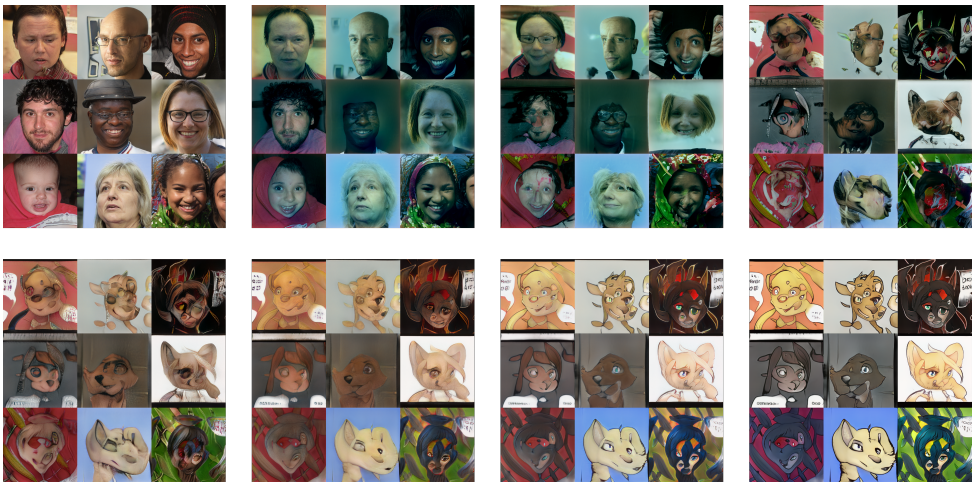
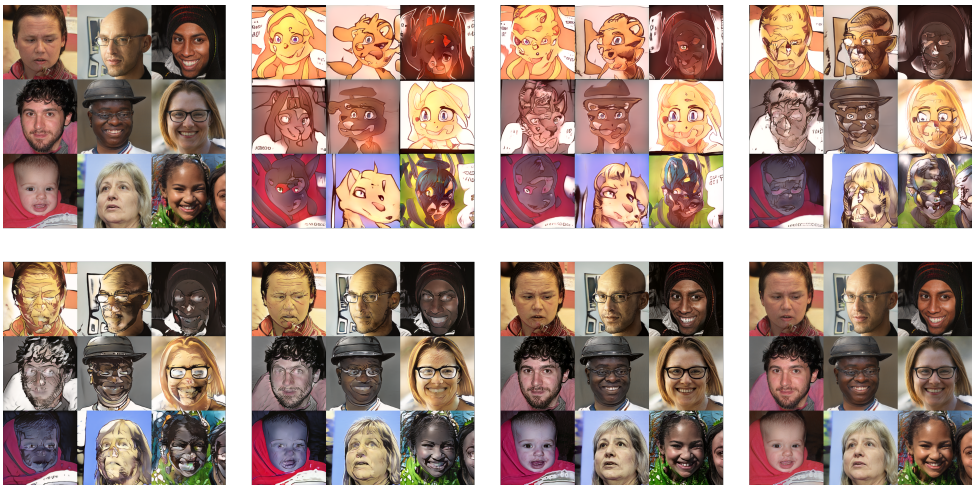


Figure 5.3: Layer swapping. The blue layers are from the source generator while the orange layers are from the target generator. Low resolution layers can also be from the source model with high resolution layers from the target model. The interpolated model will generate images with interpolated features.



(a)



(b)

Figure 5.4: (a) Results with $r_{swap} = 4$, 8, 16, 32, 64, 128 and 256. The first image in the upper left corner is the input image. (b) Results using the same r_{swap} as (a), but the postion of the source generator and the target generator are swapped.

Chapter 6

Conclusion

6.1 Conclusion

In this project, a generative adversarial network is trained with the help of transfer learning. The initial weights are loaded from a StyleGAN2 model pre-trained on the FFHQ 256×256 dataset. These weights are then trained on a new dataset, the E621 Faces Dataset. Two loss functions, structure loss and similarity loss, are involved in training apart from those used in the standard StyleGAN2 training framework. Layer swapping technique is also applied to the selected model and is effective. Consequences show that furry features are successfully transferred to the input image. However, the combination of those features with human faces are still imperfect and less satisfactory.

6.2 Future Works

Possible future works include:

- **Improve the quality of dataset.** Currently, there are noises e.g. dialogue boxes in the training images, which may bring disruptions in the generated images. Besides, compared to some other famous dataset like FFHQ, the eyes in the images are not aligned in the dataset used. Carefully curated images in the E621 Faces Dataset

may result in better consequences.

- **Fine-tune hyperparameters.** Currently, the default configurations from the source model is used for transfer learning due to cost reasons. The training was conducted on a rented high-end server to faster speed. Although the code can also run on my personal computer with batch size 2, it can take too long time to see the results, which is unacceptable within limited period for this project. Using a different set of parameters may yield better image quality.

- **Support conversion from plain images.** Currently, all images are generated from samples in $\mathcal{Z}$ space. Easier usage can be achieved with an additional encoding module to the current model. Some choises are available (Richardson et al. (2021), Huang and Belongie (2017)) but the effects are not tested for transfer learned latent spaces.

- **Allow furry image generation of specific species.** Currently, the model is unable to produce images of a certain species e.g. cat-like images. A multi-path structure proposed by Song et al. (2021) may be helpful in solving the problem, which has already demonstrated its capability of generating images in the chosen style.

Bibliography

arfafax (2020). E621 faces dataset. `https://github.com/arfafax/E621-Face-Dataset`.

Back, J. (2021). Fine-tuning stylegan2 for cartoon face generation. *arXiv preprint arXiv:2106.12445*.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heeger, D. J. and Bergen, J. R. (1995). Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238.

Huang, X. and Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020a). Training generative adversarial networks with limited data. In *Proc. NeurIPS*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020b). Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR.

Mo, S., Cho, M., and Shin, J. (2020). Freeze the discriminator: a simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*.

NVlabs (2021a). stylegan2-ada-pytorch. `https://github.com/NVlabs/stylegan2-ada-pytorch/blob/main/Dockerfile`.

NVlabs (2021b). stylegan2-ada-pytorch. `https://github.com/NVlabs/stylegan2-ada-pytorch`.

NVlabs (2021c). stylegan2-pretrained. https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada-pytorch/pretrained/transfer-learning-source-nets/ffhq-res256-mirror-paper256-noaug.pkl.

Pinkney, J. N. and Adler, D. (2020). Resolution dependent gan interpolation for controllable image synthesis between domains. *arXiv preprint arXiv:2010.05334*.

Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., and Cohen-Or, D. (2021). Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ross, A. and Doshi-Velez, F. (2018). Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29.

Sangkloy, P., Lu, J., Fang, C., Yu, F., and Hays, J. (2017). Scribbler: Controlling deep image synthesis with sketch and color. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6836–6845.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, G., Luo, L., Liu, J., Ma, W.-C., Lai, C., Zheng, C., and Cham, T.-J. (2021). Agilegan: stylizing portraits by inversion-consistent transfer learning. *ACM Transactions on Graphics (TOG)*, 40(4):1–13.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251.

Appendix

For presentation reasons, the full code is hosted on https://gitlab.scss.tcd.ie/zguo/kemogan. Please refer to the repository for more information.