Facial Image Infilling with Controllable Gender Expression Using Generative Adversarial Networks

BY

Victoria (Jin) Cheng

April 2020

Supervised by

Dr. Shalmali Joshi
PhD UT, Austin
Vector Institute

Dr. Marzyeh Ghassemi
PhD MIT CSAIL
University of Toronto, Computer Science and Medicine
Canadian CIFAR AI Chair, Vector Institute
NeurIPS Workshop Co-Chair

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Facial Image Infilling with Controllable Gender Expression Using Generative Adversarial Networks

BY

Victoria (Jin) Cheng

April 2020

Supervised by

Dr. Shalmali Joshi
PhD UT, Austin
Vector Institute

Dr. Marzyeh Ghassemi
PhD MIT CSAIL
University of Toronto, Computer Science and Medicine
Canadian CIFAR AI Chair, Vector Institute
NeurIPS Workshop Co-Chair

Contents

List of Figures

List of Tables

Abstract

Generative adversarial networks (GANs) have the potential to facilitate the creation of image datasets for research studies by providing the ability to synthesize artificial images. For facial image datasets, this includes images where one or multiple attributes of the face are tuned along a latent space. These models can also be extended to infill the facial portion of an arbitrary image by imposing a generated face onto any image with auxiliary information, such as the rest of the body or background, and create controlled images that solely modify attributes associated with the face in the image, while keeping the remainder of the image static.

This thesis will outline the implementation of a series of GANs that achieve controllable high-quality facial image generation by allowing for the tuning of gender expression in the synthetic face. The results of two infilling approaches, one using an encoder component of the generative adversarial network and one using a gradient descent optimization of the latent space, and their respective fidelities are highlighted. The infilling models are also extended for use on arbitrary images using a pre-trained model to detect the facial portion of an image, processing that portion of the image through the model, then replacing that portion with the model output.

Although gender expression was chosen as the conditional attribute for the models in this thesis, the conditional GAN models can be applied to any singular attribute label.

Using the models and techniques presented in this thesis, controlled datasets of facial images can be artificially synthesized to fit the needs of a research study, saving time and resources.

Acknowledgements

I would like to express my thanks to Marzyeh for giving me to opportunity to work with her and all the wonderful people in her lab. Thank you for your support and guidance, it has been an amazing learning opportunity and I have really enjoyed working on the project.

I would also like to express my appreciation to Shalmali, for her encouragement, helpful suggestions, and insightful comments which were of tremendous help to overcoming the roadblocks faced during this project.

Thanks to the other members of the lab, especially Vinith, Bret, Nathan, Amy, and Taylor for their emotional support and friendship.

Finally, to the amazing people of the Ops Team at Vector, my deepest gratitude for the time spent teaching me about the computing tools and resources, as well as the assistance with debugging the overwhelming amount of technical issues that occurred during the course of this project.

- *Vic*

Chapter 1

Introduction

In facial image-based user studies, much of the work is concentrated in the manual creation or gathering of the images. The formulation of these pictures involves tasks such as searching for the portrait subjects, gathering media releases from each of the models in the photos, and taking the photos, all while maintaining controlled attributes for the study. With the abundance of images available on the internet today, we would like to be able to leverage this supply of images for such studies while respecting copyright laws and the personal identity privacy of the subjects in the photos. As well, while images are plentiful online, it remains difficult to find a set of images ideal for research studies, where only one aspect of the photo is changed while the the other aspects are controlled. In order to create these desired images more easily, we would like to be able generate them artificially.

Much progress has recently been made with respect to artificially generated images with the invention of generative adversarial networks (GANs) [16], which have succeeded in generating artificial facial images [16, 21], including images where one or multiple attributes of the face are tuned along a latent space [31] or where a generated face is imposed on top of an existing one [5].

As this technique would allow for the tuning of an image along the dimension of a single attribute, such as perceived gender expression, using these generative models would allow for more controlled facial image-based studies. These models would also allow us to impose a generated face onto any image with other desired attributes, such as clothing or the rest of

the body, and perform controlled studies that solely modify attributes associated with the face in the image, while keeping the remainder of the image static.

This thesis focuses on the implementation of a conditional infilling general adversarial network that allows for the completion of facial images with controllable gender expression.

The project is composed of four sequential building blocks:

1. Creating a generative adversarial network that is able to synthesize realistic facial images.
2. Creating a conditional version of the generative adversarial network that is able to tune the expression of gender in the face.
3. Implementing an infilling version of the generative adversarial network, which is able to realistically impose a generated face onto a given masked facial image.
4. Extending the infilling algorithm to detect and replace the facial portion of a given arbitrary image, which may contain auxiliary information.

The models presented in this thesis primarily concentrate on controlling the expression of gender in the faces according to the MALE label from the dataset, but may be extended to other attributes. All of the models are trained of the CelebA-HQ dataset [21], and biases from the dataset are reflected in the learned attributes of the model.

Chapter 2

Background

Controllable facial image infilling relies on a combination of developments that have been made in various different aspects of generative models and image completion techniques. The technology must not only be able to realistically complete the image with the appropriate structure and texture in accordance with the rest of the global structure of the image, but the face that is generated must be controllable within the latent space.

Here, we will introduce the relevant background knowledge in each of the relevant fields related to these tasks, focusing on progress made with regards to image completion and controllable generative models.

2.1 Traditional Image Infilling Techniques

Figure 2.1: Historical image infilling examples. First row: Original images. Second row: Infilled images. [51]

The goal of image infilling is to replace missing or unwanted target regions of an image with synthetic content to create a result that looks natural and realistic [2]. Its applications include editing the visual content of an image to remove damaged portions or objects, or extending the visual content of an image [2]. It is a technique that was initially prevalent in the area of art restoration. Some examples are shown in Figure 2.1.

The basic strategy behind traditional image infilling relies on the composability and reusability of visual patterns [14]. Any arbitrary image can be represented as a hierarchy of parts. These parts are able to be combined and reused to form an assortment of perceived meaningful combinations. Image infilling systematically exploits these constituent parts of the image along with contextual constraints in order to create content for the target area of the image to be filled. This process was extremely manual [2, 51], even with the advent of digital photo editing [4, 51]. However, with the development of algorithms for automatic digital infilling, the process has become much faster.

Early image completion methods relied on non-learning based approaches such as patch matching or texture synthesis [6, 1, 7]. Other algorithms propagated information from known neighbourhoods in the image, either locally or globally in order to complete the image [11, 2].

Initial texture synthesis algorithms required the user to specify the texture to be copied into the specified region [2]. If the user wanted to fill a region of the image with multiple textures, they had to manually segment that region and select corresponding textures for each of the sections.

These algorithms have now been automated to the point where the user input only has to specify the region infill [2, 11]. The algorithm would then find examples of plausible replacement patches based on context similarity in the known space of images, either from the given image [11] or an external database of images [17], and stitch them together to fill in the selected region. This is done by estimating the conditional distribution of a pixel given its neighbours [11], synthesizing the texture by sampling from the conditional probability distribution.

These initial techniques were restricted to filling in textures, and could not reproduce general structures, though with an interactive editing algorithm that used additional input from the user guiding the completion of the structure [1], these algorithms were able to be greatly improved.

Texture synthesis is not the only important aspect of infilling. In order for the entire image to look coherent, the structure, colour, and texture of the gap must be consistent with the area surrounding the region. In order to do this, the global structure of the picture must be ascertained to determine how the gap is filled [2].

Local infilling algorithms look at the information at the boundaries of the region to be completed in order to ensure that the content that is synthesized for the gap is consistent with the content at the regions [2, 3]. The issue with local algorithms is that they will produce the same completion for any two regions that have the same boundary, regardless of the content in the rest of the image [32]. Global algorithms address this issue by using image-specific information to infill the regions [24], such as building an exponential family distribution over images that is based on the histograms of local features [32]. Other algorithmic approaches to infilling involved automatically adjusting the target region size to find the best shape of the source patch for the target in order to assure smooth blending [26, 48, 17].

The disadvantage of these traditional algorithms is that the all rely on the assumption that similar patches to the missing content can be found in the given images, which may not

necessarily be true. Although some algorithms attempt to augment the search space with additional photometric and geometric transformations [7], this is still a strong assumption that does not hold in all cases. Therefore, these algorithms may be effective in tasks involving generic scene completion where there is a high chance of finding similar patterns in the given images, but are ineffective with tasks involving the completion of a specific object, such as a face.

In the cases of the latter task, it is essential to consider the semantics and the structure of the object, and may require the synthesis of new content, not just the reproductions of already existing content. Since they do not consider this information, these traditional algorithms typically fill in the regions with content that is inconsistent with the global structure of the object [39, 19, 50], such as removing key facial features such as eyes, or filling in the incorrect feature, such as putting a mouth in the place of an eye.

Recent developments with learning-based approaches have been able tackle this problem by learning the data distribution behind the given images, allowing them to generate novel content. These global content-aware, learning-based infilling algorithms are based on developments in artificial neural networks and generative adversarial networks.

2.2 Artificial Neural Networks

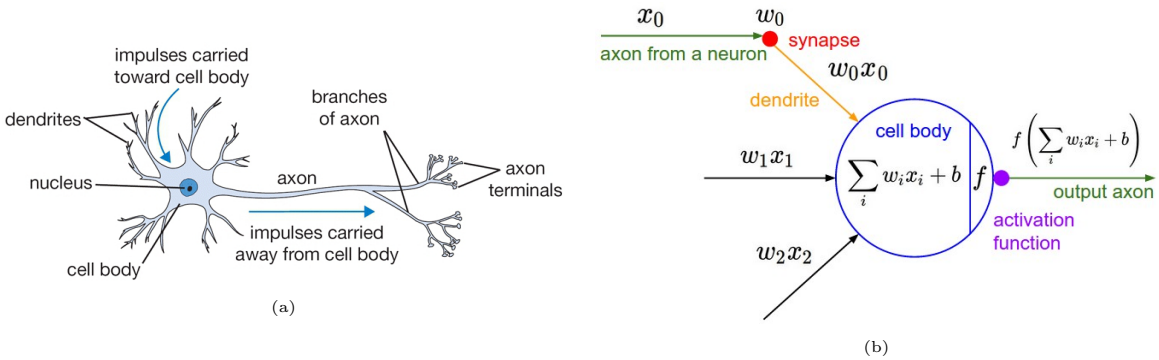


Figure 2.2: A cartoon drawing of a biological neuron (a) and its mathematical model (b). [20]

Artificial neural networks are a biologically-inspired machine learning tool that mimic the interconnection of neurons in an organic brain with layers of connected nodes [20]. Ad diagram of a biological neuron and its mathematical representation is shown in Figure 2.2. A network is trained to identify an object or a person by showing it millions of examples, and adjusting the weightings between the layers and the nodes using a technique called backpropagation [28]. The weights between the layers and the nodes affect the predictions of the network. The networks contain activation functions, which are nonlinearities that allow networks to go from linear function mappers to universal function approximators [20]. Typically, the network is used to classify or identify object in a given image, but can also be used for tasks such as image segmentation or image generation.

2.3 Convolutional Neural Networks

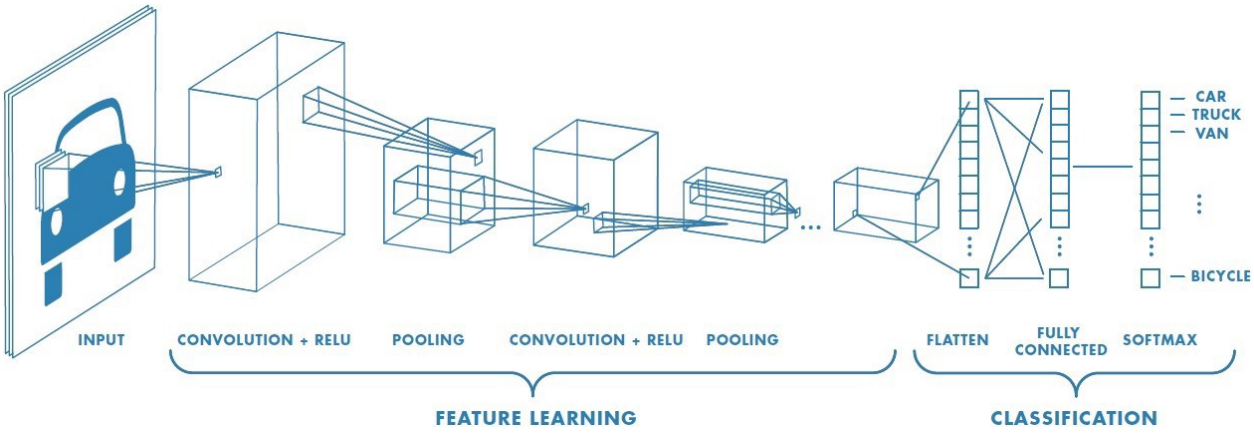


Figure 2.3: Simplified Example Architecture of a Convolutional Neural Network [41]

Tasks involving images typically use a neural network model called a Convolutional Neural Network (CNN) [29, 30], which uses convolution operations on the data to learn image features. An example of a CNN is shown in Figure 2.3. CNNs have been shown to be particularly effective on image data tasks [25]. Compared to traditional fully connected neural networks, CNNs reduce the image information to a form that is richer and easier to process, which makes them much more scalable to larger and higher-resolution datasets.

Traditional neural networks with fully connected layers have an extremely large number of parameters. This makes it more difficult to train these networks, as they have a larger capacity and thus require a larger training set in order to learn the required data [41]. As well, the learned outputs are often not invariant to translations or scale.

CNNs use the replication of weight configurations across the input in order to achieve robustness to these geometric distortions [29]. These weight configurations, or kernels, restrict the receptive fields of hidden units in the networks, which encourages the extraction of local features.

Through the application of different kernel filters, CNNs are able to detect various different features in the image [41]. These low-level features can then be combined to form higher level features, which can then be used for tasks such as determining the object in an image.

To produce a convolved feature, we perform matrix multiplication between the kernel and

a portion of the image. Then we shift the kernel by a specified stride length, and repeat the operation. This proceeds until we have traversed over the entire image. In order to maintain the dimensionality of the resulting operation, we add padding to the original image so that the output of the convolution is the same size as the input. Between each convolution layer, the model also performs average pooling in order to reduce the resolution of the feature map and reduce the sensitivity of the model to geometric distortions.

CNNs have been shown to able to successfully complete image-related tasks such as classification [25], object detection [15], and image segmentation [36, 38] due to their ability to capture hierarchical feature information in high-dimensional image spaces. They have also been used in generative models [16] due to the same advantages.

Therefore in order to generate high-resolution images, we will use a CNN within our image generation model.

2.4 Image Generation Models

Generative models are a type of neural network that are designed to synthesize novel data samples from the same distribution as the training data. They are able to do this without the need for any data labelling because they learn the high-dimensional distribution of the data. They are able to generate a variety of data types according to the given input data, most relevantly images for our case.

The most prominent approaches for image generation models can be split into 3 categories: variational auto-encoders (VAEs)[22, 23], autoregressive models [45, 44], and generative adversarial networks (GANs) [16, 21].

2.4.1 Variational Auto-Encoders

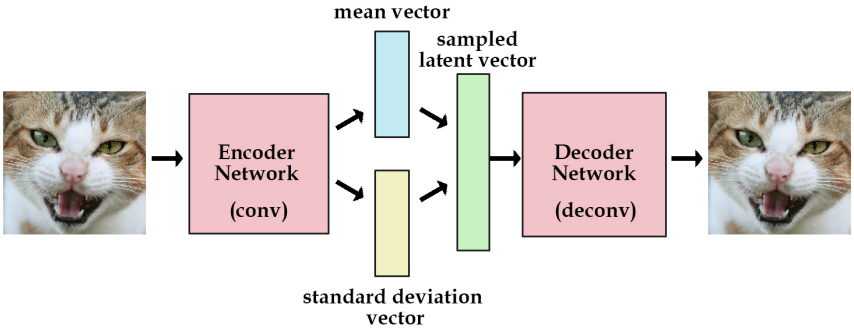


Figure 2.4: An example of a VAE Network. [13]

VAEs (Figure 2.4) are composed of two components: an encoder and a decoder. The encoder attempts to parametrize the input into mean and standard deviation variables that represent the latent space of the distribution, while the decoder samples from the resulting distribution to generate images [22, 23].

VAEs are trained by comparing the reconstruction error between the output image and the input image [22]. Both the encoder and decoder can be represented using neural networks, which makes them easy to train using backpropagation.

Unfortunately, the images that are produced by VAEs tend to be visibly blurry since they rely on Markov Chains, and require the distribution to be somewhat blurry in order for the chains to be able to mix between modes in the distribution. This makes VAEs a poor option for generating higher resolution images.

2.4.2 Autoregressive Models

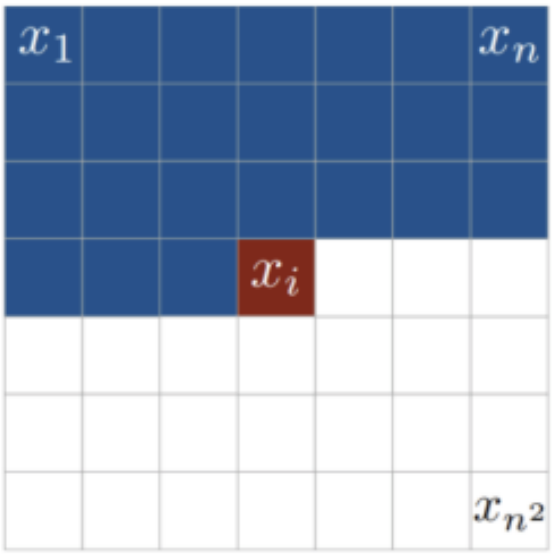


Figure 2.5: Process of generating individual image pixels using an autoregressive model [12]. x_i is the current pixel being generated. The blue pixels are all the previously generated pixels, and the white pixels are all the pixels that have not yet been generated.

Autoregressive models treat an image as a sequence of pixels and directly model the conditional distribution over pixels, which allows them to produce sharp images than VAEs [45, 44]. The probability of each pixel's intensity is conditioned on all of the previous pixels (represented pictorially in Figure 2.5):

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

Unfortunately, this means that they lack latent representations, and therefore are unable to control different attributes in the generated images. Due to the pixel-wise generation of

the output, autoregressive models are also very slow to evaluate and tend to be limited by memory and computation requirements to low-resolution images [44].

2.4.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [16] are created using an adversarial process where two models are simultaneously trained:

- A generative model that attempts to learn the distribution of the training data and synthesize images from latent vectors that are indistinguishable from the training data.
- A discriminative model that estimates the probability that a sample came from the same distribution as the training data.

The training procedure uses backpropagation and follows a minimax game, where the generator is trained to maximize the probability of the discriminator making a mistake. Unlike VAEs or autoregressive models, GANs are able to generate sharp images from latent vectors.

Therefore, in order to produce high-quality images with controllable attributes in latent space, the focus of this thesis will be using GANs for image generation.

2.5 Training Generative Adversarial Networks

In the generator-discriminator structure of a GAN, the generator attempts to produce samples from a distribution that is as close to the training distribution as possible. GANs are able to learn this distribution using gradient backpropagation through both networks.

2.5.1 Problem Formulation

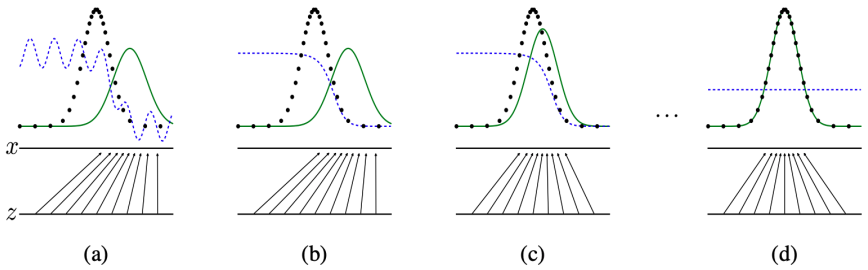


Figure 2.6: The training process of a GAN. [16] The discriminator is represented by the blue line, the generator is represented by the green line, and the true data distribution is represented by the black line. The arrows represent the mapping from random noise to the generator distribution. **(a)** The initial state of the distributions associated with the model. **(b)** The discriminator is trained to discriminated between the true samples and the samples synthesized by the generator. **(c)** The generator uses the gradient of the discriminator to adjust its output distribution. The training steps in (b) and (c) are repeated until convergence. After the model has converged in **(d)**, the discriminator will be unable to differentiate between generated samples and true samples.

The minimax game setting used to train GANs is outlined as follows [16].

Let x represents the input data, define $P_G(x)$ as the generator distribution and $P_{data}(x)$ as the true data distribution repetitively.

Let $P_{noise}(z)$ represent the prior on the random noise variable that is used as the input to the generator.

The generator, G, is a differentiable function represented by a neural network with parameters represented by θ_G. $P_G(x)$ is parameterized using a generator network G. G transforms a given latent random noise variable z into a sample through its mapping from input noise to the data space, $G(z, \theta_G)$. The generator network is trained to recover the training data distribution, or in other words, G is trained to match $P_G(x)$ and $P_data(x)$.

The adversarial discriminator network is represented by D. D is also a neural network

that maps between the input data, and the binary classification of real or fake, $D(x, \theta_D)$, where θ_G are the parameters of the network. D is trained to maximize the probability of assigned the correct labels to the training examples and the samples synthesized by the generator network.

The loss function used to train the network is the adversarial loss function, $\mathcal{L}_{adv}(G, D)$. The minimax game used to train both networks simultaneously can be defined as follows:

$$\min_G \max_D \mathcal{L}_{adv}(G, D) = \mathbb{E}_{x \sim P_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim P_{noise}(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

Using this loss function, we can see that for a given generator G, the optimal discriminator is:

$$D(I) = \frac{P_{data}(I)}{P_{data}(I) + P_G(I)} \tag{2.2}$$

This training process is illustrated in Figure 2.6. After the training is completed, the discriminator network can be discarded, as only the generator network is required to generate image samples.

2.5.2 Challenges with Training GANs

Unfortunately, since the generator and discriminator are trained simultaneously, the training process for GANs is usually quite unstable [42]. Mode collapses can occur during training, where the discriminator overshoots, leading to exaggerated gradients in both networks that produce undesirable samples[21]. As well, when the discriminator can easily tell between real and fake images, we encounter a vanishing gradient, which makes it difficult to progress with training.

These issues are what make it difficult to generate high resolution images with GANs, especially since the higher-resolution makes it easier for the discriminator to tell generated images apart from real images. High-resolution images also cause memory constraints, requiring the use of smaller minibatches during training that further compromising training stability [21].

Recent work surrounding GANs have developed various methods to stabilize training,

such as fractional-strided convolutions [40], batch normalization [40], Laplacian pyramid frameworks [9], weight normalization [49], and progressive growing of GANs [21].

We will focus on the progressive growing technique in our training, which achieves better training performance by growing the generator and discriminator simultaneously from low to high resolution. This method was shown to have excellent quality of results in the generated samples, and also demonstrated faster training time than the other methods [21].

2.5.3 Progressive Growing of GANs

With the Progressive Growing of GANs, Karras et al. proposed a training method that grows both the generator and discriminator progressively, starting from low resolution images and adding new layers that introduce higher resolution details as training progresses. This was shown to speed up training and improve the stability of the model in higher resolutions. The incremental nature of the training means that the network does not have to handle information across all image scales simultaneously, and can first learn holistic image structures, and then focus on producing finer details.

In this technique, the generator and discriminator network are mirrors of each other and always grow synchronously. New layers are faded in smoothly during the growing phases, and all existing layers in both networks remain trainable throughout the training process. This procedure was demonstrated to stabilize training sufficiently to reliably synthesize mega-pixel scale images.

This progressive growing process is discussed in greater detail in Section 3.2.3.

2.6 Existing Generative Image Infilling Approaches

GANs can also be used as the basis for generative learning-based image infilling approaches. These techniques differ from the traditional infilling techniques in that they are able to synthesize novel content from the latent space of the data distribution, and have been shown to successfully generate regions that are more consistent with the global structures of objects.

There are several different learning-based approaches to the infilling problem. Pathak et al. generated the contents of an arbitrary region of an image conditioned on the surroundings of the region using context encoders and decoders [39]. The encoder in the model mapped the context of masked images to latent representations and the decoder translated them to natural content images. Through training using reconstruction loss and adversarial loss, this technique was demonstrated to capture the appearance and semantics of visual structures. However, the generated image patches were often blurry and had inconsistencies along the border of the regions.

Ye et al., introduced a technique to produce sharper images by using a pre-trained generative model to search for the closest encoding of the corrupted image in the latent image manifold, and the pass this encoding through the network in order to infer the missing content [50]. This success of this technique depended heavily on the quality of the pre-trained generative models, and also involved post-processing steps in order to blend the output of the generator with the original input image.

Other generative models used both a global and a local discriminator combined with post-processing in order to generate patches that are both globally and locally consistent [19, 33]. The global discriminator looks at the entire image and assesses the coherence as a whole and the local discriminator looks at a small area centered around the completed region to ensure the local consistency of the generated patches. The generator network is then trained to fool both discriminator networks. The images that have been generated by these models are consistent and they are able to handle masks of arbitrary shapes, but the images are of low resolution.

Although the infilling approaches listed above have found some success with low resolution images, no work has yet been done to pursued attempts to generate extremely high-quality

images. Therefore, these models could potentially benefit from applying the techniques developed for standard GANs as discussed in Section 2.5. As well, all of the approaches require post-processing steps in order to blend the infilled region, which slows down the image generation process.

2.7 Generative Models with Controllable Attributes

The generative methods discussed above focus on generating realistic content from the data distribution. However, these models are unable to control attributes of the synthesized content. In order to create a controllable conditional model, the GAN structure must be modified to feed the conditional label to both the generator and discriminator, in addition to the image data. After training, these conditional GANs [37] can then be passed attribute vectors to control the properties of produced images explicitly.

Using the conditional label, the objective function of the minimax game the becomes:

$$\min_G \max_G \mathcal{L}_{adv}(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_{noise}(z)}[\log(1 - D(G(z|y)))] \tag{2.3}$$

where y is the label that we would like to condition on.

During training, the discriminator now needs to correctly classify both the real/fake label of the data, as well as the conditional label. This can be done by adding an additional layer to the discriminator for classification[38], adding a separate feature classifier network in the model [31], or adding an additional head to the existing discriminator structure that predicts the attributes[5].

This thesis will use a discriminator model with an additional head classifier to control attributes in a facial generation model.

2.8 Existing Facial Image Completion Techniques with Multiple Controllable Attributes

Facial image completion with multiple controllable attributes will draw upon many of the developments in artificial intelligence presented in the previous sections. However, few models exist at the intersection of these methods. Completion models are often not designed to allow for manipulation of the properties of the synthesized contents, and the existing conditional GANs are not able to perform infilling on high quality facial images.

Chen et al. have proposed a fully end-to-end framework that trains generative adversarial networks progressively from low resolution to high resolution with multi-dimensional conditional vectors [5]. The model is able to complete masked faces in a single forward pass without any post processing. It leverages the techniques of progressive growing of GANs along with auxiliary feature prediction and attribute prediction discriminator heads in order to synthesize infilled images. The architecture of the model is based on U-Net [38], with skip connections, in order to better reproduce the orginal context region. The model also employs new loss functions that induces the network to blend the synthesized content with the contexts in a realistic way.

This thesis will demonstrate the results of an implementation of this proposed technique in Section 3.4.

Chapter 3

Approach

In this section, we outline our approach to generating high-quality facial images using convolutional progressively growing GANs.

First, we provide an overview of the creation process of the CelebA-HQ dataset that was used for training. We also highlight various aspects of the dataset that contribute to biases in the resulting outputs from the models.

Then, for the base GAN model, the conditional model, and the infilling models, we provide the network structure, the problem formulation, and the training process, along with the loss functions that were used during training.

Finally, we describe the process of extending the infilling models to be applied to arbitrary images.

Detailed model architectures have been provided in Appendix B.

3.1 Creating the CelebA-HQ Dataset

Figure 3.1: Samples of images from the CelebA-HQ dataset.

Attribute	P(MALE|Attribute)	P(FEMALE|Attribute)	P(Attribute|MALE)	P(Attribute|FEMALE)
Blond Hair	215/5126	4911/5126	215/11057	4911/18943
5126 Images (17.09%)	(4.19%)	(95.81%)	(1.94%)	(25.93%)
Eyeglasses	1226/1468	242/1468	1226/11057	242/18943
1468 Images (4.89%)	(83.51%)	(16.49%)	(11.09%)	(1.28%)
Heavy Makeup	17/13708	13691/13708	17/11057	13691/18943
13708 Images (45.69%)	(0.12%)	(99.88%)	(0.15%)	(72.27%)
No Beard	5395/24328	18933/24328	5395/11057	18933/18943
24328 Images (81.09%)	(22.18%)	(77.82%)	(48.79%)	(99.95%)
Smiling	4363/14092	9729/14092	4363/11057	9729/18943
14092 Images (46.97%)	(30.96%)	(69.04%)	(39.46%)	(51.36%)
Young	6654/23368	16714/23368	6654/11057	16714/18943
23368 Images (77.89%)	(28.47%)	(71.53%)	(60.18%)	(88.23%)

Table 3.1: Attribute Label Distributions for CelebA-HQ

The subsequent models described in this section were all trained on the CelebA-HQ dataset [21].

The CelebA-HQ dataset is a high-quality facial image version of the CelebA dataset [35], consisting of 30000 images at 1024×1024 resolution. The dataset was generated from the original CelebA dataset [35], and is overall a cleaner dataset with fewer artifacts and more consistent image quality.

Several image processing steps are applied to center the images on the facial region and to ensure consistent image quality [21], such as:

- Pre-processing each image using a convolutional auto-encoder to remove jpg image artifacts.
- Modifying the resolutions of the images using an adversarially trained super-resolution network.
- Padding and filtering images where the facial region extends outside the image.
- Cropping the image to size based on the dimensions of the facial landmarks in the image.

During training, the dataset is also augmented by mirroring the images (for a total of 60000 training images). Samples of images from the final dataset are shown in Figure 3.1.

In addition to the images, the CelebA dataset also contains 40 labelled attributes for each image. Using the correspondences between the CelebA and CelebA-HQ images [21], we matched each of the corresponding labels to each of the images in CelebA-HQ. Table 3.1 shows the dataset statistics for a subset of the labels in CelebA-HQ, split by the MALE and FEMALE labels. Appendix A contains the complete statistics about the dataset.

Although the dataset does not have explicit labels for ethnicity or skin tone, a quick overview of the images in the dataset (such as the examples shown in Figure 3.1), show low ethnic diversity amongst the faces in the dataset.

These biases in the dataset will be learned and reflected in the outputs of the models. For example, a greater portion of the females in the dataset have heavy makeup, which will cause the conditional models to learn to associate that attribute to female faces.

3.2 Base Generative Model

This section outlines the basic problem formulation of synthesizing images using Generative Adversarial Networks (GANs), and presents the proposed network structure and training process for a facial image generation model. This model will serve as the base case that the remainder of the models presented will build upon.

3.2.1 Base GAN Network Structure

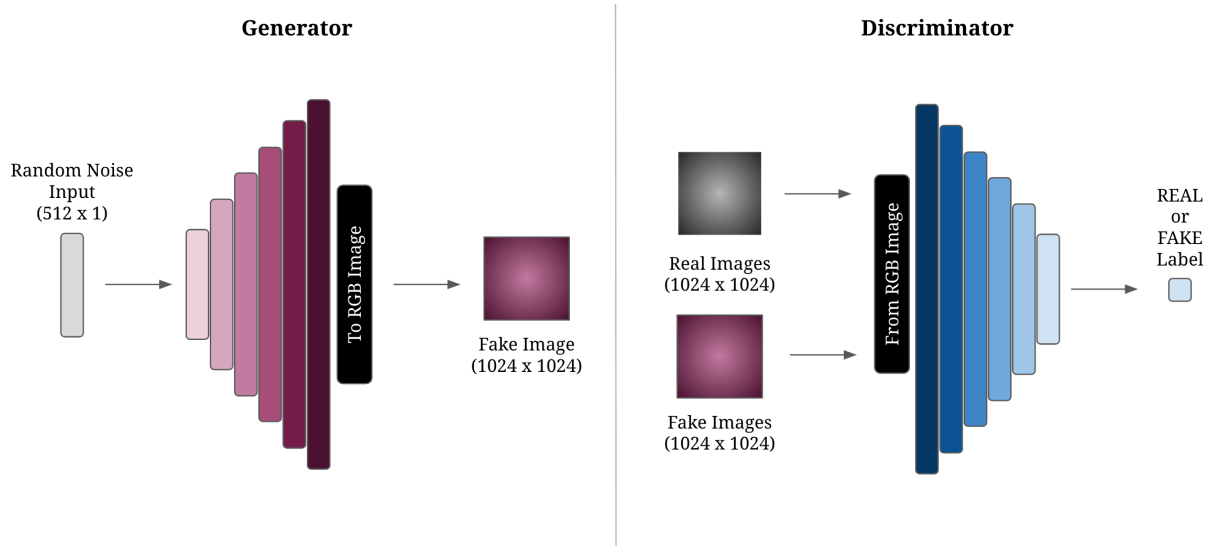


Figure 3.2: Network structure of the generative adversarial network that was used to synthesize the images. For the detailed network structure of the model, please see Appendix B.

The general network structure is shown in Figure 3.2. This model consists of a generator and a discriminator. The generator takes in a random noise vector of size 512×1 and outputs generated images of size 1024×1024. The discriminator takes in both real and fake images of size 1024×1024 and outputs a binary prediction for whether or not the image is real or fake.

Each of the generator layers shown in Figure 3.2 consist of a nearest neighbour interpolation, a convolution, and a LeakyReLu non-linearity. Each of the discriminator layers shown in Figure 3.2 consist of a convolution, a LeakyReLu non-linearity, and an average pooling operation.

3.2.2 Base GAN Problem Formulation

In this section, we reformulate the problem presented in Background Section 2.5 on GANs for the specific case of generating our desired facial images.

Denote the image lattice as Λ and let I_Λ be an image defined on the lattice Λ. The model will generate an image to fill the lattice.

Denote x as the training data (images from CelebA-HQ), and z as the random noise variable.

Let $G(z, \theta_G)$ and $D(x, \theta_D)$ represent our generator model and discriminator model respectively. The generator takes in the random noise variable z and the generator parameters θ_G and outputs a fake image, x_{fake}. The discriminator takes in a piece of data x, which can be a real image, x_{real}, or a fake image created by the generator, x_{fake}. The discriminator outputs a binary prediction for whether the given piece of data is real or fake.

Let $P_G(x)$ represent the data distribution of the outputs of the generator and the $P_{noise}(z)$ be the prior on our random noise variable. In this case, the random noise vector z will be size 512×1 and sampled from standard Gaussian distributions. The generator maps $z \sim P_{noise}(z)$ to a point in the distribution $P_G(x)$. We will denote the distribution of the real data as $P_{data}(x)$.

The goal of the training process for this model is to train $G(z, \theta_G)$ to match $P_G(x)$ to $P_{data}(x)$, and to train $D(x, \theta_D)$ to correctly map between x and the binary classification of real or fake.

The gradient descent algorithm will be motivated by the adversarial loss as defined in Section 2.5, which we have duplicated here:

$$\min_G \max_D \mathcal{L}_{adv}(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_{noise}(z)}[\log(1 - D(G(z)))]$$

3.2.3 Base GAN Training Process

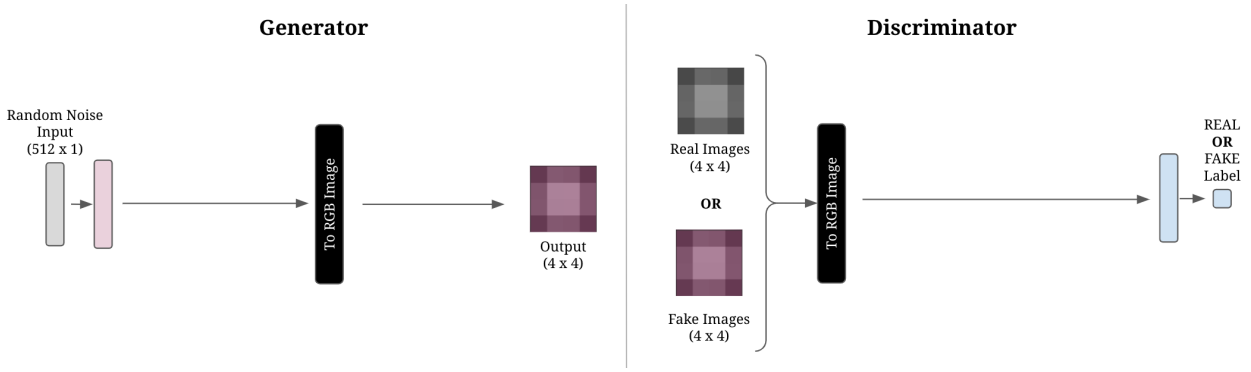
In order to improve training speed and maintain stability during the training process of this model, we will implement a progressive growing technique [21].

The progressive growing of a GAN model is comprised of 2 phases, a growing phase and a stabilization phase.

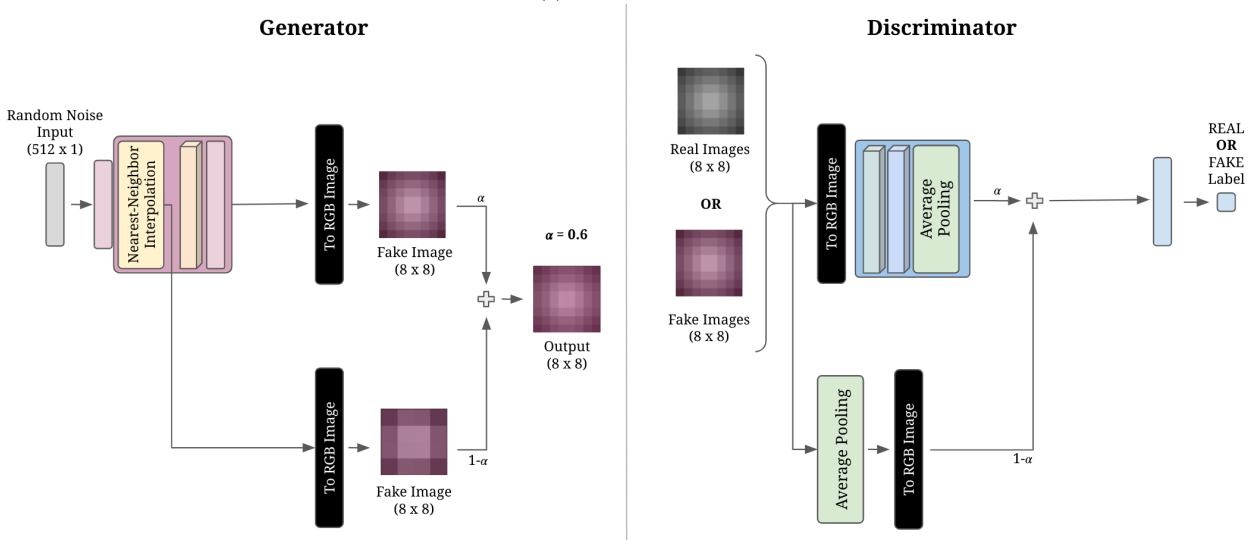
As illustrated in Figure 3.3, after the model's current layers have been stabilized we can commence the growing phase for the next layer. In order to prevent sudden shocks and high gradients to the existing trained layers, the layer to be added is faded in gradually. This is done through the linear interpolation between the output of the previous layer and the output of the current layer, which is controlled by parameter α. During the growing process α gradually increases from 0 to 1. After α reaches 1, we have completed the growing process and can proceed to train the current structure of the network in order to stabilize the newly added layer.

This growing and stabilizing procedure is repeated for each of the layers in the generator and discriminator. The generator and discriminator are grown synchronously. After all of the layers in the model have been added, we continue to train the model at the highest resolution until the results of the output are satisfactory.

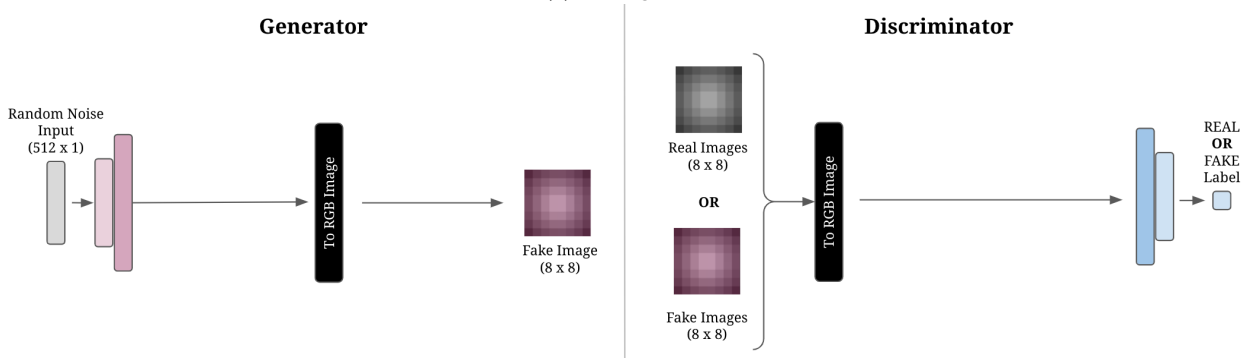
Empirically, training the model for 600000 images each during both the growing phase and the stabilizing phase for each layer, then proceeding to train the full model for a total of 12 million images produces satisfactory results.



(a) Stabilized Model



(b) Growing Phase



(c) Stabilization Phase

Figure 3.3: An example of the growing and stabilization procedure [21] applied to grow our network from 4×4 to 8×8 resolution. (a) The existing layers in the model have been stabilized. (b) During the Growing phase, the new layer is faded in slowly. (c) After the layer has been faded in, it is trained for additional iterations to stabilize it.

3.3 Conditional GAN Model

The conditional version of the GAN model builds on top of the base GAN model discussed in Section 3.2, with additional modifications in order to integrate labelled attributes into the training process and synthesized content.

3.3.1 Conditional GAN Network Architecture

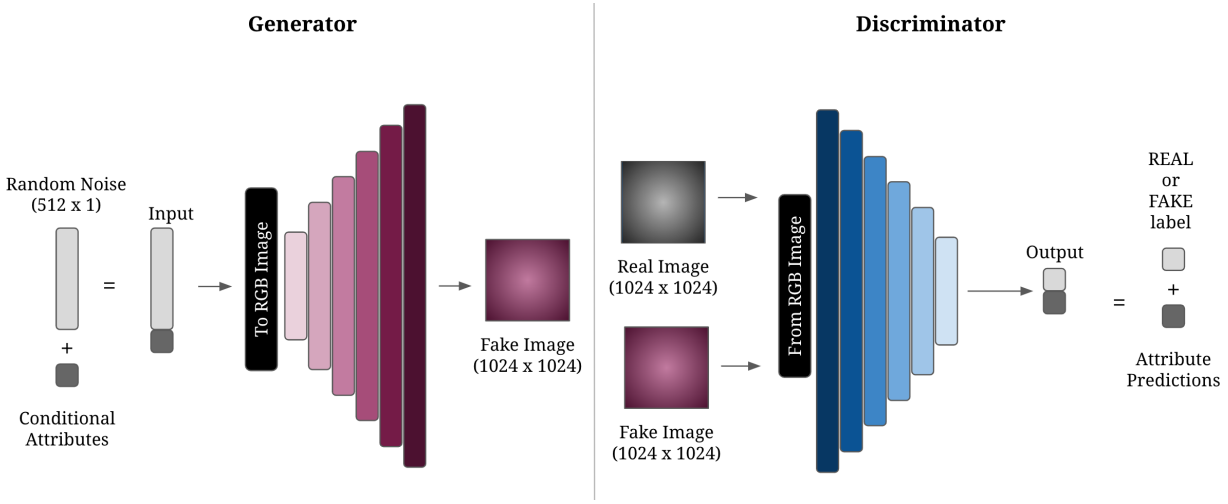


Figure 3.4: Network structure of the conditional generative adversarial network.

The modified network architecture is shown in Figure 3.4.

For this model, the input of the generator has been modified to be a concatenated variable consisting of the latent variable and the conditional label. The generator is trained to output a realistic fake image that matches the attribute label provided.

The output of the discriminator is modified to add an additional head classifier that predicts the conditional label of the given image.

3.3.2 Conditional GAN Problem Formulation

The problem setup is similar to that of the base GAN model in Section 3.2, except now the input data will also include a label, y. Most of the experiments for this thesis were done

using a single binary label for gender expression, with the value 0 representing FEMALE and 1 representing MALE. However, the conditional model can be extended to labels of arbitrary sizes.

Now, instead of training the generator to produce an arbitrary piece of data, we want to train it to output a piece of data that also matches a given label.

The adversarial loss function will be modified to account for the conditional component, as discussed in Section 2.7:

$$\min_G \max_D \mathcal{L}_{adv}(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_{noise}(z)}[\log(1 - D(G(z|y)))]$$

We have used sigmoid cross-entropy loss for the loss between the predicted label and the desired label.

3.3.3 Conditional GAN Training Process

The training process for the conditional version of the model is very similar to the training process for the non-conditional version, including the progressive growth process. The only modification in the training between this model and the non-conditional model is the change of the loss function to the one conditional one above.

3.4 Infilling with Encoder GAN Model

The final iteration of the model allows us to generate infilled images from masked images by adding an additional encoder component in the generator structure of the base GAN. This network also requires modifications to the input dataset by adding masks for each image, as well as additional loss functions in order to encourage the model to infill the masked area.

3.4.1 Infilling GAN Network Structure

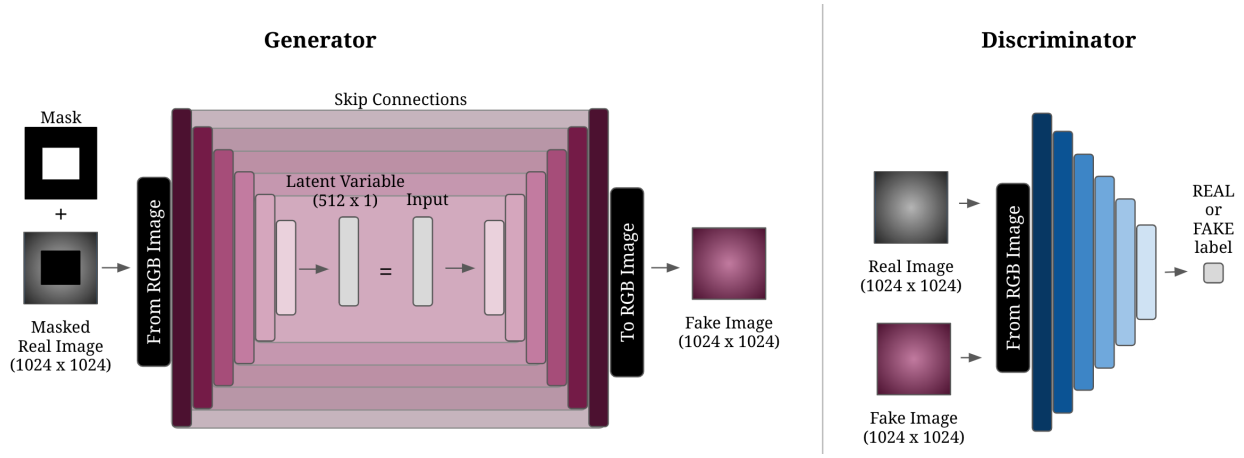


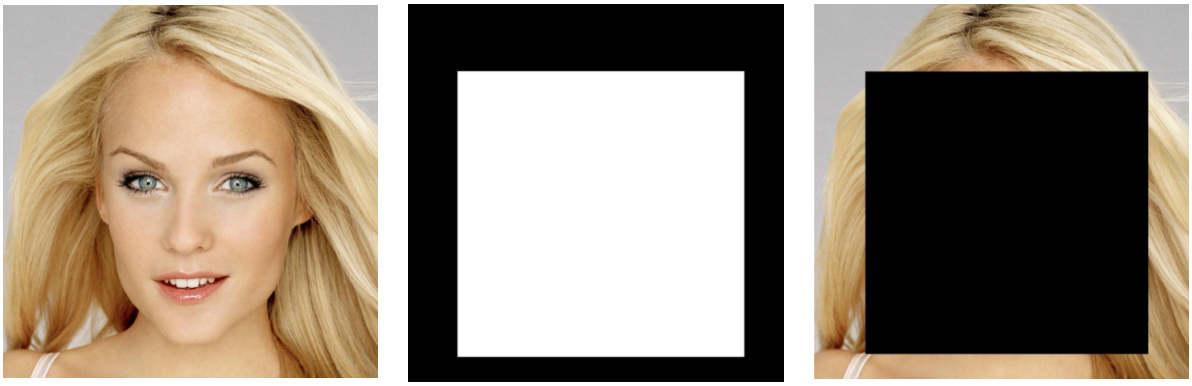
Figure 3.5: Network structure of the conditional generative adversarial network.

The infilling model structure and training process are adapted from the proposed method by Chen et al., as discussed in Background Section 2.8.

The architecture of this network is adapted from U-Net [38], using skip connections in order to consolidate information across different scales, which allows us to reproduce the context region of the given image more accurately.

Figure 3.5 shows the network structure of the infilling model. Both a binary mask and the masked image are given as inputs to the generator, which aims to output an infilled version of the masked image. The discriminator attempts to distinguish between the generated images and the original unmasked images.

3.4.2 Infilling GAN Problem Formulation



(a) I_{Λ}^{real} (b) M_{Λ} (c) I_{Λ}^{obs}

Figure 3.6: An example of a real image, mask, and observed image.

The implementation of the infilling model will be pursued as follows.

Denote the image lattice as Λ and let I_{Λ} be an image defined on the lattice Λ. Partition the image using into two regions, denoting Λ_t as the target region to complete and Λ_c as the remaining context region of the image. These regions should be chosen such that $\Lambda_t \cap \Lambda_c = \emptyset$ and $\Lambda_t \cup \Lambda_c = \Lambda$.

M_{Λ} represents the binary mask image with all pixels in $M_{\Lambda_t} = 1$ and $M_{\Lambda_c} = 0$.

Let I_{Λ_t} be masked out so that the values of all channels within the mask are set to 0.

The objective is to generate a synthesized image I_{Λ}^{syn} that is close to the real image I_{Λ}^{real}, given the observed image with the target region masked out I_{Λ}^{obs}. During the image synthesis, $I_{\Lambda_c}^{obs}$ and $I_{\Lambda_c}^{syn}$ must be kept similar.

The synthesized image is defined by $I_{\Lambda}^{syn} = G(I_{\Lambda}^{obs}, M_{\Lambda}; \theta_G)$ where θ_G contains all parameters of the generator.

The generator is defined by the equation:

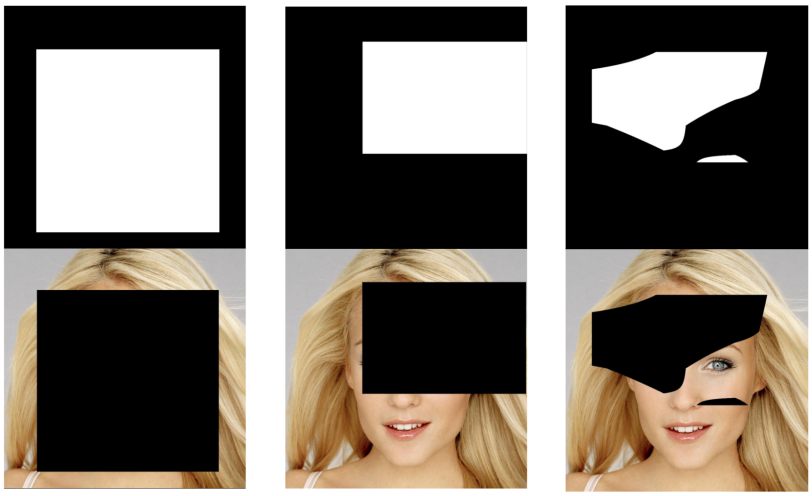
$$G(I_{\Lambda}^{obs}, M; \theta_G) = G_{dec}(G_{enc}(I_{\Lambda}^{obs}, M; \theta_G^{enc}); \theta_G^{dec}),$$

where the inputs are the observed masked image and the binary mask, and the output is the completed image.

The generator contains two components G_{enc} and G_{dec}. G_{enc} encodes the input pair (I^{obs}, M) to a latent low dimensional vector. The latent vector is then used in G_{dec}, which decodes it into the completed image. G_{enc} and G_{dec} are mirrors of each other in terms of their network structures.

The discriminator is defined by $D(I; \theta_D)$ where the input data I is either the ground truth non-corrupted image or the completed image from the generator. The discriminator computes the binary classification between real and fake and is parametrized by θ_D.

The complete set of parameters used in the complete model is $\theta = (\theta_G, \theta_D)$.



(a) Full Face Mask (b) Random Rectangle Mask (c) Random Noise Mask

Figure 3.7: Different mask types that were used in the infilling GAN model.

There were 3 different types of masks that were used during the model training in order to test the capabilities of the model. These are shown in Figure 3.7.

The first type of mask is the Full Face Mask, where the entire face in the image is masked out. This is done by using a pre-trained Haar feature-based cascade classifier [47] to detect a bounding box for the face in the image, then masking out everything inside that bounding box.

The second type of mask is a Random Rectangle Mask, which is created by generating a rectangle of random size and location.

The third type of mask is the Random Noise Mask, which is created through the following

steps:

1. Starting with an all-zero one-channel image, choose a rectangular region of random size and location.
2. Generate a low resolution noise image (4 size) and up sample to the size of the chosen rectangle with bi-linear interpolation, creating a rectangular region with continuous random values.
3. Convert image to a binary mask with thresholding at 0.5.

3.4.3 Infilling GAN Training Process

The model is trained using a combination of the following loss functions:

1. Adversarial Loss, similar to the previous models:

$$l(I_{\Lambda}^{real}, M_{\Lambda}, I_{\Lambda}^{obs}|G, D) = (1 - \log(1 - D_{cls}(I^{syn})) + \log D_{cls}(I^{real})$$

2. Reconstruction Loss, which encourages the preservation of both target region and context region:

$$l_{rec}(I^{real}, M, I^{obs}|G) = \left\|\alpha \cdot M \odot (I^{real} - I^{syn})\right\|_1 + \left\|(1-\alpha) \cdot (1-M) \odot (I^{real} - I^{syn})\right\|_1$$

   α is the trade-off parameter between the target region and the context region and is a value in $[0, 1] \in \mathbb{R}$. The symbol $\odot$ here represents element-wise multiplication. Our experiments used an α of 0.7.

3. Feature Loss, which encourages the synthesized image to have a similar feature representation to the original image based on a pre-trained neural-network ϕ:

$$l_{feature}(I^{real}, M, I^{obs}, A^{obs}|\phi, G) = \left\|\phi_j(I^{real}) - \phi_j(I^{syn})\right\|_2^2$$

ϕ_j is the activations of the jth layer of ϕ. In our experiments, ϕ_j is the `block2_conv2` of a 16-layer VGG network [43] pre-trained on the ImageNet dataset [8].

4. Boundary Loss, a modified reconstruction loss that encourages the blending of target region with original context region:

$$l_{boundary}(I^{real}, M, I^{obs}, A^{obs}|G) = \left\| w \odot (I^{real} - I^{syn} \right\|)$$

w is a weighted kernel that is computed by blurring the mask M with a mean filter so that the pixels closer to the mask boundary in the context area have higher weights. Our experiments used a mean filter size of 7.

Combining these loss functions, minimax game becomes:

$$\min_G \max_D \mathcal{L}(G, D) = \mathcal{L}_{adv}(G, D) + \lambda_{rec} \cdot \mathcal{L}_{rec}(G) + \lambda_{feature} \cdot \mathcal{L}_{feature}(G, \phi) + \lambda_{boundary} \cdot \mathcal{L}_{boundary}(G)$$

where the λ are trade-off parameters between the different loss terms.

Progressive growing will also be used during the training process of this model. Also, due to the size of the model, in order to speed up model training, the training set was decreased from the original CelebA-HQ dataset [21] size of 30 thousand images of 1024 resolution to 10 thousand images of 64 resolution for this model only.

3.5 Infilling with Latent Variable Optimization

The second method of infilling that was explored was infilling through optimizing the latent variable encoding of the facial image. This method attempts to find a latent variable that will generate an image that is similar to the given image by taking gradient steps in the latent space rather than in the model parameter space [34].

3.5.1 Latent Variable Optimization Model Structure

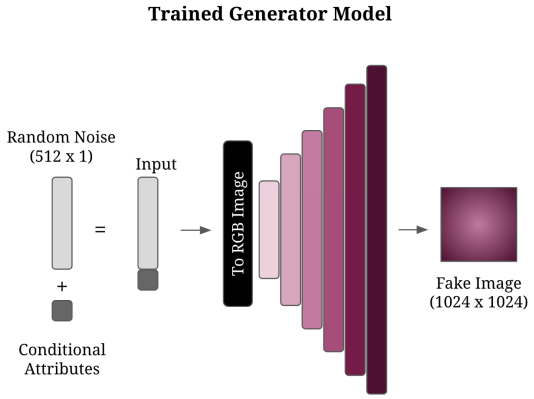


Figure 3.8: Network structure of the generator component of the conditional GAN model from Section 3.3.

This infilling technique does not require a new model architecture. Instead, we will be using the generator from the pre-trained conditional GAN model from Section 3.3, whose structure is shown in Figure 3.8.

3.5.2 Latent Variable Optimization Process

Given a desired image x, the conditional attribute associated with it y, and an initial latent variable value z_0, we want to optimize the latent variable so that it produces an image that resembles the desired image x.

For the given latent vector z_0, we produce the generated image $G(z_0, \theta_G)$. In order to recover the input z that most closely produces the desired image x, we successively update

z_i using gradient descent over L_2 loss [34] as follows:

$$z_{i+1} = z_i - \eta \nabla_{z_i} ||x - G(z_i, y, \theta_G)||_2^2$$

where η represents the learning rate.

When z_{i+1} is given as an input to the generator, will produce an output image that is more similar to the original given image than z_i.

Given a binary mask M_Λ on the image with the target and context regions as defined in Section 3.4.2, the loss function becomes:

$$z_{i+1} = z_i - \eta \nabla_{z_i} ||(1 - M) \odot (x - G(z_i, \theta_G))||_2^2$$

where the symbol $\odot$ here represents element-wise multiplication and 1 is a matrix of ones of the same size as the image.

This encourages the model to find an image that is similar to the given image only in the context region, removing the masked areas from consideration. We rely on the ability of the original model to generate faces in order to assure that the global structure of the output image, including the masked areas, remains realistic. Therefore, the results of this infilling technique depend heavily on the fidelity of the trained conditional model.

Empirically, it was found that optimizing over 5000 iterations with an initial learning rate of 0.000001 that is decreased by a factor of 1.1 every 100 iterations starting from iteration 3000 produced good results. The minimum learning rate is capped at 0.00000005.

3.6 Extended Infilling Models

Both the encoder infilling model and the latent optimization method can be extended to infill the facial portion of an image of arbitrary size with auxiliary information in the image.

The extended infilling models work as follows:

1. Use the pre-trained Haar feature-based cascade classifier [47] to detect the facial portion of the image. This model will return the size and location of a bounding box for the face.

2. Create a larger bounding box for the detected face that is square, centered on the original bounding box, and 1.5 times the size of the original box. This creates a crop of the face that more closely matches the appearance of the facial images in the CelebA-HQ [21] training set.

3. Crop the bounded area of the image and resize it to the expected input size of the model, 1024×1024 pixels.

4. Feed the cropped and resized image into the infilling model, which can be either the encoder infilling model or the latent optimization method.

5. Take the output of the infilling model, which will be of size 1024×1024 pixels, and resize it to be the size of the original cropped location.

6. Replace the cropped area of the image with the resized output of the model.

The resulting image will have only the facial image portion of the image modified, with the remainder of the image staying the same.

Since these extended infilling models use the original infilling models during their image generation process, the results of the extended models depend heavily on the fidelity of the original infilling models.

Chapter 4

Results

This section outlines the results of the models described in the previous chapter.

Additional images of the results, including interpolation videos and videos of the images over the training process of the models can be found online at:

`https://drive.google.com/drive/folders/1sez56gIqWx5zUkOHXnPbjzMZNq01dy3I?usp=sharing`

4.1 Base GAN Model Results

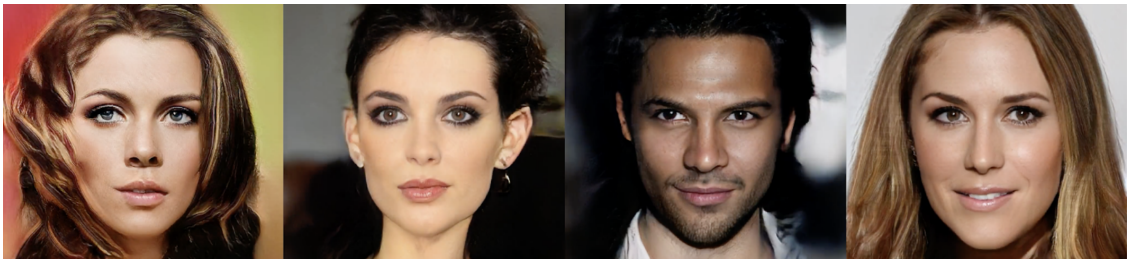


Figure 4.1: Samples of images produced by a base GAN model trained with growing.

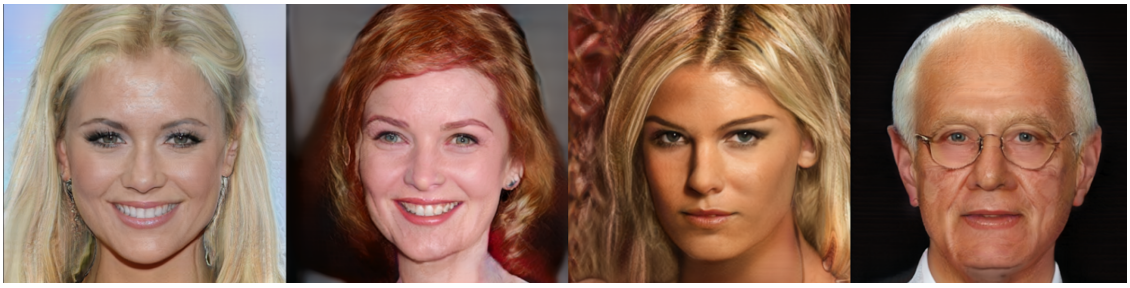


Figure 4.2: Samples of images produced by a base GAN model trained without growing, before the model had mode collapsed.

Figure 4.3: Samples of images produced by a base GAN model trained without growing, after the model had mode collapsed.

Figure 4.1 shows a sample of the images generated by the base GAN model outlined in Section 3.2, trained with progressive growing. When compared to the results of the model trained without growing shown in Figure 4.2, the quality of the results look similar.

However, the advantage of the progressively growing model is that the training process is more stable. When compared to the results of the same non-growing model after the model had mode collapsed (shown in Figure 4.3), we can see that the results for the growing model are significantly better.

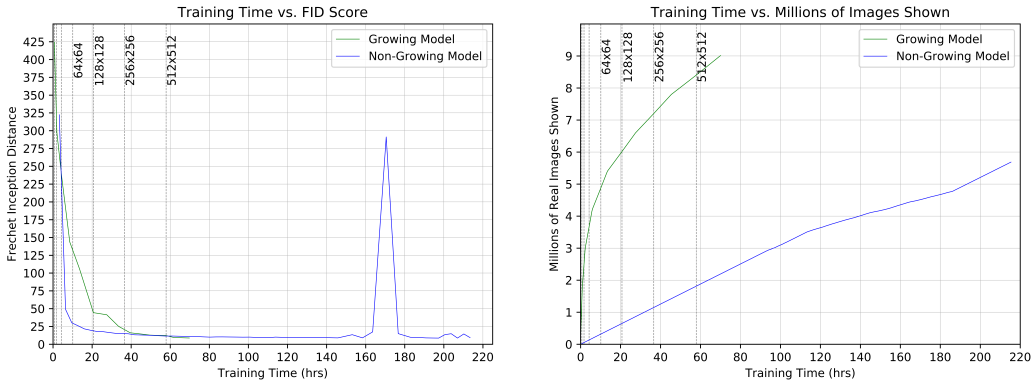


Figure 4.4: **Left:** Training time vs. Average Frechet Inception Distance [18] over 50 thousand images for the growing and non-growing versions of the base GAN model. We can see that the mode collapse in the growing model happens around 170 hours into the training, corresponding with the spike in the Frechet Inception Distance (FID) metric. **Right:** Training time vs. Number of images shown for the growing and non-growing versions of the base GAN model.
The dashed lines in the diagrams above show points where the resolution of the growing model increases.

Figure 4.4 shows the Frechet Inception Distance (FID) score [18] throughout the training process for both the growing and non-growing models, as well as the number of images shown to each model. The growing version of the model is able to achieve a slightly better FID than the non-growing model, in a similar amount of training time. We can also see the mode collapse of the non-growing model occurring in the model around 170 hours into the training time. As well, since we are training at lower resolutions in the growing model, we are able to show the growing model many more images than the non-growing model. This potentially contributes to the improved results of the growing model.

4.1.1 Limitations of the Base GAN Model

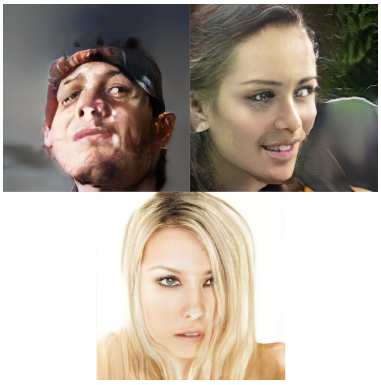
Although the performance of the base GAN model was satisfactory, the model faces limitations when generating facial images with accessories such as hats, sunglasses or collars, or faces at different viewing angles. Examples of these are shown in Figure 4.5. This is due the to the low percentages of images with accessories or at different viewing angles in the CelebA-HQ dataset, as displayed in Appendix A Table A.1, so the model is not shown an adequate amount of examples for it to reproduce them faithfully.

The model also sometimes generates mismatched facial features, such as eyes of different

colours within the same face. An example of this is shown in Figure 4.5. This could potentially be mitigated using a loss function to penalize discrepancies between facial features in the same face.

An exploration of these limitations has been left for future work.

(a) Distorted generated accessories.



(b) Distorted angled faces.

(c) Inconsistent features.

Figure 4.5: Failure cases of the basic GAN model.

4.2 Conditional GAN Model Results

Figure 4.6: Images generated by the conditional GAN model trained **with growing**. The top row is generated from the FEMALE conditional label, and the bottom row is generated from the MALE conditional label. Each of the columns is generated from the same latent vector, with only the conditional label changed.

Figure 4.7: Images generated by the conditional GAN model trained **without growing**. The top row is generated from the female conditional label, and the bottom row is generated from the male conditional label. Each image is generated using a different latent vector.

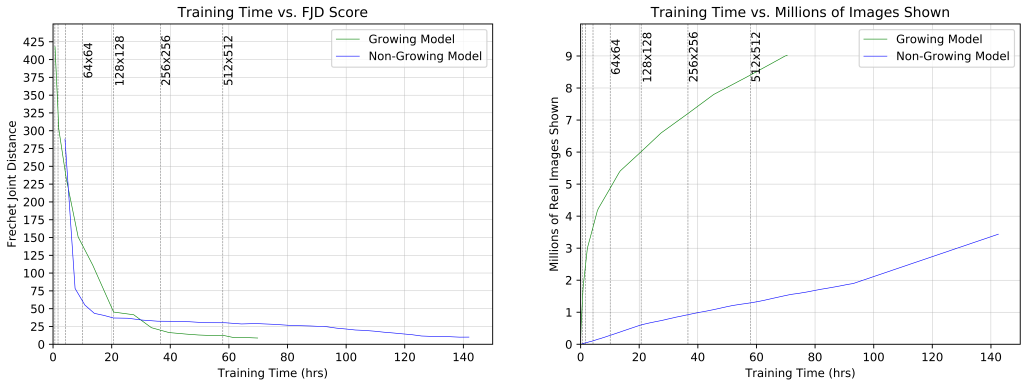


Figure 4.8: **Left:** Training time vs. Average Frechet Joint Distance [10] over 50 thousand images for the growing and non-growing versions of the base GAN model. **Right:** Training time vs. Number of images shown for the growing and non-growing versions of the base GAN model.
The dashed lines in the diagrams above show points where the resolution of the growing model increases.
Due to the conditioning of the model, we have used Frechet Joint Distance [10] rather than the Frechet Inception Distance [18] as the evaluation metric for the model.

Figure 4.6 shows the results produced by the conditional model outlined in Section 3.3. The quality of the images generated with the conditional model is similar to the non-conditional model, and the model is able to successfully learn to generate images that match the desired label.

Comparing these results to the results produced by the non-growing model shown in Figure 4.7, we can see that although the non-growing model is still able to produce realistic facial images, it is unable to learn to distinguish the differences between the different conditional label values.

These qualitative results are reflected in the Frechet Joint Distance (FJD) [10] metrics of both models shown in Figure 4.8. The growing model is able to achieve a better result in shorter training time than the non-growing model. Although the FJD score of the non-growing model still appears low due to the model's ability to nevertheless generate realistic facial images, and due to the many shared attributes between male and female faces, the inability of the model to learn the conditional label is reflected in the larger gap between the scores of the growing and non-growing model when compared to the FID scores from the non-conditional model.

4.2.1 Interpolating and Extrapolating the Conditional Label

Using the conditional model trained on the binary MALE label, we can also interpolate and extrapolate within the latent space of the label in order to control the gender expression within the face.

The interpolation of the MALE label is shown in Figure 4.9. When we interpolate the values of the label, we can see that there are certain features such as short hair, facial hair, and less makeup that the model associates more with male faces over female faces.

When we extrapolate the values of the label, we can also visualize what features the model associates with extremely female faces and extremely male faces. This extrapolation is shown in Figure 4.10. All of the faces on the extremities of either side of the spectrum are quite homogeneous. All of the female faces are blonde and lighter skinned with extremely heavy makeup, whereas all of the male faces are slightly tan with brown hair and heavy facial hair. These biases are due to the skewed data distributions in the CelebA-HQ training set that are learned by the model. There is a greater portion of the females with blonde hair and heavy makeup in the dataset than males, and there is a greater portion of males with facial hair than females. A breakdown of the distributions skews of these features are included in Appendix A.

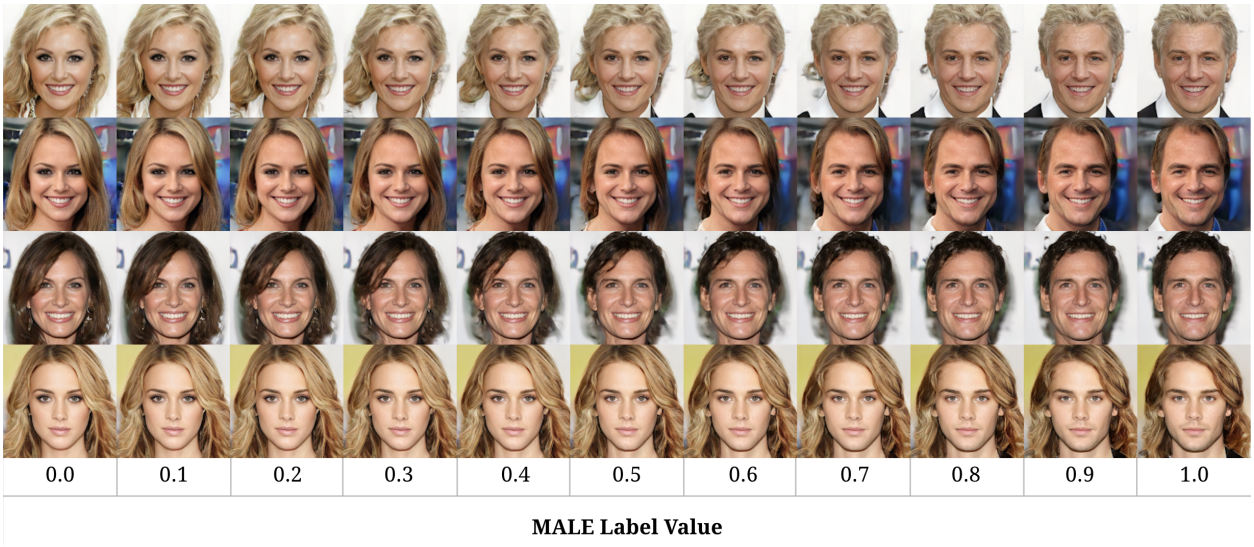


Figure 4.9: Images generated by interpolating the binary MALE label in the conditional GAN model.

Figure 4.10: Images generated by extrapolating the binary MALE label in the conditional GAN model.

4.2.2 Limitations of the Conditional GAN Model

The conditional version of the model also encounters similar failure cases to the base GAN model when attempting to generate images with accessories and faces at different viewing angles. In addition to those failure cases, the conditional model encounters other failure cases resulting from the use of the conditional label.

Diversity of Extrapolated Generated Faces

Due to the the biased distributions in the training dataset, the ethnic diversity of the faces produced by the model is low. Although it is possible for the model to produce more varied faces (as shown in Figure 4.11), the majority of values in the latent space map to faces more similar to the ones shown in Figure 4.6.

When we interpolate the conditional variable for the faces in Figure 4.11, we can see that the model is able to maintain key features of the individual such as skin colour and hair colour for the intermediate gender expression values in the face. These interpolations are shown in Figure 4.12. However, for extrapolated values of the gender label, as shown in Figure 4.13, the model does not maintain those attributes in the individual and resorts to generating faces that are very similar to the ones shown on the extremities in Figure 4.10. This is also due to the data distribution in the training dataset, and could potentially be mitigated by adding additional examples to the dataset.

Figure 4.11: Images generated by the conditional GAN model trained **with growing**. The top row is generated from the FEMALE conditional label, and the bottom row is generated from the MALE conditional label. Each of the columns is generated with the same latent vector, with only the conditional label changed. This set of results attempts to showcase a more diverse range of faces that the model is able to generate in terms of attributes such as skin tones, facial features, and hair colour.

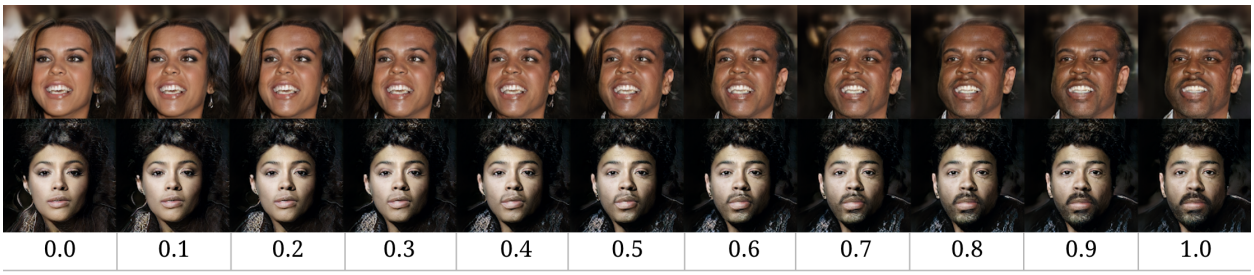


Figure 4.12: Images generated by interpolating the binary MALE label in the conditional GAN model from the images in Figure 4.11.

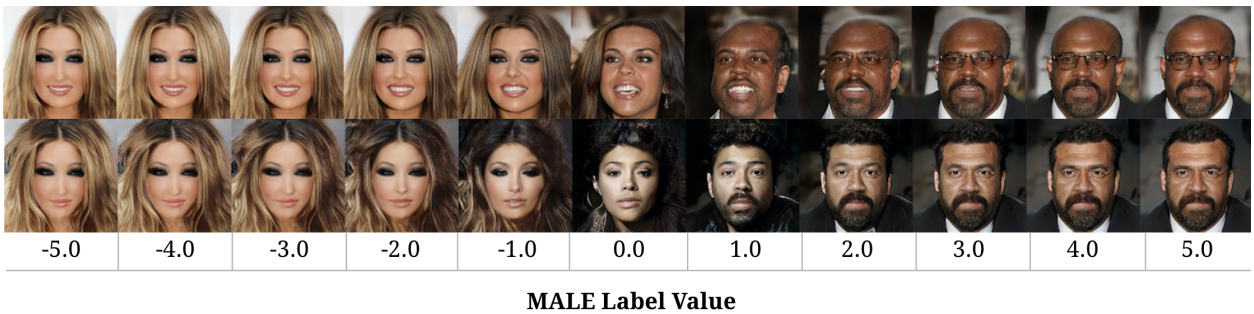


Figure 4.13: Images generated by extrapolating the binary MALE label in the conditional GAN model from the images in Figure 4.11.

Size of Conditional Label

The current conditional model is limited to faces with a singular label. When this model is extended to labels with multiple attributes, the model fails to generate images that match the desired corresponding label. This failure case is shown for a label size of 2 in Figure 4.14 and a label size of 3 in Figure 4.15. As shown in Figure 4.14, this limitation still occurs throughout a range of different conditional loss functions.

This could potentially be overcome by training a separate classifier model in order to more accurately predict the label, or by using a different labelling technique, such as one-hot labelling with 2^n categories for n binary labels. Extending this model to multiple labels has been left for future work.

(a) Sigmoid cross-entropy over the entire attribute label.

(b) Sigmoid cross-entropy over each individual label.

(c) Softmax cross-entropy over the entire attribute label.

Figure 4.14: Samples should the failure of the model to generate faces that match the desired label when the conditional label is extended to 2 attributes. The failure occurs with a variety of different loss functions.

Figure 4.15: Samples should the failure of the model to generate faces that match the desired label when the conditional label is extended to 3 attributes. This figure showcases a model trained with softmax loss as the conditional loss function.

Inconsistencies between Faces from the Same Latent Vector

Finally, the conditional version of the model also has a rare failure case where the model produces male and female versions of a face from the same latent vector that are inconsistent with each other. An example is shown in Figure 4.16. This is also mostly likely caused by the limitations of the dataset, where one gender's photos are more ethically diverse than the other.

Figure 4.16: Failure case of the conditional model, where the male and female variations of faces generated from the same latent vector do not resemble the same person.

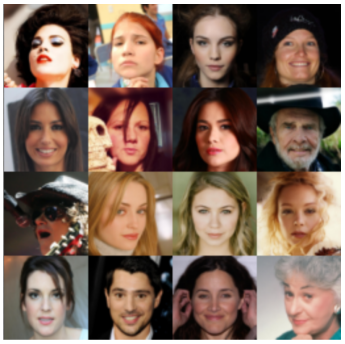
4.3 Infilling Encoder GAN Model Results

This section presents the results of training various versions of the encoder GAN model outlined in Section 3.4. The results of the model have thus far been unsatisfactory, but have been presented here nonetheless for informative purposes.

A sample of the original images that the model is attempting to produce is shown in Figure 4.17a. The full-face masked version of these images is shown in Figure 4.17b.

Initial experiments with the infilling model attempted to reproduce the unmasked images directly, by feeding in the original unmasked images into the model. Figure 4.18a displays the results produced by the model without skip connections. As shown, the model is able to match some of the details in the backgrounds for each of original images, but it appears that the model has mode collapsed for the facial region, as the faces that it generates for all the images are quite homogeneous and do not match the context of each of the images. As well, the images produced are rather blurry, and are not of the same quality as the original images.

When the skip connections are added, the model is able to reproduce the original image much more faithfully, as shown in Figure 4.18b. Although the results presented are from a model that was not converged due to limitations with computation time, we can still see that the general results are of much better quality than those generated by the model without skip connections.



(a) A sample of the original images from the CelebA-HQ dataset that our model is attempting to reproduce.

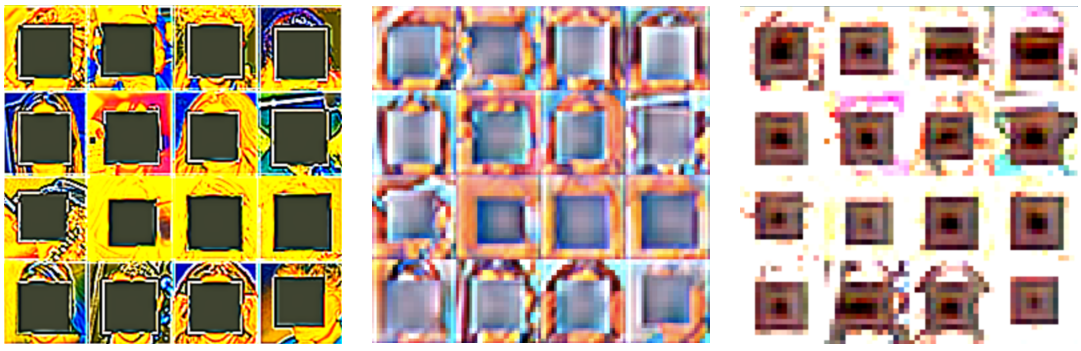
(b) A masked version of the images using Full Face Masks.

Figure 4.17: Unmasked and masked versions of the images used during training.

(a) Results of reproducing the original image **without** skip connections in the model.

(b) Results of reproducing the original image **with** skip connections in the model.

Figure 4.18: Results of attempting to reproducing the original image directly.



(a) (b) (c)

Figure 4.19: The results for the 3 models specified in Table 4.1

Model Parameter	(a)	(b)	(c)
Reconstruction α	0.5	0.7	0.7
Masked Images Only in Generator	False	True	False
Reconstruction Loss	True	True	True
Feature Loss	True	True	False
Boundary Loss	True	True	False
Growing	True	True	True
Skip Connections	True	True	True

Table 4.1: Modified Model Parameters in the models for the results shown in Figure 4.19

The next set of experiments attempted to generate infilled images from the masked versions of the images. Despite attempting several versions of the model, these experiments were unable to produce infilled versions of the images. The model consistently emphasizes the masked area of the image and did not infill it. A subset of the outputs of these models are shown in Figure 4.19, and any modifications to the original model listed in Section 3.4 for each of these models is listed in Table 4.1. This emphasis on the masked area also occurs with the other mask shapes, as shown in Figure 4.20.

(a) (b)

Figure 4.20: Results from applying the infilling model on a variation of the dataset with random noise masks. The model that produced these results did not contain any modifications from the original model listed in Section 3.4.

The encoder GAN infilling model faces many training challenges, primarily due to GPU memory constraints caused by the additional parameters added to the model through the encoder and skip connections. These constraints force the use of much smaller mini-batches, which significantly increases training time, and could potentially degrade the results [21].

An exploration of the cause of these unsatisfactory results and the emphasis on the masked area is left for future work.

4.4 Latent Variable Optimization Results

Figure 4.21: Results of using the optimized latent to reproduce the original image. The original images are on the left and the reproduced images from the optimized latents are on the right. **First Row:** Results for an image that was originally generated by the model. **Second Row:** Results for an image that was in the training set for the model. **Third row:** Results for an image that was not within the training set.

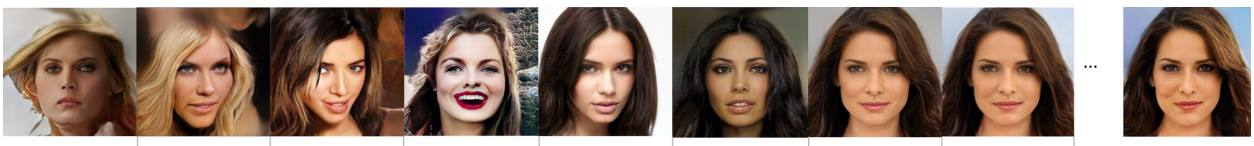
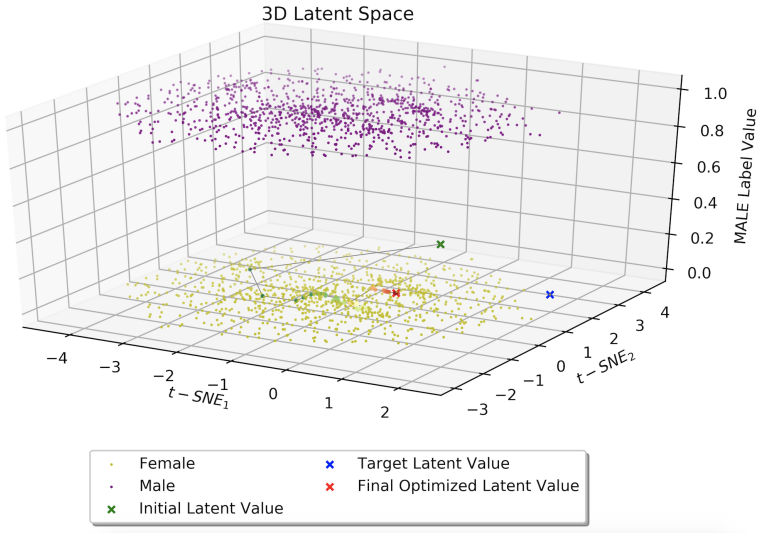
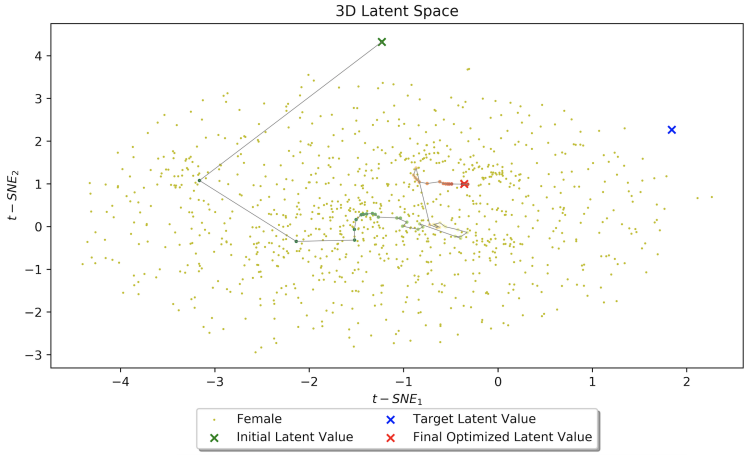


Figure 4.22: The images produced from the latent variable at various optimization iterations.



(a) 3D Latent Space Projection, with MALE label on the z axes.



(b) 2D Latent Space Projection, showing only the **FEMALE** latent point cloud.

Figure 4.23: The gradient descent path through latent space when optimizing the latent variable. The 513×1 size latent space (512×1 random noise with an additional dimension for the conditional label) has been projected using t-SNE [46] onto 3D and 2D here for visualization purposes.

Figure 4.21 shows the results of the generated images from optimizing the latent variable. When given an image that was originally produced by the model, the latent optimization is able to reproduce the original image almost exactly. This can be contributed to the fact that the given image is guaranteed to exist within the generator distribution. Figure 4.22 displays the iterations of the latent variable for the first row of results in Figure 4.21, and Figure 4.23 displays the path of the gradient descent of this optimization through the latent

space.

When given an image that was not generated by the model but was in the training data, we can see that the latent optimization is unable to match the image exactly, but is able to generate an image that is very similar. Many of the key features of the face are matched, such as the eye colour, hair colour, general facial structure, and skin tone.

When given an image that was not generated by the model and not in the training set, we can see that the model is still able to produce a face that is visually similar.

When comparing the reconstruction loss of these 3 cases as shown in Figure 4.24, we can see that the image originally generated by the model ends up with an optimized latent that has the lowest L_2 loss and the image not from the training set ends up with the highest loss, which is consistent with our qualitative image results.

Figure 4.24: The L_2 value over the training iterations for the latent vector optimization. This graph demonstrates the results of the optimization for up to 10 thousand iterations. However, during evaluation it was found that the images generated after 5000 iterations and beyond were extremely similar and did not provide much benefit over the additional computation time required.

Figure 4.25 shows the results of applying this latent optimization algorithm on different types of masked images. As shown, the infilled results from the masked faces are still able to maintain a similar context region to the original image and produce a face, although the faces produced from the masked images are slightly more distorted than the one produced from the non-masked image.

Figure 4.25: Results of using latent optimization for infilling with various masks **First row:** Original image, along with 3 different mask types that were used (full face mask, random rectangle mask, and random noise mask). **Second Row:** The resulting image from the optimized latent vector for the masked image in the row above.

4.5 Extended Infilling Model Results

4.5.1 Extended Infilling Model Using Infilling GAN

Although the infilling model did not produce satisfactory results, we were still able to visualize its application within the extended infilling model, the results of which are shown in Figure 4.26. As shown, the extended model is able to detect the face in the photo, feed a resized version of that area of the photo into the infilling model, and replace the area with the output of the infilling model. As expected, due to the limitations of the infilling model, the appearance of the replaced area does not match that of the rest of the image.

(a) Original Image. (b) Full Face Mask. (c) Random Rectangle Mask. (d) Random Noise Mask.

Figure 4.26: The extended infilling model applied to a photo with additional auxiliary information. This figure shows the original photo of the left, followed by the results of applying the extended infilling model with 3 different mask types.

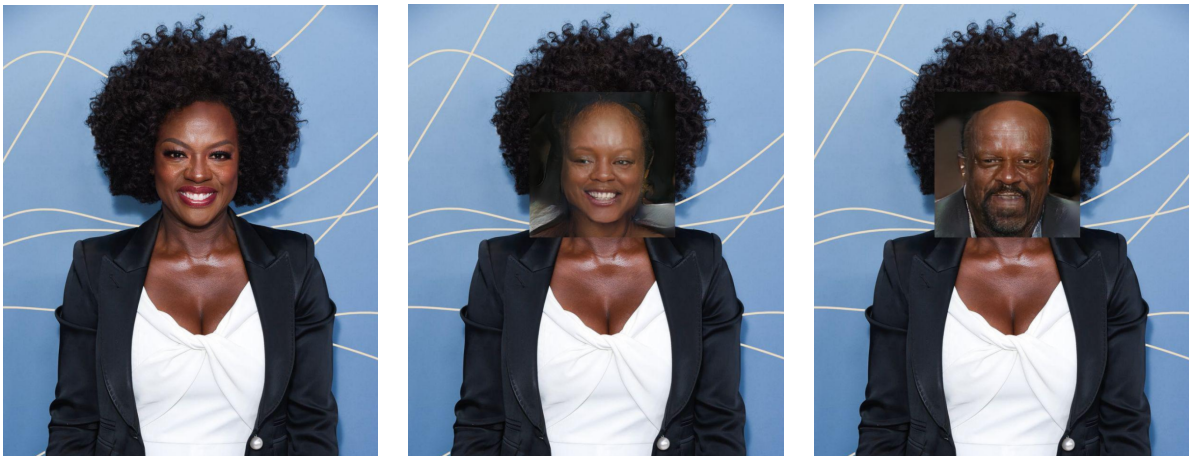
4.5.2 Extended Infilling Model Using Latent Optimization

A sample result of the extended infilling model using the latent optimization is shown in Figure 4.27. This technique does a much better job of infilling the image compared to the previous model, especially for the case where the conditional label of the generated region matches the conditional label of the original image.

However, like the conditional model that the latent optimization technique is based on, this model has a bias towards generating paler skin tones. Therefore, this model encounters failure cases where it infills images with generated faces that do not match the ethnicity of the face in the original image. As shown in Figure 4.28, this can cause mismatches between the skin tone of the face and other areas of the body, such as the hands.

As well, since the distribution learned by the discriminator is extremely non-convex, the result of the optimization process is highly dependent on the initial starting value of the latent that is given. Figure 4.29 shows the results of infilling the same image, starting from two different latent vectors, where the resulting images from the first latent vector are much better than the results from the second latent vector.

Finally, as shown in the examples, the background regions of the generated portions of the images are blurry and distorted compared to that of the original image. This could potentially be resolved by training a new model that has a closer crop of the face. We could then use this model to infill a more closely cropped area of the face, excluding the need for the model to generate the background region. An exploration of this extension has been left for future work.



(a) Original Image. (b) Infilled image using the FEMALE label. (c) Infilled image using the MALE label.

Figure 4.27: The a sample result of the extended infilling algorithm using latent variable optimization. This result was produced with a full face mask on the face.

(a) Original Image. (b) Infilled image using the FEMALE label. (c) Infilled image using the MALE label.

Figure 4.28: A failure case where the model produces a face that does not match the original ethnicity of the person in the image. This result was produced without any masking on the face.

(a) Original Image.

(b) Infilled version 1. (c) Infilled version 2.

Figure 4.29: The results of the latent optimization algorithm, with different initial starting latents.

Chapter 5

Conclusion

This thesis has covered the results of using generative adversarial networks to synthesize fake facial images while also controlling for key attributes (in this case, gender expression). We have explored four different models, including a base model, a conditional version, and two infilling techniques. Finally, we extended both of the infilling techniques to detect and replace the facial portion of an arbitrary image.

All of the models except for the encoder infilling model were shown to be able to produce satisfactory results. We were able to use the conditional model to tune the gender expression of a generated face within the latent space of the learned distribution of the generator. Although the infilling technique using an encoder component was unsuccessful, the optimized latent technique was able to successfully infill masked facial images.

Future work related to this project will be focused on improving the extended infilling models to ultimately provide a full solution to generating synthesized faces on arbitrary images. Specifically, the improved extended infilling model will then be used the replace the faces on the models in the DeepFashion dataset [27] and generate a new dataset with multiple variations of each image tuned along the latent space of gender expression in the face. These images could then be used to study the effect of gender expression in the face on dress code enforcement.

There is also the potential to extend the conditional model to be applied to multiple conditional labels. Other areas of exploration related to the conditional model include the

level of attractiveness using the labels from CelebA-HQ [21]. This model could be used to investigate the impact of perceived attractiveness on various decisions. As well, the results could be used to discuss questions surrounding the definition of beauty ingrained into facial adjustment algorithms, such as beauty filters on phone cameras, and the impact that has in different racial groups. For example, will the algorithm believe that changing a person's skin to be lighter makes them more attractive? Do aspects such as their eye colour, hair colour/texture, etc. affect this evaluation of beauty?

The models presented in this thesis allow us to achieve our original goal of producing artificial facial images for research studies, and serve as the groundwork that will allow us to explore these expansions related to other aspects of the face and body.

Bibliography

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 2009. URL `https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/patchmatch.pdf`.

[2] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000. URL `https://www.researchgate.net/profile/C_Ballester/publication/220720382_Image_inpainting/links/02e7e52c7c949ba3af000000/Image-inpainting.pdf`.

[3] Marcelo Bertalmio, Luminita Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processings*, 2003. URL `https://www.math.ucla.edu/~lvese/PAPERS/01217265.pdf`.

[4] Carol Braverman. *Photoshop Retouching Handbook*. Wiley, 1998.

[5] Zeyuan Chen, Shaoliang Nie, Tianfu Wu, and Christopher G. Healey. High resolution face completion with multiple controllable attributes via fully end-to-end progressive generative adversarial networks. *arXiv*, 2018. URL `https://arxiv.org/pdf/1801.07632.pdf`.

[6] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. *Computer Vision and Pattern Recognition*, 2003. URL `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/criminisi_cvpr2003.pdf`.

[7] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B. Goldman, and Pradeep Sen. Image melding: Combining inconsistent images using patch-based synthesis. *UCSB*, 2009. URL https://www.ece.ucsb.edu/~psen/Papers/SIGGRAPH12_ImageMelding.pdf.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *arXiv*, 2015. URL https://arxiv.org/pdf/1506.05751.pdf.

[10] Terrance DeVries, Adriana Romero, Luis Pineda, Graham W. Taylor, and Michal Drozdzal. On the evaluation of conditional gans. *arXiv*, 2019. URL https://arxiv.org/pdf/1907.08175.pdf.

[11] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. *IEEE International Conference on Computer Vision*, 1999. URL https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/papers/efros-iccv99.pdf.

[12] Reza Fazeli. Generating high-resolution images using deep autoregressive models. *Towards Data Science*, 2019. URL https://towardsdatascience.com/generating-high-resolution-images-using-autoregressive-models-3683f9af0db4.

[13] Kevin Frans. Variational autoencoders explained, 2016. URL http://kvfrans.com/variational-autoencoders-explained/.

[14] Stuart Geman, Daniel F. Potter, and Zhiyi Chi. Composition systems. *Quarterly of Applied Mathematics*, 2002. URL https://www.researchgate.net/profile/Stuart_Geman/publication/267466711_Composition_systems/links/5c48d010a6fdccd6b5c429f5/Composition-systems.pdf.

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv*, 2014. URL https://arxiv.org/pdf/1311.2524.pdf.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *arXiv*, 2014. URL https://arxiv.org/pdf/1406.2661.pdf.

[17] James Hays and Alexei A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics*, 2007. URL http://graphics.cs.cmu.edu/projects/scene-completion/scene-completion.pdf.

[18] Martin Heusel, Thomas Unterthiner Hubert Ramsauer, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv*, 2017. URL https://arxiv.org/abs/1706.08500.

[19] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics*, 2017. URL http://iizuka.cs.tsukuba.ac.jp/projects/completion/data/completion_sig2017.pdf.

[20] Andrej Karpathy. Neural networks part 1: Setting up the architecture. *CS231n Convolutional Neural Networks for Visual Recognition*, 2019. URL http://cs231n.github.io/neural-networks-1/.

[21] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv*, 2018. URL https://arxiv.org/pdf/1710.10196.pdf.

[22] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv*, 2014. URL https://arxiv.org/pdf/1312.6114.pdf.

[23] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *arXiv*, 2017. URL https://arxiv.org/pdf/1606.04934.pdf.

[24] Nikos Komodakis and Georgios Tziritas. Image completion using global optimization. *IEEE CVPR*, 2006. URL https://www.csd.uoc.gr/~tziritas/papers/completion_cvpr2006.pdf.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPs*, 2012. URL https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[26] Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 2003. URL https://www.cc.gatech.edu/~turk/my_papers/graph_cuts.pdf.

[27] Victor Kwon. Deepfashion dataset, 2019.

[28] Yann LeCun. A theoretical framework for backpropagation, 1988. URL http://yann.lecun.com/exdb/publis/pdf/lecun-88.pdf.

[29] Yann LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code digit recognition, 1989. URL http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf.

[30] Yann LeCun, Patrick Haffner, Leon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning, 1999. URL http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf.

[31] Minhyeok Lee and Junhee Seok. Controllable generative adversarial network. *arXiv*, 2019. URL https://arxiv.org/pdf/1708.00598.pdf.

[32] Anat Levin, Assaf Zomet, and Yair Weiss. Learning how to inpaint from global image statistics. *CSAIL*, 2003. URL http://people.csail.mit.edu/alevin/papers/inpainting-iccv03.pdf.

[33] Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang. Generative face completion. *arXiv*, 2017. URL https://arxiv.org/pdf/1704.05838.pdf.

[34] Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks, 2017. URL https://vision.cornell.edu/se3/wp-content/uploads/2017/03/recovering-latent-vectors.pdf.

[35] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrellk. Fully convolutional networks for semantic segmentation, 2015. URL `https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf`.

[37] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv*, 2014. URL `https://arxiv.org/pdf/1411.1784.pdf`.

[38] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv*, 2017. URL `https://arxiv.org/pdf/1610.09585.pdf`.

[39] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL `https://arxiv.org/pdf/1604.07379.pdf`.

[40] Alec Radford, Luke Metz, and Soumith Chintala. Unnsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 2016. URL `https://arxiv.org/pdf/1511.06434.pdf`.

[41] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018. URL `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`.

[42] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv*, 2016. URL `https://arxiv.org/pdf/1606.03498.pdf`.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. URL `https://arxiv.org/abs/1409.1556`.

[44] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv*, 2016. URL `https://arxiv.org/pdf/1601.06759.pdf`.

[45] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv*, 2016. URL `https://arxiv.org/pdf/1606.05328.pdf`.

[46] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008. URL `http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf`.

[47] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Conference on Computer Vision and Pattern Recognition*, 2001. URL `https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf`.

[48] Marta Wilczkowiak, Gabriel J Brostow, Ben Tordoff, and Roberto Cipolla. Hole filling through photomontage. *BMVC*, 2005. URL `http://mi.eng.cam.ac.uk/research/projects/HoleFilling/HoleFilling_BMVC05_Wilczkowiak_etal.pdf`.

[49] Sitao Xiang and Hao Li. On the effects of batch and weight normalization in generative adversarial networks. *arXiv*, 2017. URL `https://arxiv.org/pdf/1704.03971.pdf`.

[50] Raymond A. Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson, and Minh N. Do. Context encoders: Feature learning by inpainting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. URL `https://arxiv.org/pdf/1607.07539.pdf`.

[51] Sameh Zarif, Ibrahima Faye, and Dayang Awang Rambli. Image completion: Survey and comparative study. *International Journal of Pattern Recognition and Artificial Intelligence*, 29:1554001, 12 2014. doi: 10.1142/S0218001415540014.

Appendix A

CelebA-HQ Dataset Statistics

The CelebA-HQ dataset [21] is generated from the CelebA dataset[35]. This high-quality facial image dataset consists of 30000 images at 1024×1024 resolution. The steps to generate the CelebA-HQ dataset are outlined by Karras et al.

Table A.1 shows the distributions of a subset of the CelebA-HQ labels between the MALE and FEMALE images.

Attribute	P(MALE|Attribute)	P(FEMALE|Attribute)	P(Attribute|MALE)	P(Attribute|FEMALE)
5 o'Clock Shadow	4490/4491	1/4491	4490/11057	1/18943
4491 Images (14.97%)	(99.98%)	(0.02%)	(40.61%)	(0.01%)
Arched Eyebrows	833/11020	10187/11020	833/11057	10187/18943
11020 Images (36.73%)	(7.56%)	(92.44%)	(7.53%)	(53.78%)
Attractive	3437/17218	13781/17218	3437/11057	13781/18943
17218 Images (57.39%)	(19.96%)	(80.04%)	(31.08%)	(72.75%)
Bags Under Eyes	5778/8634	2856/8634	5778/11057	2856/18943
8634 Images (28.78%)	(66.92%)	(33.08%)	(52.26%)	(15.08%)
Bald	711/712	1/712	711/11057	1/18943
712 Images (2.37%)	(99.86%)	(0.14%)	(6.43%)	(0.01%)
Bangs	1127/5425	4298/5425	1127/11057	4298/18943
5425 Images (18.08%)	(20.77%)	(79.23%)	(10.19%)	(22.69%)
Big Lips	2712/10890	8178/10890	2712/11057	8178/18943
10890 Images (36.30%)	(24.90%)	(75.10%)	(24.53%)	(43.17%)
Big Nose	6659/9734	3075/9734	6659/11057	3075/18943
9734 Images (32.45%)	(68.41%)	(31.59%)	(60.22%)	(16.23%)
Black Hair	3194/6592	3398/6592	3194/11057	3398/18943
6592 Images (21.97%)	(48.45%)	(51.55%)	(28.89%)	(17.94%)
Blond Hair	215/5126	4911/5126	215/11057	4911/18943
5126 Images (17.09%)	(4.19%)	(95.81%)	(1.94%)	(25.93%)
Blurry	61/113	52/113	61/11057	52/18943
113 Images (0.38%)	(53.98%)	(46.02%)	(0.55%)	(0.27%)
Brown Hair	1941/6925	4984/6925	1941/11057	4984/18943
6925 Images (23.08%)	(28.03%)	(71.97%)	(17.55%)	(26.31%)
Bushy Eyebrows	3702/5676	1974/5676	3702/11057	1974/18943
5676 Images (18.92%)	(65.22%)	(34.78%)	(33.48%)	(10.42%)
Chubby	1825/2102	277/2102	1825/11057	277/18943
2102 Images (7.01%)	(86.82%)	(13.18%)	(16.51%)	(1.46%)
Double Chin	1501/1786	285/1786	1501/11057	285/18943
1786 Images (5.95%)	(84.04%)	(15.96%)	(13.58%)	(1.50%)
Eyeglasses	1226/1468	242/1468	1226/11057	242/18943
1468 Images (4.89%)	(83.51%)	(16.49%)	(11.09%)	(1.28%)
Goatee	2287/2290	3/2290	2287/11057	3/18943
2290 Images (7.63%)	(99.87%)	(0.13%)	(20.68%)	(0.02%)
Gray Hair	1014/1242	228/1242	1014/11057	228/18943
1242 Images (4.14%)	(81.64%)	(18.36%)	(9.17%)	(1.20%)
Heavy Makeup	17/13708	13691/13708	17/11057	13691/18943
13708 Images (45.69%)	(0.12%)	(99.88%)	(0.15%)	(72.27%)
High Cheekbones	3270/13847	10577/13847	3270/11057	10577/18943
13847 Images (46.16%)	(23.62%)	(76.38%)	(29.57%)	(55.84%)
Male	11057/11057	0/11057	11057/11057	0/18943
11057 Images (36.86%)	(100.00%)	(0.00%)	(100.00%)	(0.00%)
Mouth Slightly Open	4393/14139	9746/14139	4393/11057	9746/18943
14139 Images (47.13%)	(31.07%)	(68.93%)	(39.73%)	(51.45%)
Mustache	1735/1735	0/1735	1735/11057	0/18943
1735 Images (5.78%)	(100.00%)	(0.00%)	(15.69%)	(0.00%)
Narrow Eyes	1500/3516	2016/3516	1500/11057	2016/18943
3516 Images (11.72%)	(42.66%)	(57.34%)	(13.57%)	(10.64%)
No Beard	5395/24328	18933/24328	5395/11057	18933/18943
24328 Images (81.09%)	(22.18%)	(77.82%)	(48.79%)	(99.95%)
Oval Face	1804/6243	4439/6243	1804/11057	4439/18943
6243 Images (20.81%)	(28.90%)	(71.10%)	(16.32%)	(23.43%)
Pale Skin	300/1533	1233/1533	300/11057	1233/18943
1533 Images (5.11%)	(19.57%)	(80.43%)	(2.71%)	(6.51%)
Pointy Nose	2201/9506	7305/9506	2201/11057	7305/18943
9506 Images (31.69%)	(23.15%)	(76.85%)	(19.91%)	(38.56%)
Receding Hairline	1519/2530	1011/2530	1519/11057	1011/18943
2530 Images (8.43%)	(60.04%)	(39.96%)	(13.74%)	(5.34%)
Rosy Cheeks	64/3379	3315/3379	64/11057	3315/18943
3379 Images (11.26%)	(1.89%)	(98.11%)	(0.58%)	(17.50%)
Sideburns	2428/2430	2/2430	2428/11057	2/18943
2430 Images (8.10%)	(99.92%)	(0.08%)	(21.96%)	(0.01%)

Attribute	P(MALE|Attribute)	P(FEMALE|Attribute)	P(Attribute|MALE)	P(Attribute|FEMALE)
Smiling	4363/14092	9729/14092	4363/11057	9729/18943
14092 Images (46.97%)	(30.96%)	(69.04%)	(39.46%)	(51.36%)
Straight Hair	2762/6444	3682/6444	2762/11057	3682/18943
6444 Images (21.48%)	(42.86%)	(57.14%)	(24.98%)	(19.44%)
Wavy Hair	1802/10723	8921/10723	1802/11057	8921/18943
10723 Images (35.74%)	(16.80%)	(83.20%)	(16.30%)	(47.09%)
Wearing Earrings	286/7944	7658/7944	286/11057	7658/18943
7944 Images (26.48%)	(3.60%)	(96.40%)	(2.59%)	(40.43%)
Wearing Hat	744/1070	326/1070	744/11057	326/18943
1070 Images (3.57%)	(69.53%)	(30.47%)	(6.73%)	(1.72%)
Wearing Lipstick	69/16859	16790/16859	69/11057	16790/18943
16859 Images (56.20%)	(0.41%)	(99.59%)	(0.62%)	(88.63%)
Wearing Necklace	231/5085	4854/5085	231/11057	4854/18943
5085 Images (16.95%)	(4.54%)	(95.46%)	(2.09%)	(25.62%)
Wearing Necktie	2161/2162	1/2162	2161/11057	1/18943
2162 Images (7.21%)	(99.95%)	(0.05%)	(19.54%)	(0.01%)
Young	6654/23368	16714/23368	6654/11057	16714/18943
23368 Images (77.89%)	(28.47%)	(71.53%)	(60.18%)	(88.23%)

Table A.1: Attribute Label Distributions for CelebA-HQ

Appendix B

Network Structure Details

The tables in this section contain the network structures for each of the models discussed in this thesis.

Note that after each convolution layer we also apply a `LeakyReLU` non-linearity.

N in the tables represents the size of the conditional label, which is included only for the conditional versions of the models.

Table B.1: Network structure for the encoder component of the Generator network. The output is concatenated with N attribute values for a conditional label of size N.

Layer Type	**Kernel**	**Stride**	**Output Shape**
Input Image	-	-	$4 \times 1024 \times 1024$
Conv	1×1	1×1	$16 \times 1024 \times 1024$
Conv	3×3	1×1	$32 \times 1024 \times 1024$
Conv	3×3	1×1	$32 \times 1024 \times 1024$
Downsample	-	-	$32 \times 512 \times 512$
Conv	3×3	1×1	$64 \times 512 \times 512$
Conv	3×3	1×1	$64 \times 512 \times 512$
Downsample	-	-	$64 \times 256 \times 256$
Conv	3×3	1×1	$128 \times 256 \times 256$
Conv	3×3	1×1	$128 \times 256 \times 256$
Downsample	-	-	$128 \times 128 \times 128$
Conv	3×3	1×1	$256 \times 128 \times 128$
Conv	3×3	1×1	$256 \times 128 \times 128$
Downsample	-	-	$256 \times 64 \times 64$
Conv	3×3	1×1	$512 \times 64 \times 64$
Conv	3×3	1×1	$512 \times 64 \times 64$
Downsample	-	-	$512 \times 32 \times 32$
Conv	3×3	1×1	$512 \times 32 \times 32$
Conv	3×3	1×1	$512 \times 32 \times 32$
Downsample	-	-	$512 \times 16 \times 16$
Conv	3×3	1×1	$512 \times 16 \times 16$
Conv	3×3	1×1	$512 \times 16 \times 16$
Downsample	-	-	$512 \times 8 \times 8$
Conv	3×3	1×1	$512 \times 8 \times 8$
Conv	3×3	1×1	$512 \times 8 \times 8$
Downsample	-	-	$512 \times 4 \times 4$
Conv	3×3	1×1	$512 \times 4 \times 4$
Conv	4×4	1×1	$512 \times 1 \times 1$
AttrConcat	-	-	$512(+N) \times 1 \times 1$

Table B.2: Network structure for the decoder component of the Generator network. Note that the Skip connections are only included for the Infilling GAN model discussed in Section 3.4. The skip connections are connected to the corresponding mirrored layers in the encoder component and are concatenated to the result of the previous layer.
The input includes N attribute values for a conditional label of size N.

Layer Type	**Kernel**	**Stride**	**Output Shape**
Input Latent Vector	-	-	$512(+N) \times 1 \times 1$
Conv	4×4	1×1	$512 \times 4 \times 4$
Conv	3×3	1×1	$512 \times 4 \times 4$
Upsample	-	-	$512 \times 8 \times 8$
(Skip)	-	-	$1024 \times 8 \times 8$
Conv	3×3	1×1	$512 \times 8 \times 8$
Conv	3×3	1×1	$512 \times 8 \times 8$
Upsample	-	-	$512 \times 16 \times 16$
(Skip)	-	-	$1024 \times 16 \times 16$
Conv	3×3	1×1	$512 \times 16 \times 16$
Conv	3×3	1×1	$512 \times 16 \times 16$
Upsample	-	-	$512 \times 32 \times 32$
(Skip)	-	-	$1024 \times 32 \times 32$
Conv	3×3	1×1	$512 \times 32 \times 32$
Conv	3×3	1×1	$512 \times 32 \times 32$
Upsample	-	-	$512 \times 64 \times 64$
(Skip)	-	-	$1024 \times 64 \times 64$
Conv	3×3	1×1	$512 \times 64 \times 64$
Conv	3×3	1×1	$512 \times 64 \times 64$
Upsample	-	-	$512 \times 128 \times 128$
Conv	3×3	1×1	$256 \times 128 \times 128$
(Skip)	-	-	$512 \times 128 \times 128$
Conv	3×3	1×1	$256 \times 128 \times 128$
Conv	3×3	1×1	$256 \times 128 \times 128$
Upsample	-	-	$256 \times 256 \times 256$
Conv	3×3	1×1	$128 \times 256 \times 256$
(Skip)	-	-	$256 \times 256 \times 256$
Conv	3×3	1×1	$128 \times 256 \times 256$
Conv	3×3	1×1	$128 \times 256 \times 256$
Upsample	-	-	$128 \times 512 \times 512$
Conv	3×3	1×1	$64 \times 512 \times 512$
(Skip)	-	-	$128 \times 512 \times 512$
Conv	3×3	1×1	$64 \times 512 \times 512$
Conv	3×3	1×1	$64 \times 512 \times 512$
Upsample	-	-	$64 \times 1024 \times 1024$
Conv	3×3	1×1	$32 \times 1024 \times 1024$
(Skip)	-	-	$64 \times 1024 \times 1024$
Conv	3×3	1×1	$32 \times 1024 \times 1024$
Conv	3×3	1×1	$32 \times 1024 \times 1024$
Conv	1×1	1×1	$3 \times 1024 \times 1024$

Table B.3: Network Structure for the Discriminator Network. The output includes N attribute predictions for a conditional label of size N.

Layer Type	**Kernel**	**Stride**	**Output Shape**
Input Image	-	-	$4 \times 1024 \times 1024$
Conv	1×1	1×1	$16 \times 1024 \times 1024$
Conv	3×3	1×1	$32 \times 1024 \times 1024$
Conv	3×3	1×1	$32 \times 1024 \times 1024$
Downsample	-	-	$32 \times 512 \times 512$
Conv	3×3	1×1	$32 \times 512 \times 512$
Conv	3×3	1×1	$64 \times 512 \times 512$
Downsample	-	-	$64 \times 256 \times 256$
Conv	3×3	1×1	$64 \times 256 \times 256$
Conv	3×3	1×1	$128 \times 256 \times 256$
Downsample	-	-	$128 \times 128 \times 128$
Conv	3×3	1×1	$128 \times 128 \times 128$
Conv	3×3	1×1	$256 \times 128 \times 128$
Downsample	-	-	$256 \times 64 \times 64$
Conv	3×3	1×1	$256 \times 64 \times 64$
Conv	3×3	1×1	$512 \times 64 \times 64$
Downsample	-	-	$512 \times 32 \times 32$
Conv	3×3	1×1	$512 \times 32 \times 32$
Conv	3×3	1×1	$512 \times 32 \times 32$
Downsample	-	-	$512 \times 16 \times 16$
Conv	3×3	1×1	$512 \times 16 \times 16$
Conv	3×3	1×1	$512 \times 16 \times 16$
Downsample	-	-	$512 \times 8 \times 8$
Conv	3×3	1×1	$512 \times 8 \times 8$
Conv	3×3	1×1	$512 \times 8 \times 8$
Downsample	-	-	$512 \times 4 \times 4$
Conv	3×3	1×1	$512 \times 4 \times 4$
Conv	4×4	1×1	$1(+N) \times 1 \times 1$