INSTITUT FÜR INFORMATIK
UNIVERSITÄT AUGSBURG

Bachelorarbeit

An Extensive Study of the StyleGAN Architecture for Deep Image Synthesis

Florian Barthel

Gutachter:	Prof. Dr. Rainer Lienhart
Zweitgutachter:	Prof. Dr. Elisabeth André
Betreuer:	Stephan Brehm
Datum:	September 30, 2020

verfasst am
Lehrstuhl für maschinelles Sehen und Multimedia
Prof. Dr. Rainer Lienhart
Institut für Informatik
Universität Augsburg
D–86135 Augsburg, Germany
www.Informatik.uni-augsburg.de

Abstract

This thesis analyzes the StyleGAN architecture, developed by the Nvidia research group Tero Karras et al. In the context of an ablation study, each component of the StyleGAN will be analyzed separately. Afterwards, the results are evaluated using the Fréchet Inception Distance (FID) to objectively quantify the quality of the generated images. Next to the FID metric, the generated images are also inspected by hand in order to detect artifacts or patterns. The results of the ablation study showed that most components improve the generated images quality, while few of them actually worsen it. It also showed that all components had a similar impact on the model, pointing out that no single component is responsible for the high quality images, produced by the StyleGAN. Instead the combination of the components make the StyleGAN this successful. After the ablation study, the StyleGAN is also trained on a different dataset to demonstrate that the model also performs well on datasets with different image content.

Contents

1 Introduction

With recent advances in machine learning the popularity of image synthesis has grown substantially during the past few years. One of the major reasons for that was the introduction to Generative Adversarial Networks (GANs) by Ian J. Godfellow et. al in 2014 [Goo+14]. In their paper they proposed a revolutionary network architecture that allowed state of the art image synthesis using two deep neural networks with adversarial behaviour.
Since then, GANs have found applications in many fields such as photo and video editing or data augmentation for deep learning models, making it according to Yann LeCun "the most interesting idea in the last 10 years in machine learning" [Gui+20].
Some specific examples for applications of GANs are the prediction of the next frame in videos [LKC16], the reconstruction of missing regions in images [Yeh+16], the creation of super resolution images from low resolution images [Led+16] or even the data augmentation to create a robust classification network for the detection of the coronavirus from x-ray images [Kha+20].
In theory GANs are designed to produce an image distribution similar to the training image distribution, but in reality they can be very difficult to train due to an instability between the two networks. For that reason many research groups [BDS18; Zhu+17; Mao+16] have proposed many different techniques and architectures to overcome these training problems and improved quality of the generated images. One of the most successful architectures is the StyleGAN, introduced by the Nvidia research group Tero Karras, et. al [KLA18a]. In their paper, which won the 'Best Paper Honorable Mention' award of the CVPR conference in 2019, the research group proposes a new GAN architecture that uses techniques from style transfer literature to generate images with high resolution and high quality.
This thesis will analyze the StyleGAN model by applying an ablation study. An ablation study is an analysis method in which single components are removed from a complex system in order to understand the components functionality and influence on the system. For comparing the performance of all experiments with each other, this thesis will use the Fréchet Inception Distance, which is a metric that quantifies image quality in correlation to human judgement.
After the ablation study the StyleGAN will also be analyzed when trained on a different dataset. The therefore used dataset is the car dataset, created by the chair for Multimedia Computing and Computer Vision of the Augsburg University.

2 Related Work

2.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) have become a very popular research topic in recent years. According to google scholar, the paper 'Generative Adversarial Nets', published by Ian J. Goodfellow, et. al in 2014 [Goo+14], has already been cited more than 22,000 times, making it one of the most influential papers in the field of machine learning.
In the paper, Goodfellow et. al proposed a new architecture for image synthesis that consists of two deep neural networks with adversarial behavior. The first network (the generator), is trained to produce fake images from random input vectors. And the secrond network (the discriminator), is trained to distinguish whether an image has been created by the generator or was sampled from the training data, as shown in Figure 2.1.

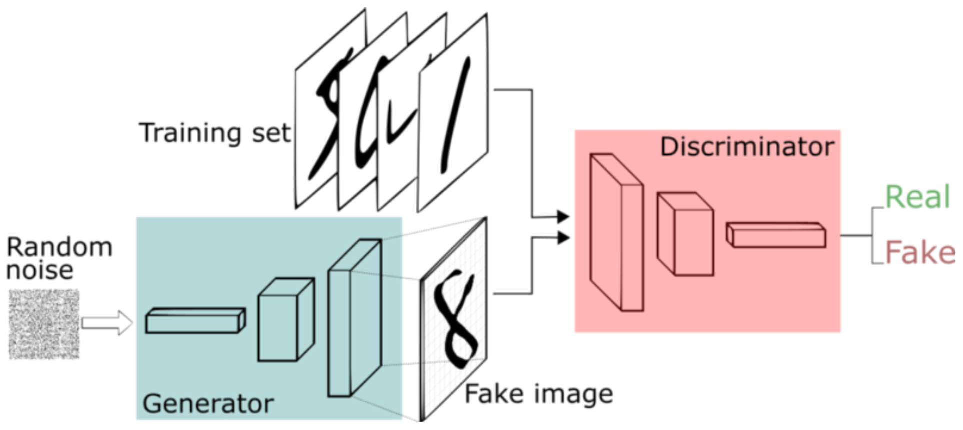

Figure 2.1: Traditional architecture for generative adversarial networks [Sil18]

As both networks are simultaneously improving to create and detect fake images over the course of the training, the image distribution of the generated images p_g converges to the image distribution of the training data p_{data} [Goo+14, Proposition 2]. Ideally, a global optimum is reached where $p_{data} = p_g$ [Goo+14, Theorem 1].

The idea of this approach can also be expressed in a non-cooperative minmax game over the value function $V(D, G)$ (2.1). On the one hand, the goal of the discriminator is to maximize the output of $V(D, G)$ by increasing the probability D to correctly label generated images $G(z)$ and training images x. And, on the other hand, the goal of the generator is to minimize $V(D, G)$ by mapping the random input vectors z to the same distribution as p_{data} and thereby decreasing the output for $\mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$.

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

This minmax game is being played between the generator and the discriminator until both models have reached a state at which they do not improve or adapt their output irrespectively to the output of the other model. This state is called the Nash equilibrium [MGN18].

If the model has been implemented and trained successfully, the output images produced by the generator can then look as shown in Figure 2.2.

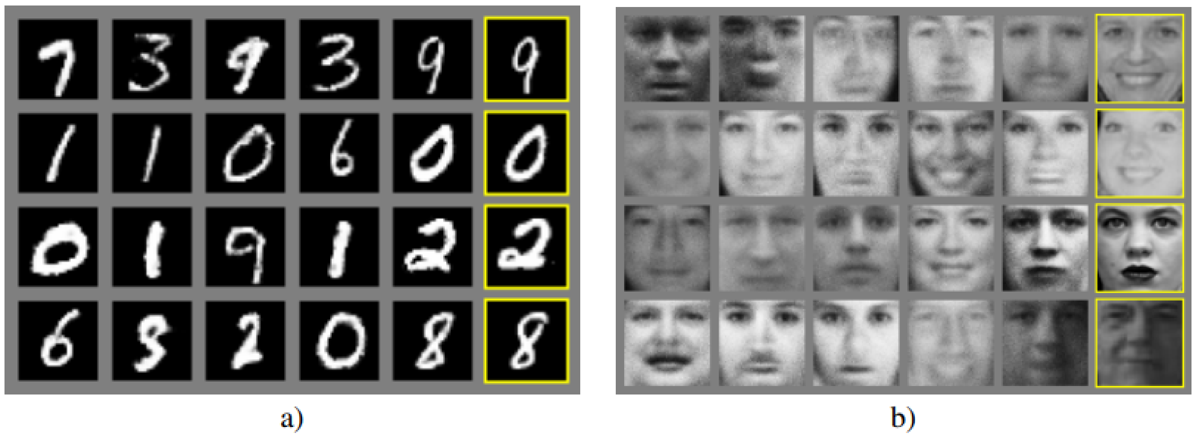

Figure 2.2: Example results of GANs by Ian J. Goodfellow. (a) is trained on the MNIST dataset and (b) is trained on the TFD dataset [Goo+14]. The yellow boxed images display the most similar image from the training dataset to the previous column, to demonstrate that the generator does not simply memorize the training data.

2.2 Challenges

In theory "if given enough capacity and training time" [Goo+14] the distribution of the generated images of GANs will match the distribution of the training images. In reality however those models can be very difficult to train, due to problems like non-convergence or mode collapse.

2.2.1 Non-Convergence

GANs are designed to converge towards a Nash equilibrium, where the discriminator and the generator do not improve anymore, irrespectively to the output of the other network. However, if the hyperparameters such as the learning rate were set poorly, it can happen that the models do not converge and instead oscillate with increasing magnitude [Goo16, Example 8.2].
Another example where the parameters can not converge to a Nash equilibrium is, when the discriminator outperforms the generator. This is often the case at the beginning of the training [Goo+14], where the generator produces random outputs. The discriminator then easily detects whether the given sample is from the dataset or the generator. As a result, the generator receives very large negative loss values and therefore large gradients, since it minimizes the term $\mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$ of the value function $V(D, G)$ 2.1 with:

$$\lim_{D(G(z)) \to 1} log(1 - D(G(z))) = -\infty \tag{2.2}$$

One solution for that problem is shown in section 5.1.1, regarding the loss functions of the generator.

2.2.2 Mode Collapse

Mode collapse is a specific form of non-convergence. It describes the state in which the generator produces images with a very limited range of variation (or modes) [Goo16]. This can happen, for instance, when the discriminator inspects the generated images separately without regarding the batch-wise statistics [Ach+18]. The generator is then able to focus on a single mode that the discriminator might have failed to detect as fake. As a result the generator generator is then enforced to focus even further on that single mode, until all generated images look the same. In Figure 2.3 the first row of result images shows the output of a working generator (trained on the MNIST dataset), that produces all numbers from 0 to 9. And the second row shows the output of a mode collapsed generator that outputs the same image for any random input vector.

A technique that helps to prevent GANs from mode collapsing is the minibatch standard deviation [Ach+18], which is discussed in section 5.7.

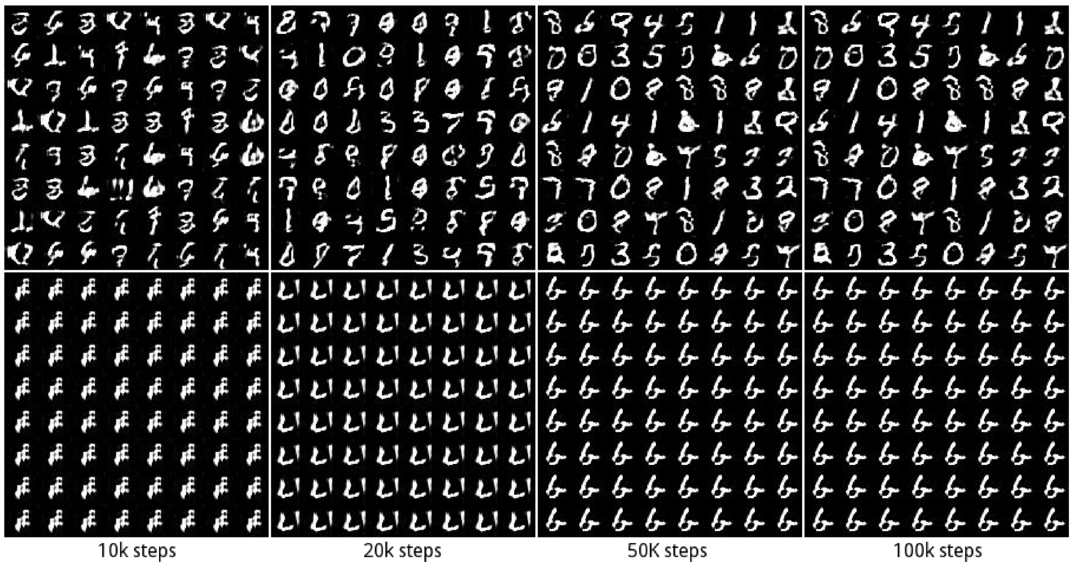

Figure 2.3: GAN model results with mode collapse (bottom row) and without mode collapse (top row) [Met+16].

2.3 Metrics

When it comes to determining the image quality for generated images of GAN models, it can be very difficult, time consuming and subjective to decide whether one set of images has a higher quality than another, as demonstrated in Figure 2.4. For that reason, it is necessary to define metrics that quickly quantify the quality for large amount of images in correlation to human judgement.
One of those metrics that has become very popular in recent years is the Fréchet Inception Distance, which will be used throughout this thesis to compare the images of different StyleGAN versions.

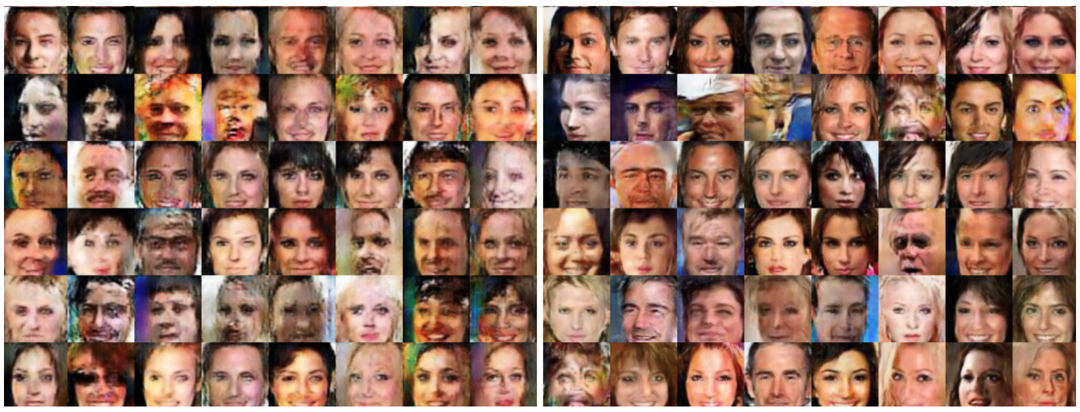

Figure 2.4: Two image distributions that are hard to compare without metrics (Left: FID 45 and Right: FID 13) [Heu+17, Figure A11]

2.3.1 Fréchet Inception Distance

The Fréchet Inception Distance (FID) was first introduced in 2019 by Martin Heusel et al. [Heu+17]. It is currently one of the most popular metrics for comparing different GAN models, that have been trained on the same dataset. The main idea of the FID is to quantify the difference between the feature embeddings of a pre-trained classification network for the generated images and the training images by comparing their statistics.

In practice the FID is calculated by taking a large number of images from the training dataset and the generated images (commonly 50 000 images each) and then computing the activation vectors from the last feature map of the pre-trained InceptionV3 model for each. The InceptionV3 model is a classification model which was developed by the Google engineers Christian Szegedy, et al in 2015 [Sze+15]. Both activation vectors with a shape of 50 000 x 2048 are compared by the following expression:

$$FID = \|\mu_1 - \mu_2\|_2^2 + Tr\left(C_1 + C_2 - 2\sqrt{C_1 C_2}\right) \tag{2.3}$$

Here μ_1 and μ_2 are the means of each activation and C_1 and C_2 are the covariances. The Tr operator stands for the trace linear algebra operation, which is the sum of the diagonal elements ($\sum_{i=1}^{n} a_{ii}$).

The effectiveness of this metric can be seen in Figure 2.5 where the FID is calculated between an image and a modified version of the same image. It shows that the FID correlates with the disturbance level of the image modifications.
By comparing the statistics for such large amounts of images with each other, the FID also penalizes image variation next to image quality.

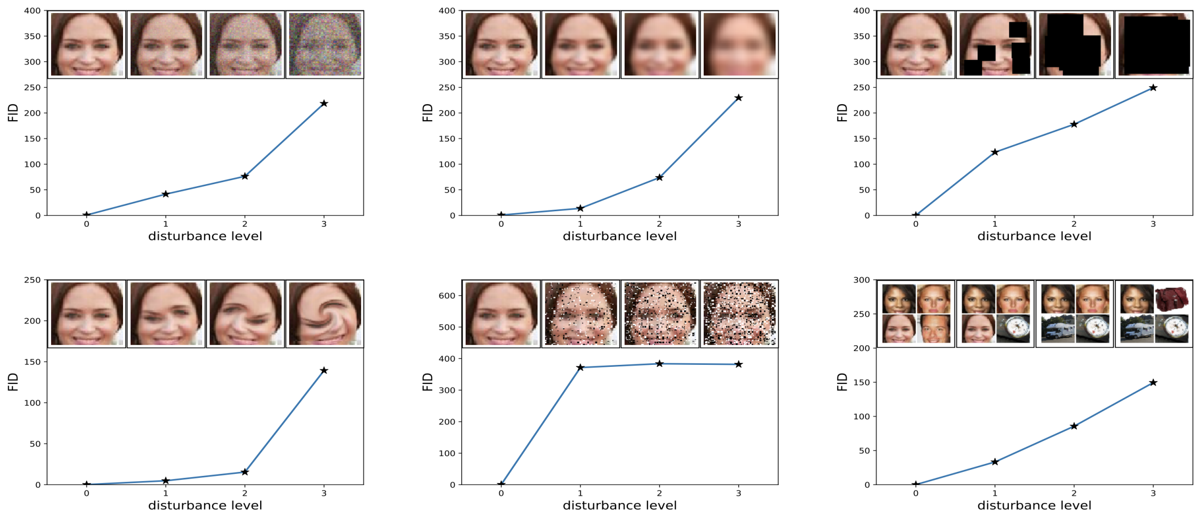

Figure 2.5: FID scores of modified images (lower is better) [Heu+17]

2.4 Adaptive Instance Normalization

One major component of the StyleGAN model that will be analyzed in this thesis is the style transfer. Style transfer denotes the technique of rendering the content of one image in the style of another image [HB17].
The style transfer operation that was chosen in the StyleGAN model is the Adaptive Instance Normalization (AdaIN) operation, which was introduced by Xun Huang and Serge Belongie in 2017 [HB17].

The AdaIN operation (2.5) is constructed similarly to the instance normalization operation (2.4). Like in instance normalization the input content x is first subtracted by its mean and then divided by its standard deviation. But instead of using trainable parameters γ and β as scale and bias, the AdaIN operation takes the mean of a second input, the style input y, as bias and the standard deviation of the style input as scale. In that way, the AdaIN applies the statistics of the style y onto the content x without requiring trainable parameters such as γ or β.

$$IN(x) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta \tag{2.4}$$

$$AdaIN(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \tag{2.5}$$

An example for style transfer with the AdaIN operation can be seen in Figure 2.6. Here the AdaIN operation is applied onto the feature embeddings of the style and the content, that were created by a pre-trained encoder network. The output of the AdaIN operation is then decoded by another pre-trained network to visualize the result of the style transfer.

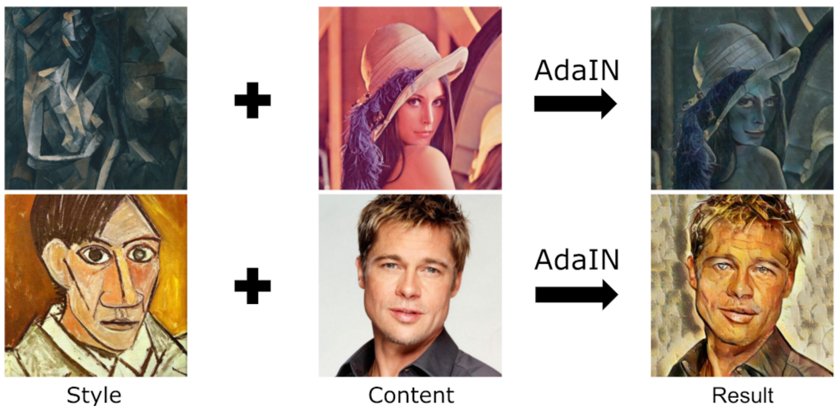

Figure 2.6: Example images for style transfer using the AdaIN operation using an encoder and a decoder network. (modified from [HB17])

3 StyleGAN Architecture

The StyleGAN model was first introduced in 2018 by the Nvidia research group Tero Karras et al. and is based on the previously, in 2017, published ProGAN model [Kar+17]. Like traditional GAN architectures the StyleGAN model also consists of a generator and a discriminator. While the discriminator used in the StyleGAN architecture is a conventional deep convolutional network, the generator was completely rebuilt to apply modern techniques from style transfer literature.
Before diving into the specific components that have been implemented in the StyleGAN model, this chapter will give a basic understanding of the model's architecture and two of its most important components. The first component is the style transfer and the second component the progressively growing structure.

The following two sections are based on the StyleGAN [KLA18a] and the ProGAN [Kar+17] paper.

3.1 Style Transfer

In traditional GAN architectures, the generator uses a random latent vector that is fed forward through the input layers of a neural network as shown in Figure 3.1 a. The StyleGAN generator however first maps the latent vector from the input latent space Z to an intermediate space W, which is also known as the disentangled latent space. This is realized by using a fully connected network (Mapping Network f, Figure 3.1 b). The mapped latent vector $w \in W$ is then taken as style input for the AdaIN style transfer operation, from section 5.6, to apply w as a style onto the feature maps of a deep convolutional neural network (Synthesis Network g, Figure 3.1 b).
This deep convolutional network, which is also called the synthesis network, is made out of multiple blocks that each consist of two convolutional layers. Those convolutional layers each have double the feature map resolution than the previous block, starting at a resolution of 4x4.
The synthesis network only receives a constant input as basis for the convolutional layer, so that it fully relies on the style inputs to adapt the produced image distribution to the training dataset. In-between each convolution and AdaIN operation Gaussian noise is added (B, Figure 3.1 b), which is resampled every training iteration and multiplied by a trainable weight that is initially set to zero.

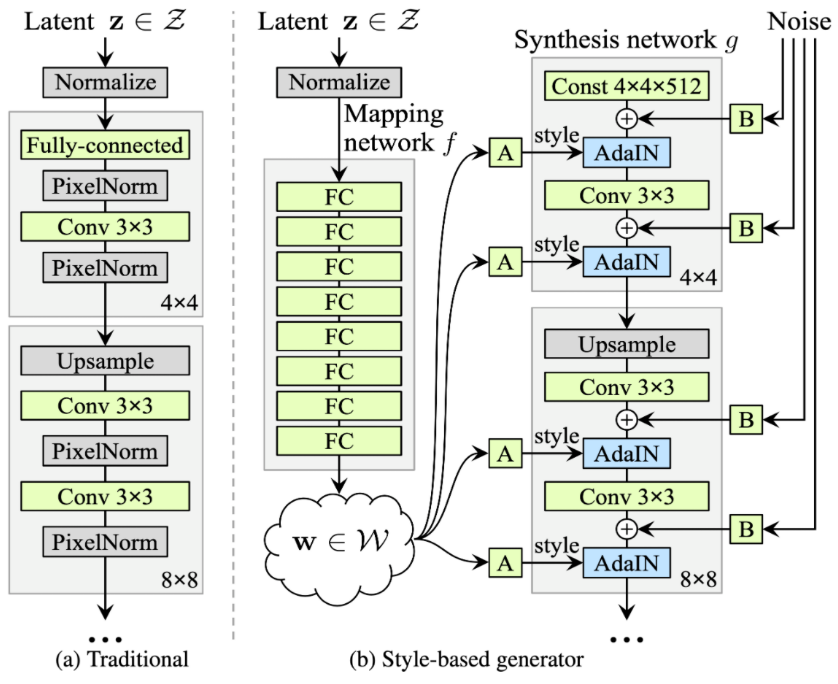

Figure 3.1: Comparing a traditional generator architecture (a), where the latent vector z is fed forward through a convolutional network, to the StyleGAN generator architecture (b), where the latent vector z is first mapped by a fully connected network f to $w \in W$, which is then used to control the output of the synthesis network g. [KLA18a, Figure 1]

3.2 Progressively Growing Structure

In order to improve the training stability and to reduce the training time, Karras, et al. [Kar+17] introduced a network structure, that dynamically increases the output resolution, by adding new layers to both the generator and the discriminator simultaneously over the course of the training (Figure 3.2).

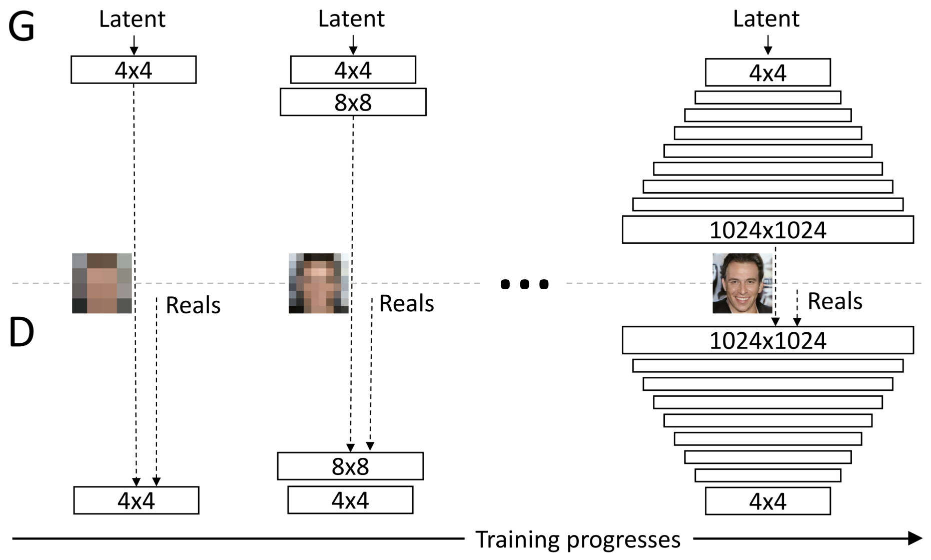

Figure 3.2: Progressive growing GAN structure over the training progress [Kar+17]

The Progressive Growing GAN (ProGAN) model initially produces 4x4 images in the generator and also detects fake images at a 4x4 resolution in the discriminator. After a number of images (600k - 800k) have been shown to the discriminator a new layer is added with twice the output resolution in the generator and twice the input resolution in the discriminator. This process is continued until the full resolution is reached.
Due to the small initial network size and the low resolution of the images that are fed to the discriminator at the beginning of the training, the progressively growing structure speeds up the training significantly [Kar+17]. Also by slowly increasing the output resolution and thereby the complexity of the GAN, the stability of the training increases [Kar+17], which helps to prevent non-convergence.

The idea of the ProGAN was inspired by prior approaches from Wang et al. [Wan+18] and Ghosh et al. [Gho+18], who have used multiple generator and discriminator networks with increasing image resolutions to improve the output resolution and the training stability. Instead of using multiple separate networks, the ProGAN structure only uses one single network for all resolutions combined. This has the advantage that new layers can be faded in smoothly, which prevents "sudden shocks to the already well-trained smaller-resolution layers" [Kar+17]. It is realized by using linear interpolation between the upsampled output from the earlier low-resolution layer and the output of the newly added layer, as shown in Figure 3.3. This is applied to both the generator and the

discriminator with mirrored architecture.
The interpolation magnitude α, which is initialized with 0.0 when a new layer is added, linearly increases to 1.0, at which point the network output comes exclusively from the new layer.

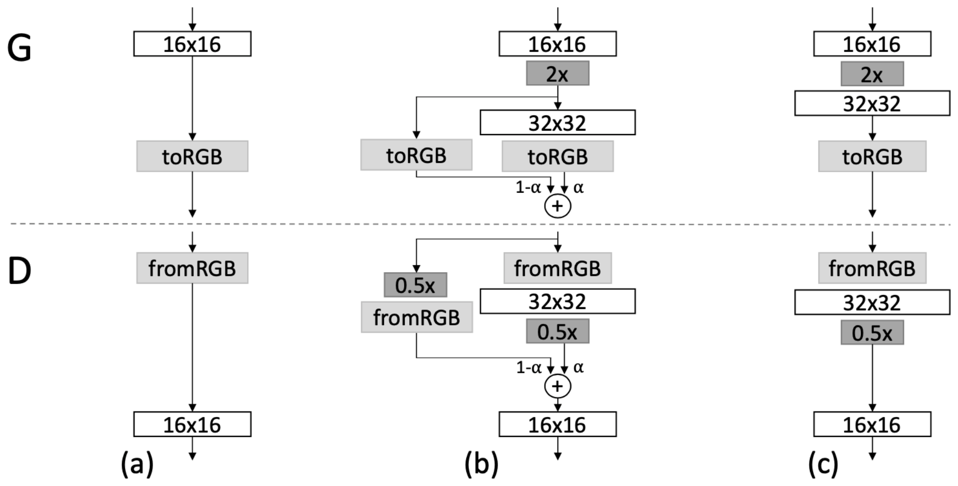

Figure 3.3: Fading from a 16x16 resolution (a) to a 32x32 resolution (c) by slowly increasing the influence of the convolution block on the output by increasing the parameter α (b) [Kar+17].

4 Disentanglement Metrics

The Linear Separability (LS) and the Perceptual Path Length (PPL) are both metrics that were published along with the StyleGAN paper [KLA18a].
Unlike the FID, those metrics do not quantify the image quality by calculating the difference between the generated image distribution and the training dataset distribution. Instead they evaluate the disentanglement of the latent space Z and especially for the StyleGAN model the disentanglement of the latent space W. The disentanglement of the latent space describes how well the latent space can be split into linear subspaces, where each subspace corresponds to single attributes in the output images of the generator [KLA18a]. The main goal of those metrics is to demonstrate that the mapping network of the StyleGAN helps the generator to "'unwarp' W so that the factors of variation become more linear" [KLA18a], which ideally makes it easier for the generator to produce more realistic images.
Further studies on the efficiency of the mapping network will be made in chapter 5.5.

Earlier research has already proposed several metrics to quantify the disentanglement of the latent space. But in contrast to most existing disentanglement metrics, the LS and the PPL do not require an encoder network. This makes the metrics easy to use and applicable to any given GAN with a latent space.

The following explanations for the LS and the PPL are based on the StyleGAN paper [KLA18a].

4.1 Linear Separability

The LS metric quantifies how well the latent spaces Z and W can be separated according to binary image attributes, such as male/female, young/old or smiling/not smiling. This is done by using a pre-trained neural network for each attribute (40 in total), to classify the generated images into two classes. This classification is then be applied to the corresponding points in the latent spaces Z and W, which are then separated using a support vector machine (SVM) as shown in Figure 4.1.

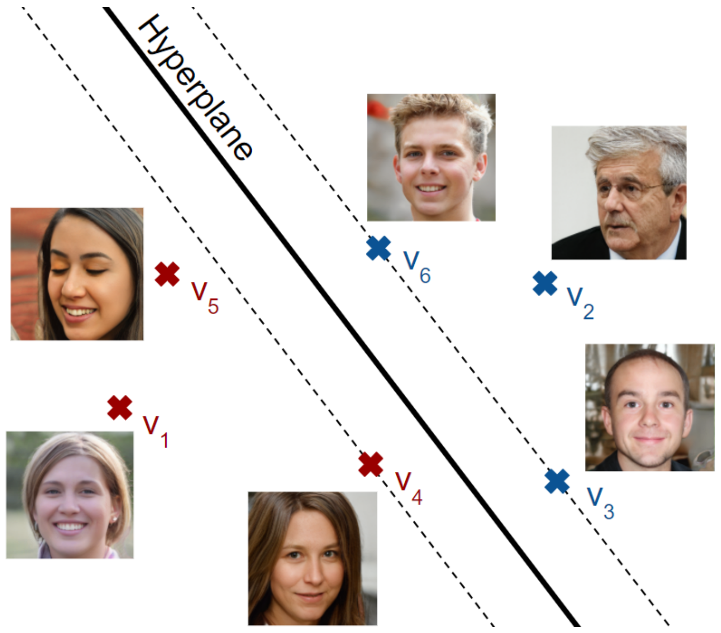

Figure 4.1: Example illustration of separating latent vectors, labeled by male and female, using a SVM. The dotted lines illustrate the margin between the hyperplane and the the samples.

Support vector machines have been developed in 1995 by Cortes and Vapnik [CV95] and describe a method of separating labeled data into two spaces by finding a hyperplane that divides the data by their classes while having the maximum margin to all instances. Since this task is not always possible without increasing the dimension of the space, some of the instances are then classified as the opposite to the classification of the pre-trained neural network, leading to a new classification. The classification of the pre-trained network and the classification of the SVM can then be compared by calculating the conditional entropy.
The conditional entropy $H(X|Y)$ measures "how much additional information is required to determine the true class of a sample, given that we know on which side of the hyperplane it lies" [KLA18a]. This means that similar classifications have a low conditional entropy and vice versa.
The conditional entropy is defined by the following expression [CT06, p. 16], where Y

is a set of instances classified by the SVM and X is a set of instances classified by a pre-trained network:

$$H(X|Y) = -\sum_{y \in Y} p(y) \left[\sum_{x \in X} p(x|y) \log p(x|y) \right] \tag{4.1}$$

In order to determine the LS value 200k images are generated by the generator network and then classified by 40 different pre-trained networks into binary classes. The resulting 200k outputs of the classification networks are then sorted by confidence, in order to filter the least 50% confident outputs. After that, a SVM is applied to determine a spacial classification of the latent-space points in Z or in W, corresponding to the image classification from the pre-trained network for each image attribute. At last, the conditional entropy between the network classification and the SVM classification is calculated and summed up for each attribute, leading to the following expression:

$$LS = \exp\left(\sum_i H(X_i|Y_i) \right) \tag{4.2}$$

Here Y_i stands for the predicted class by the SVM and X_i for the predicted class by the pre-trained network. i denotes the index of the current attribute $(1 \leq i \leq 40)$.
In the end the result is also exponentiated to receive a value with linear scale instead of a logarithmic scale, which the conditional entropy operator produced.
A low LS score therefore indicates similar classifications from the pre-trained neural network and the SVM, indicating a well disentangled latent space.

Since the computing time of the LS metric takes about 21 hours when using a single GTX 1080 Ti, this metric will only be applied when comparing the results of the baseline model to the results of the paper in section 5.1.3 and when analyzing the mapping network in section 5.5. For the same reason the LS will only be calculated for a single model checkpoint, instead of generating a graph over the course of the training.

4.2 Perceptual Path Length

The PPL is a metric that measures the smoothness of image transitions when interpolating between two latent vectors v_0 and v_1 within the latent spaces Z and W. The idea behind this metric is, that a well disentangled latent space should only show small differences in the output images when interpolating through it while heavily entangled latent spaces show big differences.

Like the LS metric, the PPL can also be applied to both latent spaces Z and W. The only difference being, that in the disentangled latent space W linear interpolation (lerp) 4.3 is applied, while using spherical interpolation (slerp) 4.4 [Sho85] within the latent space Z. This is done to prevent the interpolation paths, within the latent space Z, to "traverse locations that are extremely unlikely" [Whi16], since the latent vectors were sampled from a Gaussian distribution during the training.

$$\text{lerp}(v_0, v_1, t) = (1-t) \cdot v_0 + 1 \cdot v_1 \tag{4.3}$$

$$\text{slerp}(v_0, v_1, t) = \frac{\sin{(1-t)\theta}}{\sin\theta} \cdot v_0 + \frac{\sin t\theta}{\sin\theta} \cdot v_1 \tag{4.4}$$

In the spherical interpolation equation, θ denotes the angle between the two vectors v_1 and v_2 ($\theta = \cos^{-1} \frac{v_1 \cdot v_2}{|v_1| \cdot |v_2|}$) [Sho85].
Both interpolation techniques are illustrated in figure 4.2, showing interpolation paths within a simplified three-dimensional latent space.

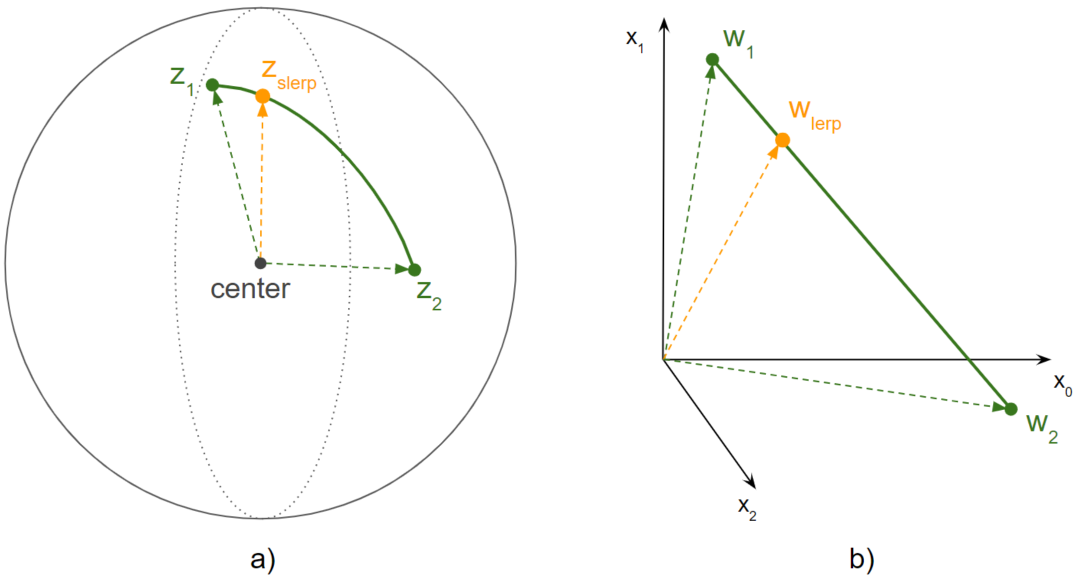

Figure 4.2: Example illustration of an interpolation path between two vectors within the simplified latent spaces Z (a), using spherical interpolation and W (b), using linear interpolation.

In order to calculate the PPL value, first two random latent vectors (z_1, z_2 or w_1, w_2) are sampled, using the same random distribution that was used during training. Those two vectors are then used to calculate a third latent vector, the interpolation vector (z_{slerp} or w_{lerp}), along the corresponding interpolation path between the two random vectors. This interpolation vector can either be calculated by the 'full path sampling' method or the 'endpoint sampling' method.

When using the 'full path sampling' method, this interpolation vector is placed at any random position of the interpolation path, as shown in Figure 4.3 (a), while when using the 'endpoint sampling' method, the interpolation vector is placed on top of one of the endpoints z_1 or w_1, as shown in Figure 4.3 (b).
After the interpolation vector has been determined, a new vector z_ϵ or w_ϵ, placed close to the interpolation vector, is calculated, by moving along the interpolation path for distance of 0.01% of the length of the path ($\epsilon = 10^{-4}$).

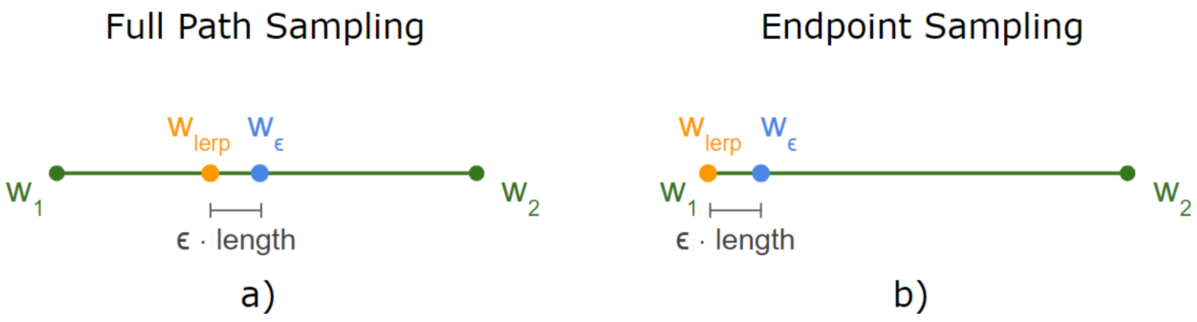

Figure 4.3: The difference between full path sampling method (a), where the interpolation vector is set at a random position along the interpolation path and endpoint sampling (b), where the interpolation vector is placed at the position of one of the endpoints.

The resulting output images of the latent vectors z_{slerp} / w_{lerp} and z_ϵ / w_ϵ are then generated and compared by calculating the distance of the embedding vectors from a pre-trained network. The therefore chosen network is the VGG16 model [SZ14], which is a deep neural network used for image classification and detection.
The result of the PPL metric is then determined by calculating the expected value of image distances for a large amount of random vector pairs z_1 and z_2 (100k pairs in the implementation [KLA18c]), leading to the following two expressions (4.5 for latent space Z and 4.6 for W):

$$PPL_Z = \mathbb{E}\left[\frac{1}{\epsilon^2} d(G(\text{slerp}(z_1, z_2, t)), G(\text{slerp}(z_1, z_2, t + \epsilon)))\right] \tag{4.5}$$

$$PPL_W = \mathbb{E}\left[\frac{1}{\epsilon^2} d(g(\text{lerp}(f(z_1), f(z_2), t)), g(\text{lerp}(f(z_1), f(z_2), t + \epsilon)))\right] \tag{4.6}$$

Here f denotes the mapping network, g the synthesis network and G is the generator that combines f and g ($G = g \circ f$). d is the distance between the embedding vectors of the VGG16 model. When using full path sampling t is sampled from $\mathcal{U}(0, 1)$ and when using endpoint sampling $t \in \{0, 1\}$. At the end the result is normalized by dividing the output of d by ϵ^2.
Similar to the LS metric, a low PPL value indicates less entanglement of the latent space.

5 Ablation Study

An ablation study is an analysis method that has its origins in medical literature. It was commonly used in the field of neuroscience to understand complex systems such as the nervous system and especially the human brain [Mey+19]. The idea of an ablation study is to remove single components from a complex system to compare the behaviour of the system before and after, in order to understand the components influence on the whole system.
This thesis will apply the ablation study technique on the StyleGAN model by training the StyleGAN from scratch after removing and modifying its main components. The results of these experiments will then be compared to a baseline model. A baseline model is a model that was trained with all components activated and default parameters taken from the StyleGAN paper [KLA18a].

5.1 Baseline Model

The default configuration that will be used as reference for all experiments was taken from the official tensorflow implementation of the StyleGAN, published by Tero Karras [KLA18c]. It has all parameters set as default, in order to reproduce the same results from the paper and also provides scripts to run the metrics described in section 4.
Although the StyleGAN is capable of generating images at a resolution of 1024x1024 pixels, all the following experiments were trained on a lightweight version of the network that only uses a 256x256 output resolution to reduce the overall training time. In order to reduce the training time even further, the network will also only be trained for 15 million training images, instead of 25 million as done by the Nvidia research group.
With this configuration a whole training takes about 14 days using a single GTX 1080 Ti and 8 days using two.
To achieve the high resolution results from the paper, the Nvidia research group has trained the StyleGAN using 8 Tesla V100 GPUs for 6 days and 14 hours [KLA18c].

5.1.1 Loss Functions

The loss functions that were used in the baseline model are the 'logistic nonsaturating loss' for the generator and the 'logistic loss with simple gradient penalty' for the discriminator [KLA18c]. Both of these loss functions were designed closely to the value function $V(D,G)$, from section 2.1:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{5.1}$$

Logistic Nonsaturating Loss

The logistic nonsaturating loss has its origin from Ian Goodfellow et. al, who have introduced this loss function along with the 'Generative Adversarial Nets' paper [Goo+14]. It can be described by the following expression:

$$\text{Loss}_G = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\sigma(D(G(z_i)))\right) \tag{5.2}$$

where N is the batch size, z_i is the noise input for the generator and σ is the logistic sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{5.3}$$

The word 'nonsaturating' describes that the loss function was designed to maximize the term $\log\left(\sigma(D(G(z_i)))\right)$ instead of minimizing $\log\left(1-\sigma(D(G(z_i)))\right)$, as implied by the right addend of the value function $V(D,G)$ (5.1). Doing so prevents the loss from saturating, when the discriminator outperforms the generator, which is often the case at the beginning of the training [Goo+14] and could lead to non-convergence of the model parameters.

Logistic Loss with Simple Gradient Penalty

The logistic loss with simple gradient penalty, used for the discriminator loss in the StyleGAN, is a loss function that first calculates the loss as suggested by the value function $V(D,G)$ 5.1 with the following expression:

$$\text{Loss}_D = \frac{1}{N}\sum_{i=1}^{N} -\left[\log\left(\sigma(D(x_i))\right) + \log\left(\sigma(D(G(z_i)))\right)\right] \tag{5.4}$$

and then adds the sum of the gradients of the discriminator loss over the fake outputs, multiplied by a design parameter γ that controls the magnitude of the gradient penalty:

$$\text{LossGP}_D = \text{Loss}_D + \gamma\nabla\left(\frac{1}{N}\sum_{i=0}^{N} - \log\left(\sigma(D(G(z_i)))\right)\right) \tag{5.5}$$

Here x_i are the real images from the training dataset and $G(z_i)$ are the fake images produced by the generator.
By adding the gradients to the loss function, the parameters of the discriminator are enforced to be small, which leads to a more stabilized training.

5.1.2 Training Dataset

The training dataset that was used throughout the whole ablation study is the Flickr-Faces-HQ (FFHQ) dataset. This dataset was also created and published by Tero Karras et al. along with the StyleGAN paper [KLA18a].
The dataset consists of 70 000 images of faces at a 1024x1024 resolution that were automatically aligned, cropped and filtered [KLA18b] to provide a high resolution dataset with few noise and similar spacial distributions. Example images of the FFHQ dataset are shown in Figure 5.1.
During the training the images were augmented by randomly flipping them along the vertical axis.

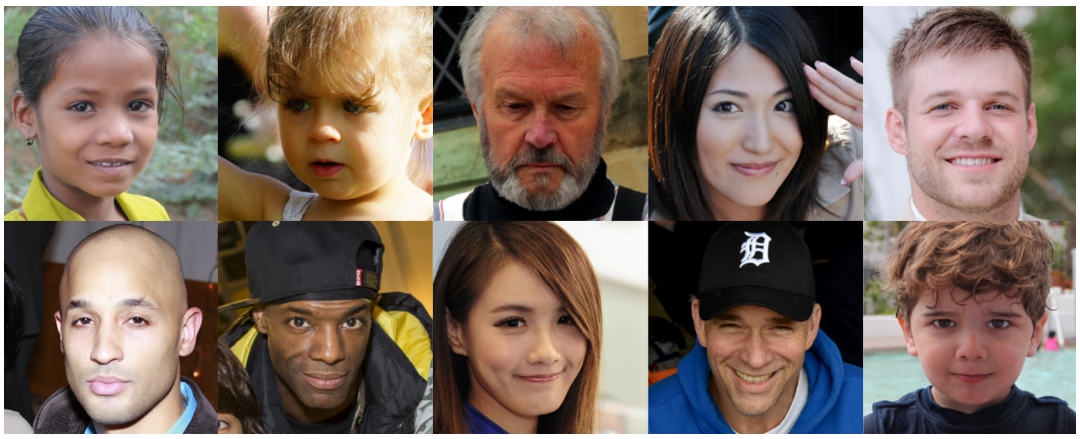

Figure 5.1: Example images of the FFHQ dataset.

5.1.3 Training Results

After training the StyleGAN model with the default configuration described in 5.1, the generator produced an image distribution with the following metric scores shown in Table 5.1 and Figure 5.2:

Metric	Baseline Model	Paper Results
FID	6.05	4.40
PPL W (end)	60.9	195.9
PPL W (full)	70.7	234.0
PPL Z (end)	210.4	666.1
PPL Z (full)	209.8	664.9
LS W	3.6	3.7
LS Z	169.6	165.0

Table 5.1: Metric scores of the baseline model and the results of the StyleGAN paper [KLA18a], [KLA18c] (the LS and PPL were measured at the model checkpoints with the lowest FID). Lower is better for all metrics.

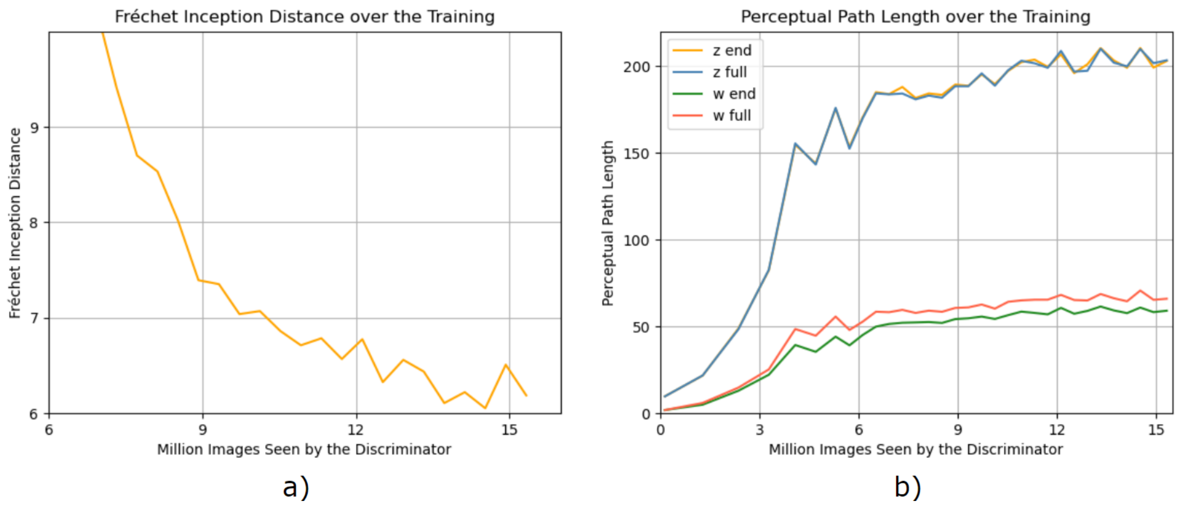

Figure 5.2: (a) FID during the training of the baseline model, starting at 6 million training images. (b) The PPL values over the course of the training of the baseline model. (lower is better)

Fréchet Inception Distance

Compared to the results of the StyleGAN paper, the FID of the baseline model is 1.65 higher. This is most likely due to the shorter training and some stochastic variation, since the training of GANs is non-deterministic. The trend of the FID graph in Figure

5.2 also shows that the FID will probably decrease even further if given more training time. Another reason for the margin in FID can be the difference in output resolution of the two models. Even though the images from the dataset, that the generated images are be compared to in the FID metric, also only have a resolution of 256 x 256, it still might impact the FID value.

Perceptual Path Length

Although the FID is worse in the baseline model, it turns out, that the latent spaces are, according to the PPL metric, much more disentangled in the baseline model than in the results of the StyleGAN paper. While for instance the PPL (in W with full path sampling) is 70.7 for the baseline model, the results from the paper scored 234.0, which is more than three times as much. One explanation for this discrepancy might be, that the high resolution models have to fit more information and variation into the same latent space than the low resolution models. This then can lead to stronger image transitions when interpolating through the latent space.

The graphs for the PPL and FID in Figure 5.2 show that even though the FID improves over the course of the training, the entanglement of the latent spaces keeps increasing. This indicates that improvements in FID come at a cost of the latent space entanglement. The same effect was also observed by the Nvidia research group (Tero Karras, et. al), who is referring to future work to analyze ”whether this is unavoidable, or if it were possible to encourage shorter path lengths without compromising the convergence of FID” [KLA18a].
It is also noticeable, that the graphs for the full path sampling and the endpoint sampling in the latent space Z are very similar to each other. This is because the latent space Z does not have any kind of mapping and is therefore equally defined throughout the whole interpolation path between two random vectors, given they were sampled from a Gaussian distribution and spherical interpolation was applied. For this reason, in this thesis, only one of the sampling methods will be applied when calculating the PPL in the latent space Z.

Linear Separability

According to the LS metric both disentangled latent spaces are similarly well separable. This indicates that the binary attributes used in the LS metric do not focus on such fine details, which would only be visible in higher resolutions and also that the mapping of the latent space does not change a lot after 15 million training images.

Generated Images

When inspecting the generated images at the model checkpoint with the lowest FID of 6.05, most faces look very realistic and have a lot of detail and variation considering age, gender, hair and many other attributes as shown in Figure 5.3.
Some of the images, although, can still be easily detected as fakes due to unusual shapes in the background (Figure 5.4 a), faces covered by poorly defined objects such as microphones (Figure 5.4 b), water droplet like artifacts at the edge of the faces (Figure 5.4 c) or unnatural face shapes (Figure 5.4 d).
The water droplet like artifacts will be further analyzed in section 5.6 about the AdaIN operator.

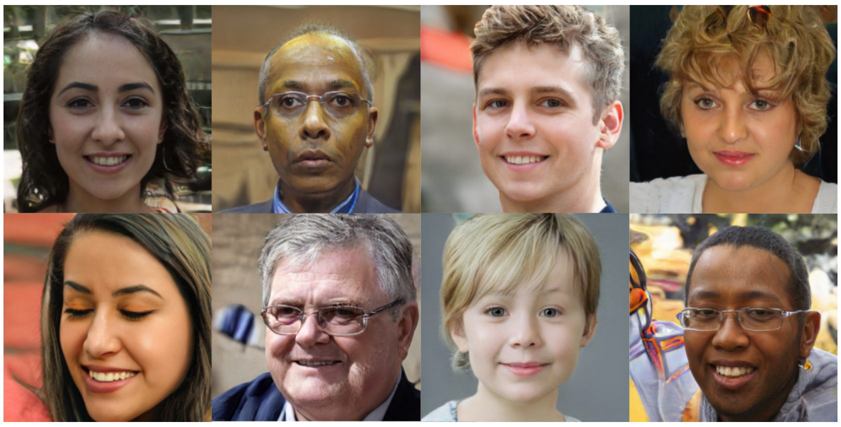

Figure 5.3: Examples for realistic looking images generated by the baseline model at the checkpoint with the lowest FID.

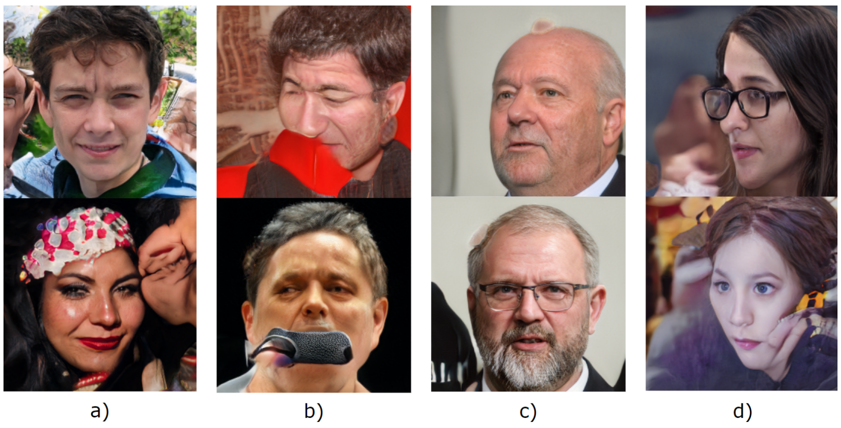

Figure 5.4: Example images with poor quality generated by the baseline model at the checkpoint with the lowest FID, due to (a) unusual shapes in the background, (b) faces covered by objects with bad quality, (c) water droplet effects, or (d) unnatural face shapes. All images have been generated by the baseline model at the checkpoint with the lowest FID.

5.2 Style Mixing

The first component that will be analyzed in the ablation study is the style mixing technique. Style mixing is a regularization method, developed by Tero Karras, et. al [KLA18a], that combines the styles from two images by exchanging the latent vectors for the AdaIN operations during a forward pass at a random layer within the generator. This regularization method was designed to "prevent the network from assuming that adjacent styles are correlated" [KLA18a]. Mixing the latent vectors then ideally leads to more image variation and decreases the chance of mode collapse.
The style mixing component is realized by first calculating the disentangled vectors w_1 and w_2 for two different latent vectors z_1 and z_2. After that, a random layer-index is determined at which w_1 will be replaced by w_2 for all further AdaIN operations within the generator.

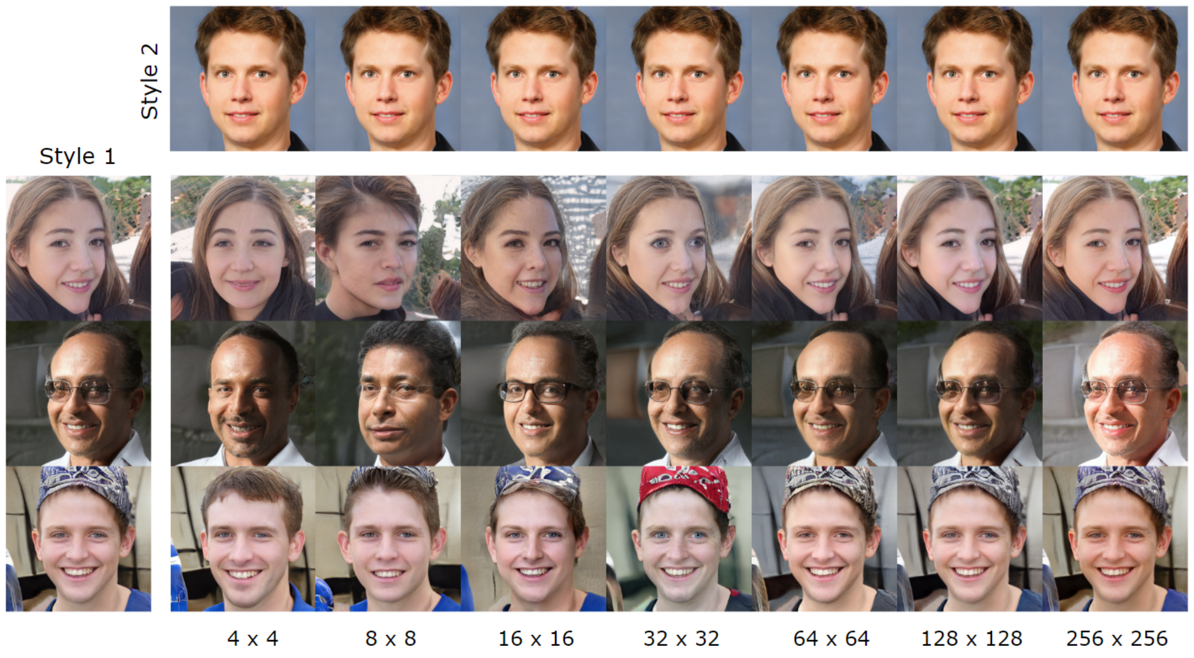

Figure 5.5: Generator output of the baseline model when exchanging the latent vector from style 1 with the latent vector from style 2 for one resolution block. It shows that earlier layer (with 4x4 to 32x32 resolution) of the generator mostly affect coarse attributes such as pose, age or gender and later layer (with 64x64 to 256x256 resolution) mainly affect finer face details, the background textures and the brightness.

Figure 5.5 shows that depending on the depth of the layer at which the latent vector is swapped out, different attributes of the face will be affected. Earlier layers (with 4x4 to 32x32 resolution) of the generator mostly affect coarse attributes such as pose, age or gender while later layers (with 64x64 to 256x256 resolution) mostly affect finer details, background textures and the brightness of the image. Figure 5.5 shows that for

example the glasses and the hat of the lower two rows disappear when mixing at the 4x4 or 8x8 layer, while only some details in the background gets changed when mixing at the 128x128 layer.
According to the results of the StyleGAN paper [KLA18a], it turned out to be best to train the StyleGAN by mixing two latent vectors during training, while using only one during image synthesis [KLA18c]. Better results were also achieved when only applying the style mixing to 90% of the training iterations [KLA18a, Table 2].
For the ablation study, a new StyleGAN model will be trained without the style mixing component, meaning that the model will only use one single latent vector throughout all AdaIN operations.

Removing Style Mixing

Training the StyleGAN model without the style mixing component, turned out to be very unstable. The graph for the FID in Figure 5.6 (a) shows that until 12 million training images, the model performed similarly to the baseline model, scoring a FID of 6.49 at best, but reaches very large values after 12 million training images that go up to 60.48.

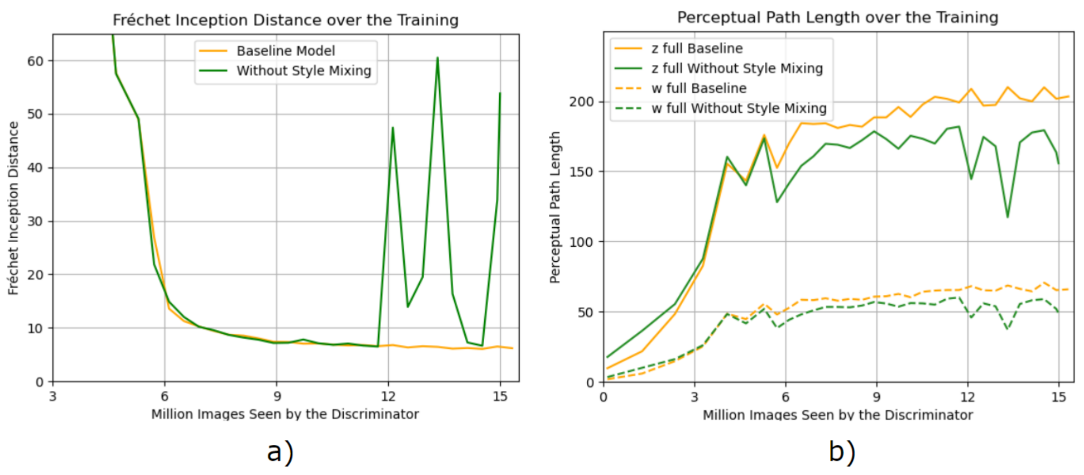

Figure 5.6: (a) FID of the baseline model (lowest 6.05) and model without the style mixing component (lowest 6.49), starting at 3 million training images seen by the discriminator. (b) The perceptual path length over the training of the baseline model and the model without style mixing. (lower is better for both metrics)

The spike in FID, after 12 million training images, can also be observed when inspecting the generated images from the model checkpoints over the course of the training. Figure 5.7 shows the images generated by the StyleGAN model trained without Style Mixing, before and after that spike. It is visible that although the generated faces have lost a lot of quality after the 12.1 million checkpoint, the images still show the same coarse

features such as pose, gender or hair and also have kept the same color. Considering that the earlier convolutions of the StyleGAN mostly affect the coarse attributes and the latter convolutions affect the color and brightness, as shown in Figure 5.5, it indicates that mostly the weights of the intermediate layers were very unstable at the spike in FID while the weights of the first and last layers remain similar. Although Figure 5.5 has been created by the baseline model, the same results, on which layers control which attributes, were observed with the StyleGAN model without the Style Mixing component.
Another reason why the images still show similar coarse attributes and colors, even though the training has become very unstable, is the exponential moving average of the generator weights, discussed in section 5.9, which is a method to prevent drastic changes of the model parameters.

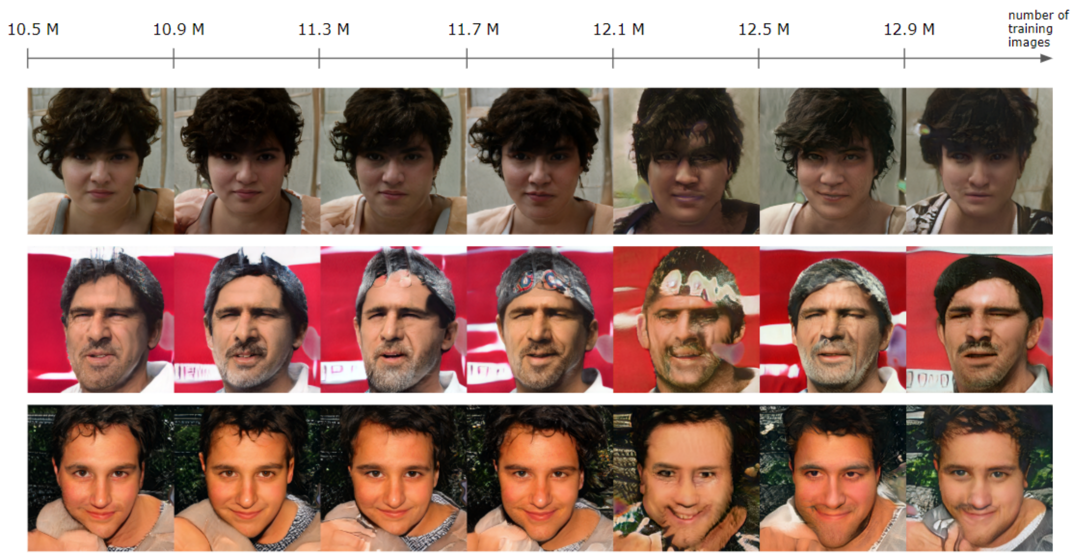

Figure 5.7: Output images of the model without Style Mixing, showing the images before and after the spike in FID at 12 million training images.

When comparing the perceptual path length of the baseline model to the modified model trained without style mixing throughout the training in Figure 5.6 (b), it shows that the entanglement in both latent spaces W and Z is lower without the style mixing component. This observation was also made by the Nvidia research group, who hypothesized that the added variation, caused by the style mixing, makes it more difficult for the mapping network to keep the latent space disentangled [KLA18a].

In order to be certain that the training instability was caused by the missing style mixing component, both the baseline model and the model without style mixing should have been trained multiple times. Due to the long computing time of up to 14 days, this was not possible without leaving out other experiments.
However compared to all future experiments, which are similar to the baseline model, the style mixing experiment was the only experiment in which this instability and such large spikes in FID could be detected.

5.3 Stochastic Variation

The second experiment will analyze the stochastic variation component of the StyleGAN. This component was designed to realistically transfer random features of real human portraits, ”such as the exact placement of hairs, stubble, freckles, or skin pores” [KLA18a] onto the generated images, by providing the generator with additional noise. Traditionally the generator network only receives the noise through the input layer of the network, causing it to invent a way to create variation in each layer only from previous activation layers. Doing so uses up network capacity and can cause repetitive patterns in the output images [KLA18a].
The stochastic variation component used in the StyleGAN architecture avoids both of those problems by adding pixel-wise Gaussian noise to the generator network, after each convolution layer, as shown in Figure 5.8 a.
The noise that is added will also be multiplied by a trainable weight which is initially set to zero. This way the generator can control itself how much noise will be added in each layer.

Removing Stochastic Variation

For the following experiments the stochastic variation component will be examined, by first analyzing the results of the baseline model when turning off the stochastic variation after it has been trained using it and second by training a new StyleGAN model without utilizing the component at all, as shown in Figure 5.8 b.

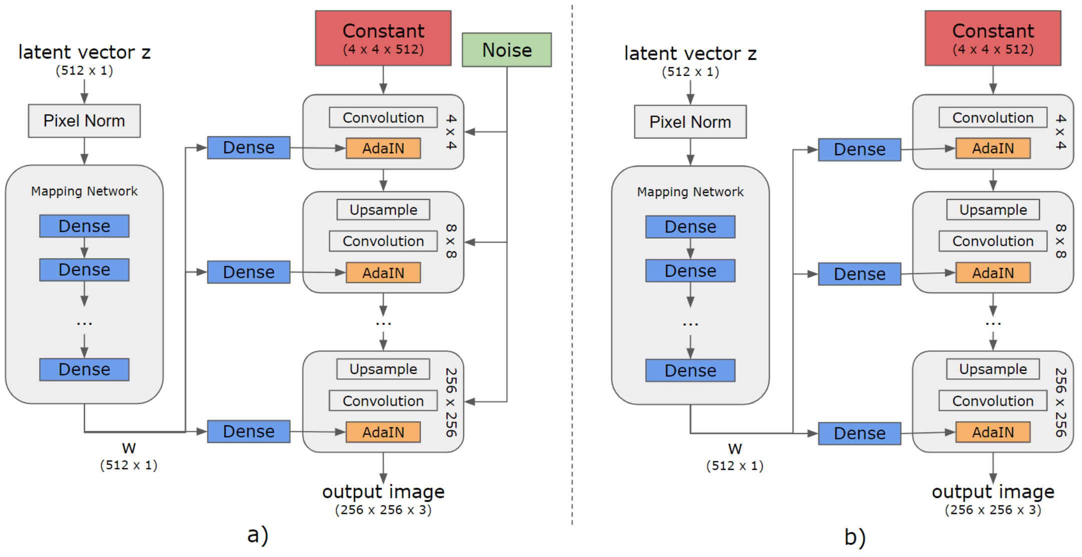

Figure 5.8: Baseline generator (a) compared to the modified generator without stochastic variation (b). (Inspired by [KLA18a, Figure 1])

First, the baseline model with and without noise and the model trained without stochastic variation are compared, according to their FID values. It shows in Figure 5.9 that the baseline model with and without stochastic variation produce very similar FID values, while the FID of the model trained without stochastic variation converges bit slower at the start of the training. This suggests that the stochastic variation component helps the StyleGAN model to converge faster to the distribution of the training dataset at the beginning of the training, although has only little effect on the FID when turning it off after it has been trained using it.
Between the three models the lowest FID value was measured at the baseline model without stochastic variation with a FID of 6.04 followed closely by the baseline model with stochastic variation with 6.05. The model trained without stochastic variation performed the worst scoring the highest FID of 6.19.

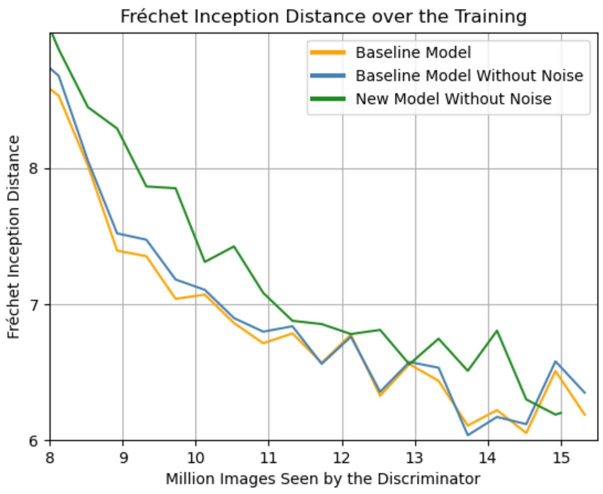

Figure 5.9: FID compared between the baseline model with stochastic variation (lowest 6.05) and without stochastic variation (lowest 6.04) and the new StyleGAN model that was trained without stochastic variation (lowest 6.19), starting at 8 million training images shown to the discriminator (lower is better).

When inspecting the images generated by the baseline model with and without noise (Figure 5.10 a and b), it shows that when training with noise and removing it in image synthesis, a lot of detail and structure from the skin and the hair disappears and also the background textures become a lot smoother.
Figure 5.10 c also shows the standard deviation of 100 images that were generated using the same input vector. The brighter an area is the more it was changed by the added noise. The resulting standard deviation image underlines that the stochastic variation component mainly impacts the fine details of the face such as the hair, the skin pores or the background textures as described by the StyleGAN paper [KLA18a].
A reason why the baseline model without noise actually performed slightly better in FID than with noise, could be that the background textures of the images become more blurry when generated without noise, which is also often seen in the images of the FFHQ dataset.

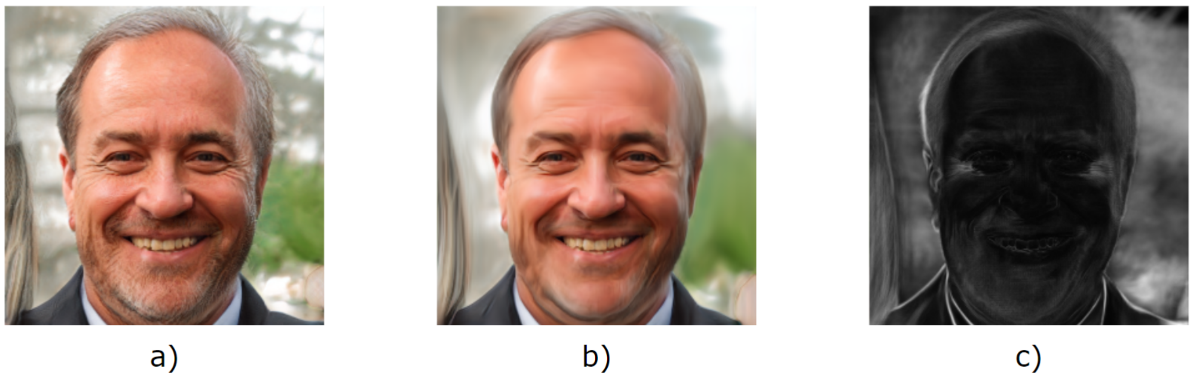

Figure 5.10: Image generated by the baseline model with (a) and without (b) noise. (c) shows the standard deviation of 100 generated images using the same latent vector, revealing which attributes are affected the most by the stochastic variation.

The difference between the images generated by the baseline model and the model trained without the stochastic variation component is hard to spot by hand, since they both create an image distribution with similar FID results. Although when observing the frequency spectrums of each image distribution by calculating the average 2D-Fourier transformation over 1000 generated images, it shows that the model trained without noise shows some spikes within the frequency spectrum (Figure 5.11 c), while the frequency spectrums of the baseline model with and without noise do not (Figure 5.11 a and b). Those spikes indicate that the generated images show repetitive patterns with the same frequencies, when trained without the component, as predicted in the StyleGAN paper.

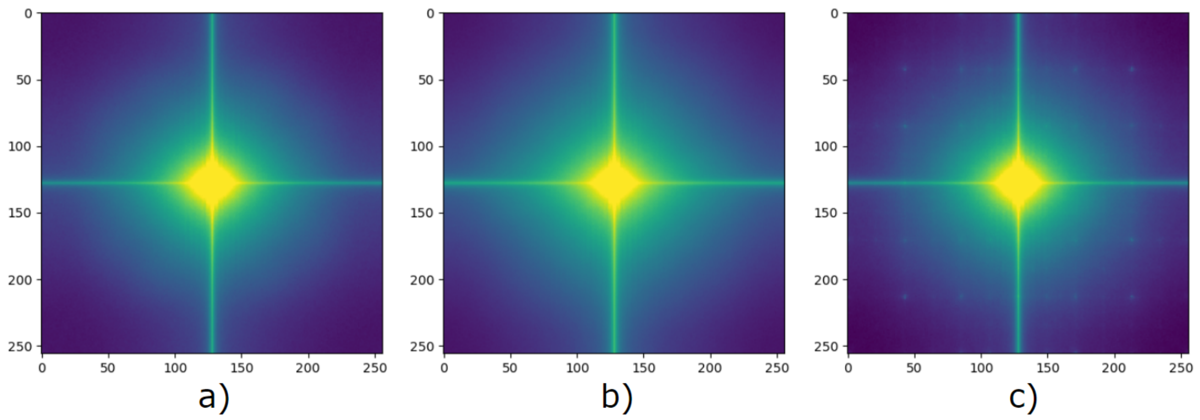

Figure 5.11: Average absolute values of the 2D-Fourier transformation for 1000 generated images from (a) the baseline model, (b) the baseline model without noise and (c) the model trained without noise, indicating repetitive patterns with similar frequencies.

5.4 Truncation Trick in W

The truncation trick, which was introduced by Andrew Brock, et. al in 2018 [BDS18], is a method to improve the generated image quality at the cost of image variation. The main idea of the truncation trick is to train the GAN with latent vectors that have been sampled from a random distribution, such as the normal distribution $\mathcal{N}(0,1)$, while using a truncated latent vector during image synthesis [BDS18].
Truncating a vector describes the transformation of a vector towards the average vector, if its distance to the average vector of the latent spaces exceeds a chosen threshold.
In traditional GANs, where the latent vectors are drawn from a normal $\mathcal{N}(0,1)$ or uniform distribution $\mathcal{U}(0,1)$, the average vector would be the zero vector. But since the StyleGAN uses a mapped latent space W, the corresponding average vector has to be approximated by a moving average that will be updated in each training iteration.
The approximated average vector is first initialized with zeros and then interpolated towards the batch average for each training iteration. The interpolation factor that was chosen by the Nvidia research group is 0.005 [KLA18c], leading to the following update operation:

$$\bar{w}_{init} = \vec{0} \tag{5.6}$$

$$\bar{w}_{new} = 0.995 \cdot \bar{w}_{old} + 0.005 \cdot \frac{1}{N} \sum_{i=1}^{N} w_i \tag{5.7}$$

where N is the batch size and w the batch of mapped latent vectors ($w = \{w_1, ..., w_N\}$).

After the training, this approximated average vector $\bar{w}$ can be used to truncate the latent vectors, used for image synthesis, by linearly interpolating in-between them. The magnitude of interpolation ψ then determines how close the latent vector will be moved towards the average vector:

$$w' = (1 - \psi) \cdot \bar{w} + \psi \cdot w \tag{5.8}$$

Figure 5.12 shows the output images of the baseline model for the same latent vectors at different ψ magnitudes. When setting $\psi = 0$ the latent vector will be replaced by the average vector, meaning that the corresponding output image shows the average face of the latent space W.
The higher the value of ψ is set, the further the distance between the latent vector and the average vector is, resulting in generated images with more variation, although slightly less quality, since the latent vectors are moved towards extreme regions of the latent space [KLA18a].

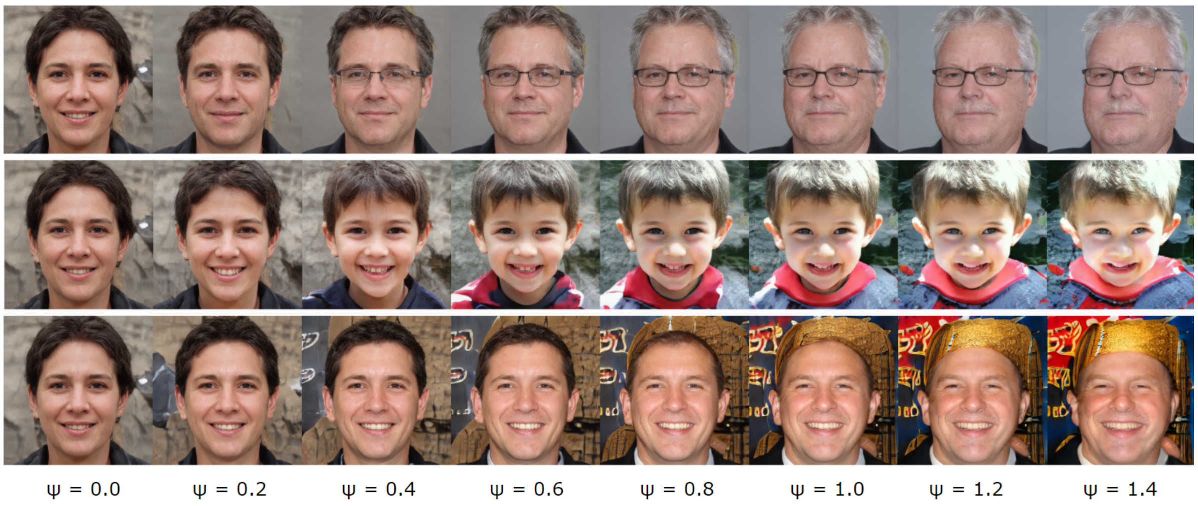

Figure 5.12: Images generated with different truncation values ψ ranging from 0.0 (showing the image of the average vector) to 1.4, where 1.0 shows the image of the latent vector without the truncation. With increasing ψ, the variation of the images is increased.

In order to find the best trade-off between image quality and image variation, Figure 5.13 plots the FIDs at different ψ values. The graph shows that the FID is lowest when $\psi = 1.0$, which means that the best FID results were achieved, when the truncation trick was deactivated. This is because the FID penalizes the image quality and image variation at the same time.

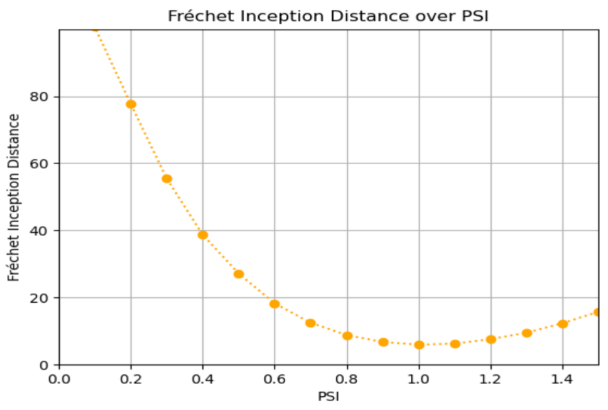

Figure 5.13: The FID over different truncation values ψ (PSI), showing that the best trade-off between quality and variation, according to the FID, is at $\psi = 1.0$, where the truncation trick is turned off.

Due to the loss in variation when using a ψ lower than 1.0, the FID scores worse even though the image quality might have improved. For that reason the truncation trick will only be applied for generating few example images with impressive quality instead of improving the scores of the metrics.

The generated images using a ψ greater than 1.0 also scored worse in FID although the variation kept increasing. This underlines that the image quality decreases for increasing ψ values and suggests that the image quality is better for lower ψ values, which was so far only claimed by examining the images from Figure 5.13 by hand.
For the example images, shown in the StyleGAN paper [KLA18a] and on the GitHub page [KLA18c], the research group Tero Karras et. al have chosen to generate the images with a ψ value of 0.7.

5.5 Mapping Network

The mapping network is a component of the generator network, introduced in the StyleGAN paper [KLA18a]. It is designed to disentangle the latent space Z in order to provide a latent space W with well separable linear subspaces each corresponding to single image attributes [KLA18a]. A disentangled latent space is desirable since it is easier for the generator to produce realistic images from a disentangled latent space than from an entangled latent space [KLA18a].
The mapping network was realized in the StyleGAN implementation by using a fully connected neural network with a depth of 8 layers as shown in Figure 5.14 a. The following experiment will remove the mapping network and use the vectors from the latent space as style inputs for AdaIN operations, as illustrated in Figure 5.14 b.

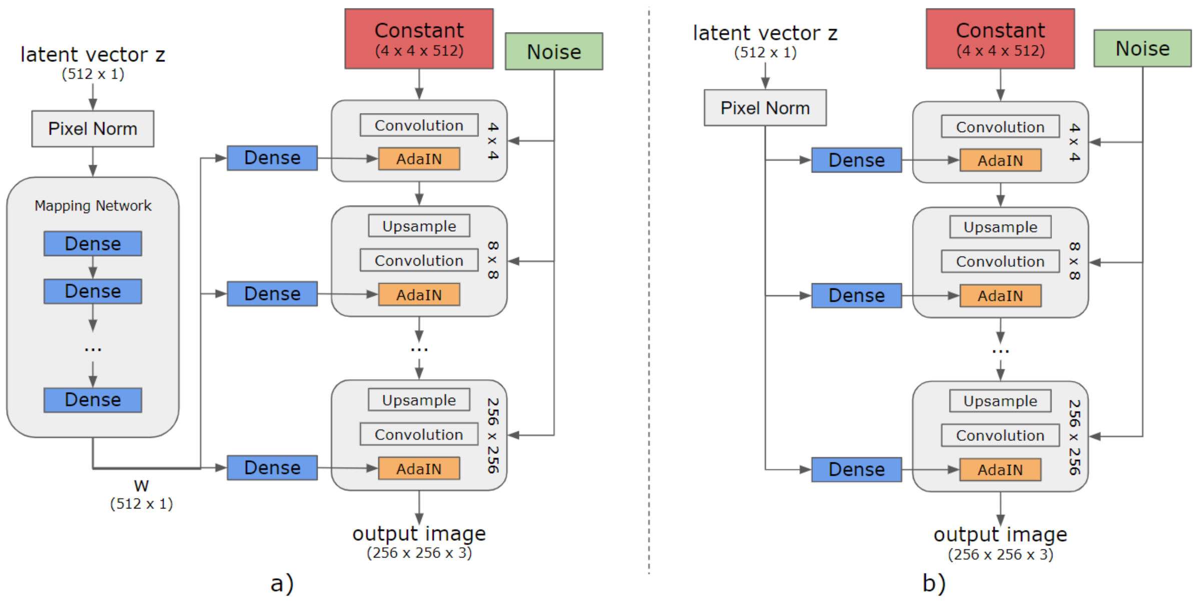

Figure 5.14: Baseline generator (a) compared to the new generator without the mapping network (b). (Inspired by [KLA18a, Figure 1])

Removing the Mapping Network

When training the StyleGAN without the mapping network, it shows in Figure 5.15 (a) that the FID increases from 6.05, scored by the baseline model, to 7.26. This margin in FID is retained throughout the whole training, indicating that by removing the mapping network the overall generated image quality or variation is decreased.

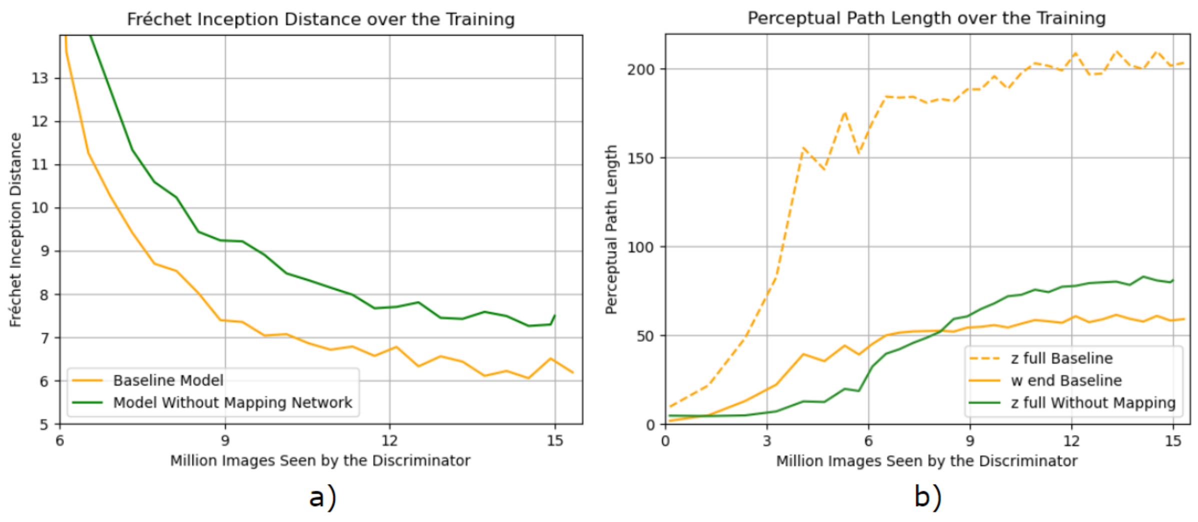

Figure 5.15: (a) FID over the training compared between the baseline model (lowest 6.05) and the model trained without the mapping network (lowest 7.26), starting at 6 million training images seen by the discriminator. (b) PPL of the baseline model compared to the model trained without the mapping network over the course of the training (lower is better for all metrics).

Inspecting the entanglement of both networks, by calculating the LS metric, shows that the trained model without the mapping network is much more entangled than the baseline model. While the baseline model had scored a LS value of 3.6 in the disentangled latent space W, the modified model scored a LS value of 8.4 in the latent space Z, which is more than twice as much.
However, when comparing the LS value of the model without the mapping network to the LS value of the baseline model in the latent space Z, it shows that the baseline model has a much more entangled latent space Z. There the baseline model scored a LS value of 169.6.

Similar results were received by the PPL metric, as shown in Figure 5.15 (b). Comparing the PPL value from the latent space W for the baseline model to the PPL for the model without the mapping network, shows that the baseline model is less entangled. Although compared to the PPL of the baseline model for the latent space Z, the latent space Z of the model without the mapping network is a lot less entangled.

Both results of the LS and the PPL metric suggest that the generator strives to disentangle the latent space even without a mapping network, although with less success than the mapping network. Doing so might also use up some of the network capacity of the generator, which also explains the higher FID.

Observing the produced images of the two models did not show any recognizable differences or artifacts.

5.6 Adaptive Instance Normalization

As described in section 5.6, the AdaIN operation is used in the StyleGAN model to transfer the 'styles', in the form of latent vectors, from the mapping network onto the synthesis network, in order to control the attributes of the output images.
For the ablation study this component can not be fully removed since it makes the only connection between the mapping network and the generator network and therefore, if removed, would also disconnect the input latent vector from the output image. For that reason the ablation will only be applied to the instance normalization of the AdaIN operation.

Removing Instance Normalization

When removing the instance normalization from the AdaIN component, only the multiplication with standard deviation of the style and the addition with the mean of the style remains. This changes the AdaIN operation from the expression:

$$\text{AdaIN}(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \tag{5.9}$$

to the new expression:

$$\text{Ada}(x, y) = \sigma(y)x + \mu(y) \tag{5.10}$$

When comparing the FID between the baseline model and the model without the instance normalization in Figure 5.16 over the course of the training, it shows that the FID of the modified model is lower at every checkpoint than the baseline model. While the baseline model scored a FID value of 6.05 at best, the modified model reached a value of 5.68, which is 0.38 lower.

An explanation for this improvement in quality, measured by the FID metric, can also be found when observing the generated images of the two models. While the baseline model had produced many images with water droplet like artifacts, as previously mentioned in section 5.1.3 and as shown in Figure 5.17 a, the model trained without the instance normalization do not show any artifacts (Figure 5.17 b).

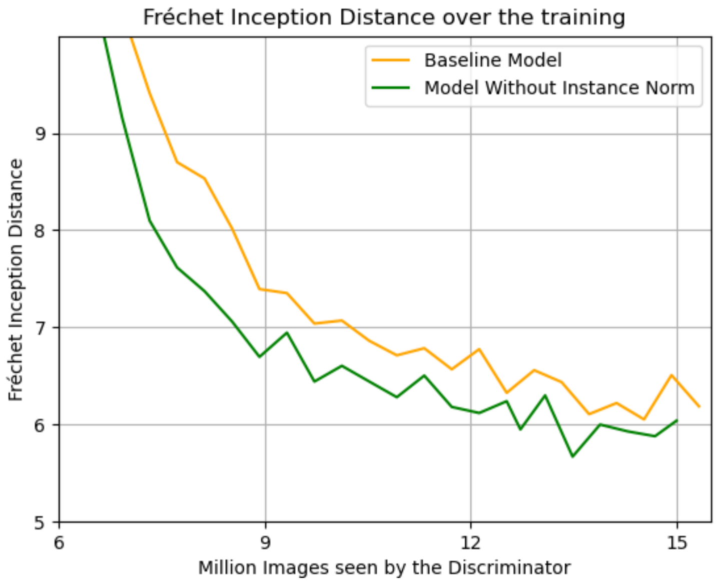

Figure 5.16: FID of the baseline model (lowest 6.05) and the model trained without instance normalization (lowest 5.68) over the course of the training, starting at 6 million images seen by the discriminator (lower is better).

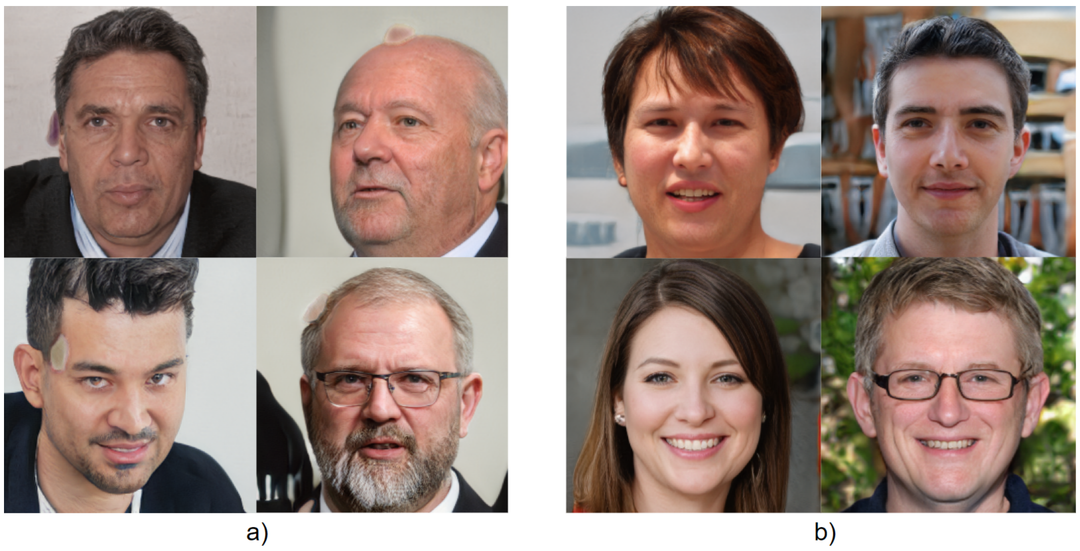

Figure 5.17: (a) images generated by the baseline model, which are showing water droplet like artifacts and (b) images generated by the model trained without instance normalization, with no water droplet like artifacts.

In the updated version of the StyleGAN [Kar+19], the Nvidia research group also addresses these water droplet like artifacts. They explain them as a side effect of the AdaIN function, where the generator tries to pass information through the AdaIN operation "by creating a strong, localized spike that dominates the statistics" [Kar+19].
This explanation can be underlined by inspecting the normalized feature maps of the generator for the images with water droplet artifacts. They show (in Figure 5.18 a) that the feature maps indeed have strong, localized spikes at the locations of the artifacts, while the images, generated by the model without instance normalization, show no spikes in the feature maps at all (Figure 5.18 b).
To resolve this issue, the research group did not remove the instance normalization, but instead replaced the whole AdaIN operation with a 'demodulation' operation that applies the styles to the kernels of the convolution instead of the resulting feature maps after the convolutions [Kar+19].

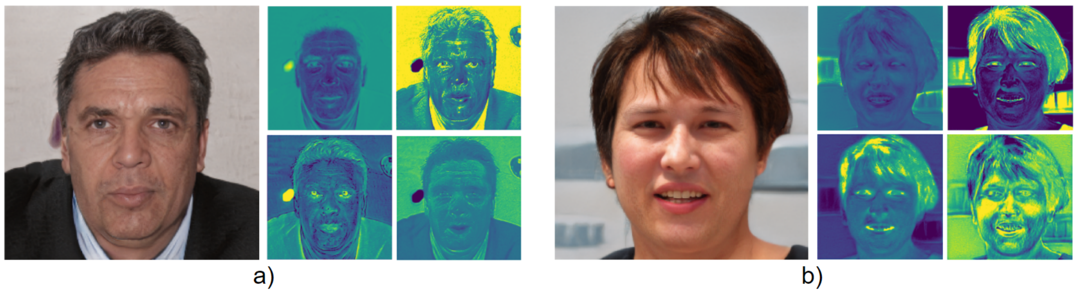

Figure 5.18: (a) Example image generated by the baseline model, which is showing a water droplet like artifact, with the corresponding normalized feature maps, showing a strong, localized spike. (b) Example image generated by the model trained without instance normalization, which show no water droplet like artifacts and no spikes within the normalized feature maps.

5.7 Minibatch Standard Deviation

Minibatch standard deviation is a technique to prevent mode collapse. It was first introduced in the paper on the progressive growing of GANs by Tero Karras et al. in 2018 [Kar+17]. Inspired was this technique from the minibatch discrimination, introduced by Salimans et al. in 2016 [Sal+16].
The concept of minibatch discrimination is to provide the discriminator with information about the batch statistics. That is done by multiplying the output of an intermediate layer in the discriminator with a trainable tensor and then comparing this product to the products of other instances from the current batch by determining the L1 distance [Sal+16]. Given this information, it is easier for the discriminator to tell whether a batch comes from the training data or from the generator if the generator produces only little variation within a batch of images. The generator is then enforced to produce batches of images with as the same amount of variation as the batches from the training dataset.

As a result, the chance of mode collapse is reduced.
The minibatch standard deviation takes the idea of the minibatch discrimination and improves it [Kar+17] while also simplifying it. Instead of using a trainable tensor and calculating the L1 distances across all batch instances, the minibatch standard deviation simply appends a new feature map that consists of the standard deviation of all feature maps from the current layer across the whole batch [Kar+17].
For the following experiment the StyleGAN will be trained without the minibatch standard deviation component.

Removing Minibatch Standard Deviation

After removing the minibatch standard deviation from the discriminator, the model scored worse in terms of FID. As shown in Figure 5.19 the FID only gets as low as 7.47, which is 1.42 higher than the baseline model.

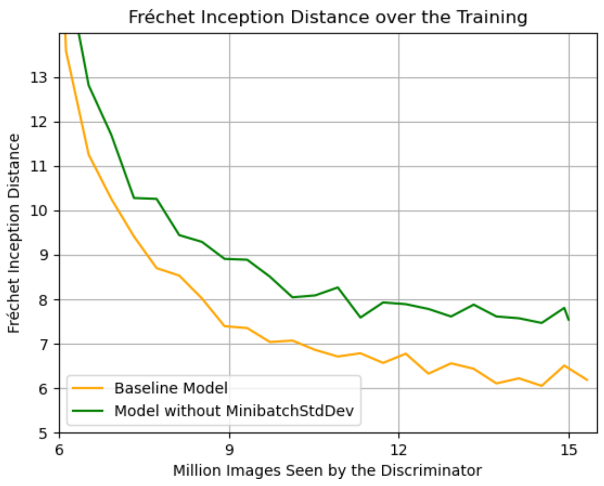

Figure 5.19: FID of the baseline model (lowest 6.05) and the model trained without minibatch standard deviation (lowest 7.47) over the course of the training, starting at 6 million images seen by the discriminator. (lower is better)

The reason for the higher FID can most likely be attributed to less variation produced by the generator. This, however, was not detectable when observing the image by hand. A hint that the variation has decreased, might be that the images showed a similar quality to the images of the baseline model, although scored a different FID. This can indicate a difference in variation, since the FID penalizes both the quality and the variation at once.

5.8 Progressive Growing

As described in 5.8 the StyleGAN uses a progressively growing structure to stabilize and speed up the training in early iterations by slowly increasing the output resolution at the beginning of the training.
With the baseline configuration, defined in 5.1, the model starts the training with an initial resolution of 8x8 pixels. This is then doubled every 1.2m training images, until the full resolution of 256x256 is reached. With the first 600k training images after each new layer was added, the output of the generator is calculated by interpolating between the output of the new layer and the upsampled output from the earlier layer. Doing so provides smooth transitions between each resolution step.
Figure 5.20 shows the output images of the baseline generator for the same latent vector over the course of the beginning of the training.
For the following experiment the StyleGAN model will be trained without the progressively growing structure, by setting the initial output resolution to the full resolution of 256x256 pixels.

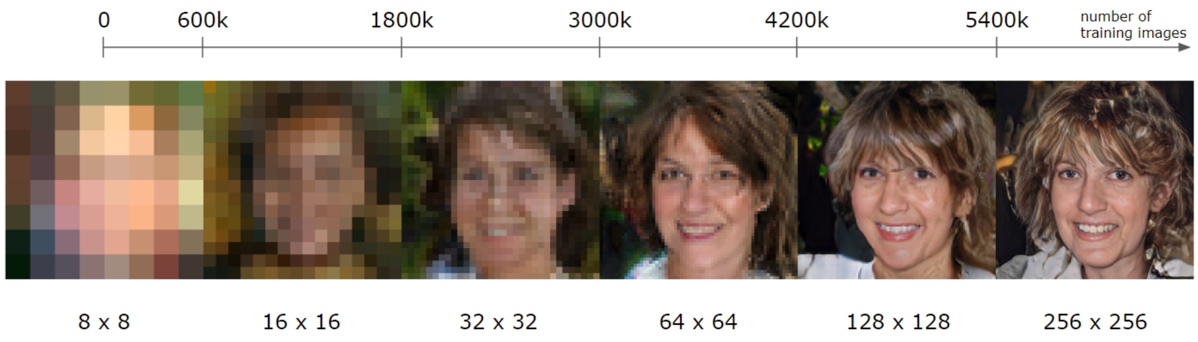

Figure 5.20: Increasing image resolution of the baseline model. After 5.4 million training images, the full resolution of 256x256 is reached.

Removing the Progressively Growing Structure

When removing the progressively growing structure from the StyleGAN model, it shows in Figure 5.21 that the FID converges much faster than it did with the baseline model, considering the number of training iterations. On the one hand, this is because the FID reaches very large values when evaluating images that have different resolutions and on the other hand this is because the discriminator receives much more information when training with images of higher resolution.
After about 10 million training images, both models have reached very similar FIDs with slightly lower FID from the baseline model than from the modified model. At best the modified model scored a FID of 6.35 which is 0.29 higher than the 6.05 scored by the baseline model.
The aforementioned speedup that comes with the progressively growing structure amounts

a total difference of 6 hours. While the model without progressive growing trained for 8 days and 6 hours, the model with progressive growing only trained for 8 days.

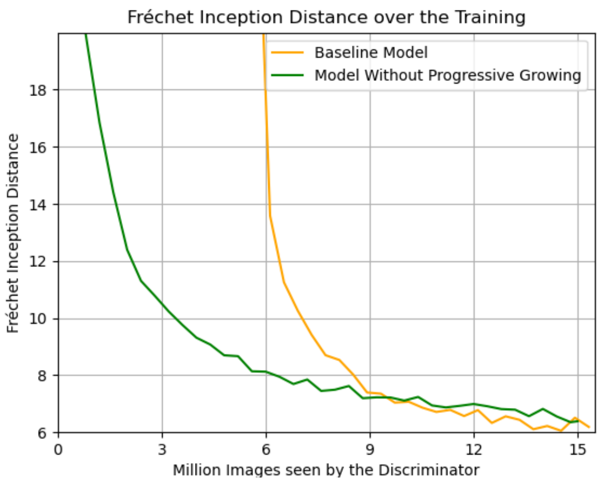

Figure 5.21: FID of the baseline model (lowest 6.05) and the model trained without progressive growing (lowest 6.35) over the course of the training. (lower is better)

Observing the output of the generator from the model without progressive growing at the beginning of the training in Figure 5.22, shows that the generated images first look very similar to each other. This indicates that the model tends to mode collapse at the beginning. However after 120k training images, the model recovers from mode collapse and produces images with more variation.
When inspecting the images from the fully trained model at the checkpoint with the lowest FID, most images look very realistic with similar quality to the images from the baseline model.

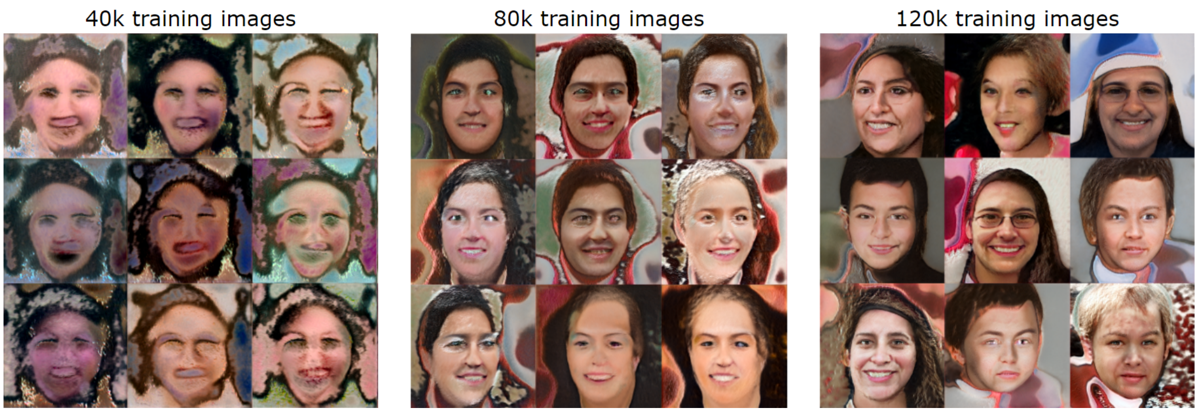

Figure 5.22: Generator output of the model trained without the progressively growing structure at the beginning of the training. At 40k and 80k training images, the model is showing some mode collapse, which however disappears after 120k training images.

A reason why the model performs well even without the progressively growing structure, might be that the full resolution of 256x256 is much lower compared to the full resolution of 1024x1024 that was used in the paper. When training the model with an output resolution of 1024x1024 without the progressively growing structure, the model would probably fully mode collapse at the beginning of the training, with a low chance of recovering.

Although the generated images of the baseline model and the modified model have very similar quality, one difference can be seen by examining the location of attributes when interpolating in-between two images with different face poses. While in the baseline model attributes such as teeth or eyes seem to be stuck in place when interpolating in-between different poses, the attributes in the model without the progressive growing seem to be moving freely around. For example when inspecting the location of the teeth in the baseline model in Figure 5.23 a, it shows that even though the pose of the face changes after the interpolation, the teeth remain in the same position, which results in images with teeth that are unaligned to the rest of the face.
Figure 5.23 b shows that in the images of the modified model, the teeth move accordingly to the rest of the face, making it look more realistic.

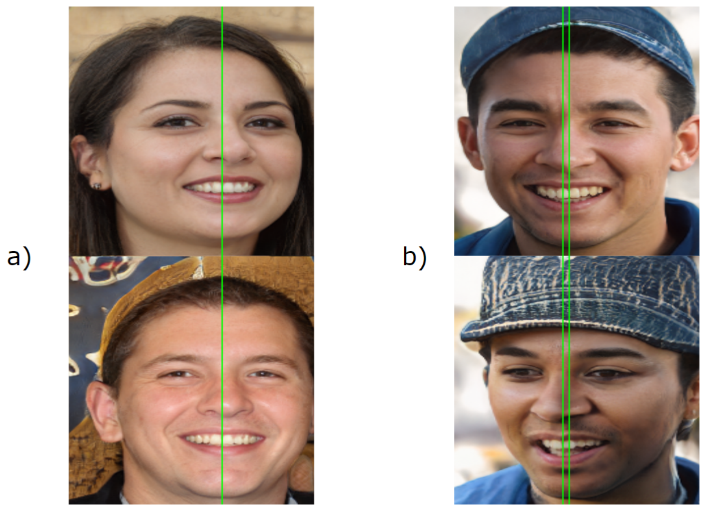

Figure 5.23: Demonstration of the 'phase artifacts'. (a) shows images generated by the baseline model and (b) images generated by the model without progressive growing when interpolating in W. The green lines are placed in-between the incisors to highlight that the teeth are at the same position in (a), which makes the teeth unaligned to the rest of the face in the upper image, while the teeth move accordingly to the pose of the face in (b).

These artifacts were also found by the Nvidia research group Tero Karras et. al [Kar+19]. They call these artifacts the 'phase artifacts' and explain them as a side effect of the progressively growing structure. The research group hypothesizes that since each layer has to perform as an output layer at the beginning of the training, the feature maps have higher frequencies, because the generator is enforced to produce images with high frequency details. The high frequencies are then causing the generator to be less shift invariant [Kar+19]. To resolve this issue, the research group has removed the progressively growing structure and replaced it with a residual architecture in the StyleGAN2 paper [Kar+19].

5.9 Exponential Moving Average of the Generator Weights

One major problem that often occurs when training GANs is the instability of the model parameters. A common solution to that problem is to apply a gradient penalty by modifying the loss functions as described in section 5.1.1.
Next to the gradient penalty the StyleGAN also uses the 'Exponential Moving Average of the Generator Weights' (EMA) component to stabilize the parameters, which was introduced by Yasin Yazıcı [Yaz+18] in 2018.
Unlike the gradient penalty, the EMA does not have any influence on the training itself and instead only copies the weights from generator (G_{train}) onto a cloned generator ($G_{synthesis}$), which will only be used for image synthesis after the training. This synthesis generator receives the current weights at every training iteration from the training generator and averages it by interpolating between the new and the old weights as described by the following expression [KLA18c]:

$$G_{synthesis} = \beta \cdot G_{synthesis} + (1 - \beta) \cdot G_{train} \tag{5.11}$$

where β is a hyperparameter that controls the magnitude of how much the weights are changed in each update. In the StyleGAN implementation β is set to 0.9997, which means that in each training iteration the weights of the synthesis generator are interpolated 0.03% towards the weights of the training generator.

Removing the Exponential Moving Average of the Generator Weights

Since the EMA component does not affect the training itself, no new model has to be trained. Instead, only the generator that was used in training G_{train} has to be observed. This makes it a lot easier to detect the difference with and without the component, by generating images from the same random input vector for both generators.

Figure 5.24 shows the training progress of G_{train} and $G_{synthesis}$ for the same input vector, demonstrating that the generated images of G_{train} change a lot during the training while $G_{synthesis}$ shows very smooth transitions between the training checkpoints.

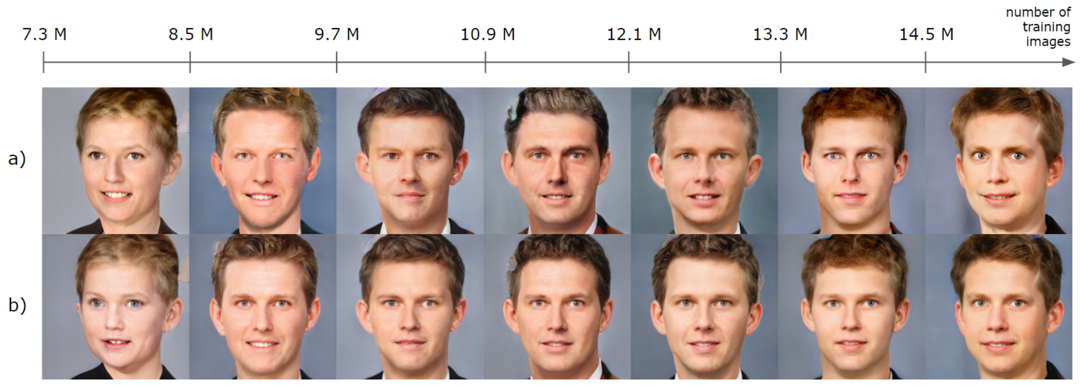

Figure 5.24: Training progress of the baseline model without EMA (a) and with EMA (b).

When comparing the FID of both generators over the training in Figure 5.25, it shows that the FID is a lot lower for $G_{synthesis}$, with a lowest value of 6.05, than G_{train}, with the lowest value of 8.22. According to that result, the EMA component had the most influence on the StyleGAN throughout the whole ablation study.

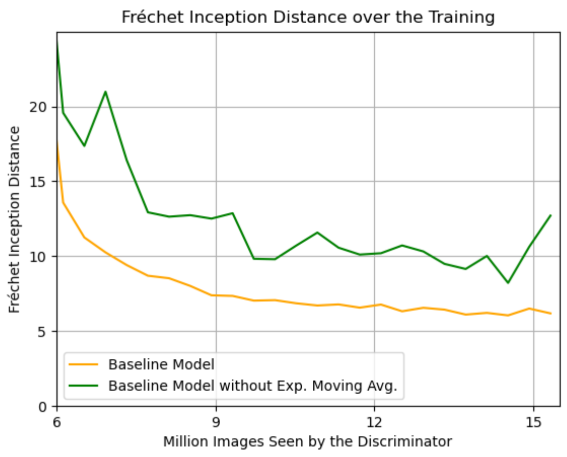

Figure 5.25: FID of $G_{synthesis}$ (lowest 6.05) and G_{train} (lowest 8.22) over the course of the training, starting at 6 million training images (lower is better).

5.10 Summary

The results of the experiments were mostly as anticipated. As soon as a component was removed, the FID increased and the image quality decreased. This was the case with the stochastic variation, the style mixing, the mapping network, the minibatch standard deviation and the EMA component.
The highest change in FID was recorded after removing the EMA component in section 5.9, which increased the FID from 6.05 to 8.22.
Although most experiments scored worse results than the baseline model, removing the AdaIN component and the stochastic variation from the baseline model actually improved the FID. The overall best result was achieved, with a FID of 5.68, after removing the instance normalization from the AdaIN component. The most likely reason for this low FID is that the water droplet artifacts disappeared, which were visible in almost every image, generated by the baseline model.
Another artifact that was observed in section 5.8 is the 'phase artifact'. Although this artifact disappeared after removing the progressively growing structure, which made the images look more realistically, the FID was still higher than the baseline model. This concludes that progressive growing component both improves and harms the quality at the same time.

A summary of all FID results from all experiments in the ablation study is shown in Table 5.2.

Model	FID	FID Difference to Baseline
Baseline Model	6.05	
Model without AdaIN	**5.68**	-0.37
Baseline Model without Stochastic Variation	6.04	-0.01
Model without Stochastic Variation	6.19	+0.14
Model without Progressive Growing	6.35	+0.30
Model without Style Mixing	6.49	+0.44
Model without Mapping Network	7.26	+1.21
Model without Minibatch Std. Dev.	7.47	+1.42
Baseline Model without Exp. Mov. Avg.	8.22	+2.17

Table 5.2: FID of all experiments in the ablation study sorted by FID.

6 Results on the Car Dataset

The following chapter will analyze the StyleGAN by training it with a different dataset for more iterations and a higher output resolution, while applying some of the analysis methods that were used in the ablation study. The therefore used dataset is the car dataset, which was created by the chair for Multimedia Computing and Computer Vision of the Augsburg University.

6.1 Car Dataset

The car dataset consists of 30 000 images of cars at resolutions ranging from 75 x 75 up to 8688 x 5792. The cars themselves can be separated into 18 car manufacturers, most of which are German car manufacturers such as Porsche, Opel or BMW.
Further information about the manufacturer and car color distributions of the car dataset are shown in Figure 6.2.

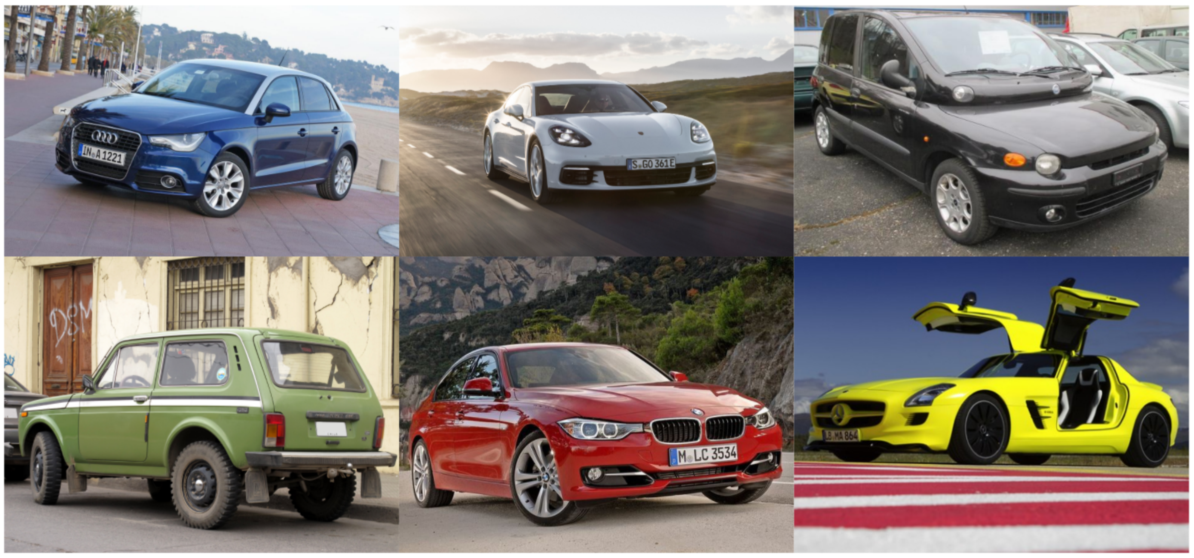

Figure 6.1: Example images from the car dataset.

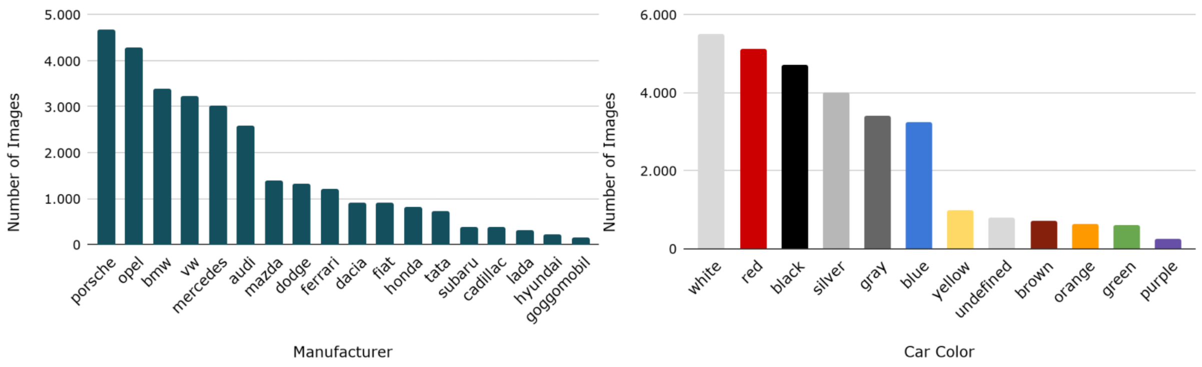

Figure 6.2: Histogram of manufacturers and colors in the car dataset.

In order to train the StyleGAN model with the car dataset, all images need to have the same size and a quadratic aspect ratio. For that reason, all the images with vertical format were filtered, which removed 0.93%. The remaining images were then resized to a resolution of 512 x 320, which is the resolution with the closest aspect ratio to the average aspect ratio of all images and can be divided by two six times. This is necessary for the progressively growing network structure in order to start the training with a resolution with an even number of pixels. After that the images are padded with zeros to a squared resolution of 512 x 512. An example for the padded training images at different resolution can be seen in Figure 6.3.
Like with the FFHQ dataset, the images from the car dataset were also augmented by flipping them along the vertical axis.

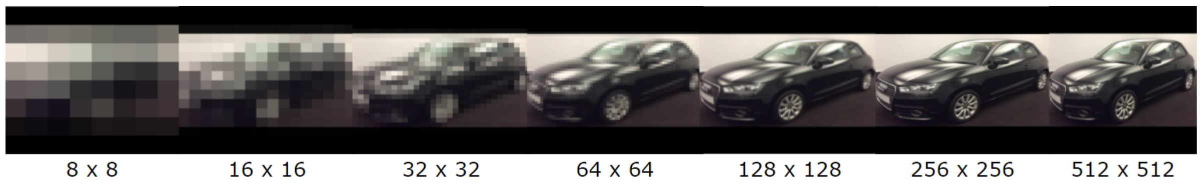

Figure 6.3: Example training image after padding with zeros and resizing.

6.2 Training Results

The StyleGAN model was trained with the default configuration, taken from the StyleGAN implementation [KLA18c], for 25 million training images at an output resolution of 512 x 512. The whole training took about 16 days and 17 hours, using two GTX 1080 Ti's.

Fréchet Inception Distance

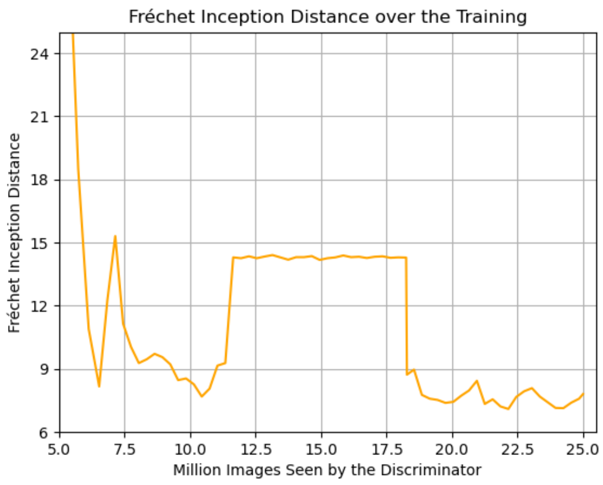

Figure 6.4: FID over the training with the car dataset (lowest 7.11), starting at 5 million images seen by the discriminator (lower is better).

The FID graph, shown in Figure 6.4, displays the FID over the course of the training. At the beginning of the training the FID decreases very quickly due to the difference in output resolution, until the full resolution is reached after 6.6 million training images. From there on the FID slowly decreases as low as 7.70 at 10 million training images. After 11 million training images the FID increases again and stays at about 14.2 for the following 7 million training images. The FID then suddenly drops again after 18 million images and from there on decreases further to the overall lowest value of 7.11.

A reason why the FID got stuck at 14.2 after 11 million training images might be that the model reached a local Nash equilibrium. This hypothesis is supported when inspecting the generated images. They shows that the images with the same input vector do not change a lot in-between the training checkpoints. This hints to low gradients and the convergence to a local Nash equilibrium.
While the networks usually struggle to recover after converging to a local Nash equilibrium with diminishing gradients, this network suddenly did recover after 18 million training images. The simple reason for this is that the training crashed at 18 million training images, due to insufficient memory on the GPU server in order to store the

models checkpoint. Then after continuing the training from the latest checkpoint, the optimizer was reset, which enabled the network to leave the local Nash equilibrium.

Generated Images

Observing the output of the StyleGAN at the checkpoint with the lowest FID of 7.11, shows that generator produces realistic looking cars with a lot of detail and variation in car model, car color and background setting (Figure 6.5).
Next to the good images, some images also showed very unrealistic car shapes or unnatural background patterns (Figure 6.6).
Also the water droplet like artifacts, that were discussed in section 5.6, can be found in many generated images.

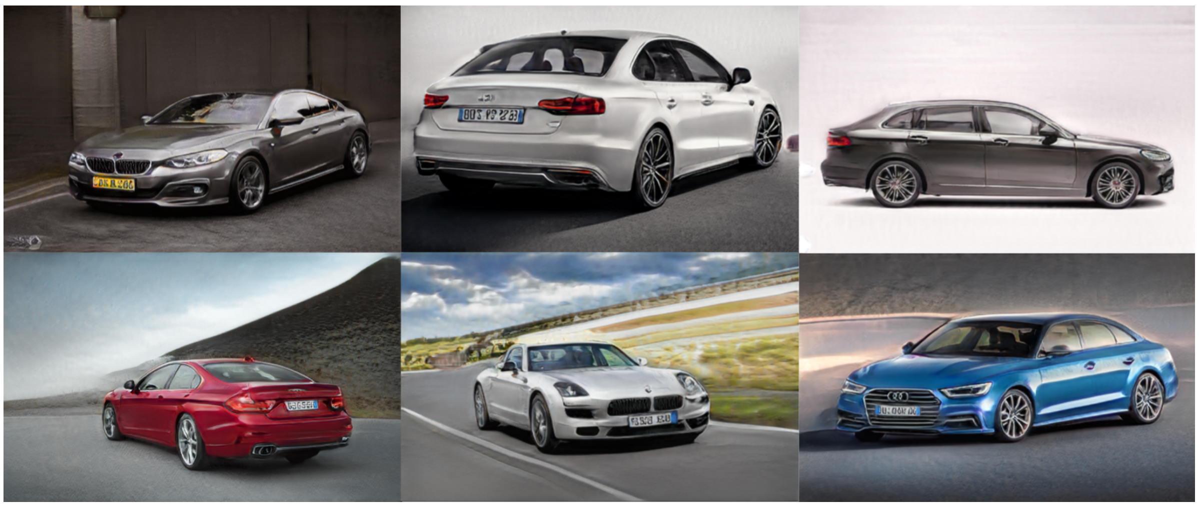

Figure 6.5: Example images generated by the StyleGAN with high quality.

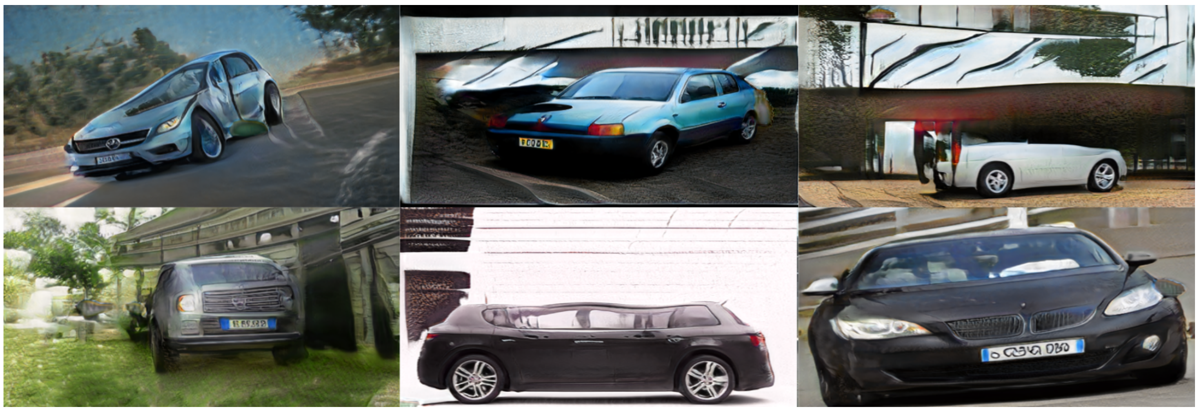

Figure 6.6: Example images generated by the StyleGAN with bad quality.

Average Image

Applying the truncation trick in W with a ψ value of 0.0, generates the approximated average image of the StyleGAN. This average car, shown in Figure 6.7, has a grey color and shows similarities to a Mercedes A-Class. This color matches well with the color distribution of the dataset (Figure 6.2), since the colors white, black, silver and grey are part of the five most common car colors. The similarities to the Mercedes A-Class could hint to some mode collapse, considering that Mercedes is only the fifth most common car manufacturer of the dataset.

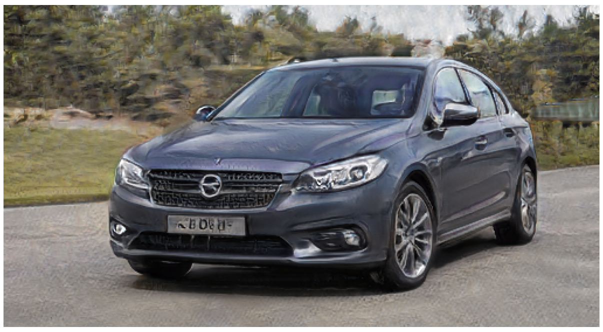

Figure 6.7: The average car image. Created by generating the image using the average latent vector in W.

Number Plates

Observing the number plates of the cars in the generated images, in Figure 6.8, shows that most letters or numbers are either an eight or a zero. This indicates that the generator mode collapsed for this specific factor of variation.
The same effect can also be observed in some of the results from the StyleGAN2 paper [Kar+19, Figure 14].

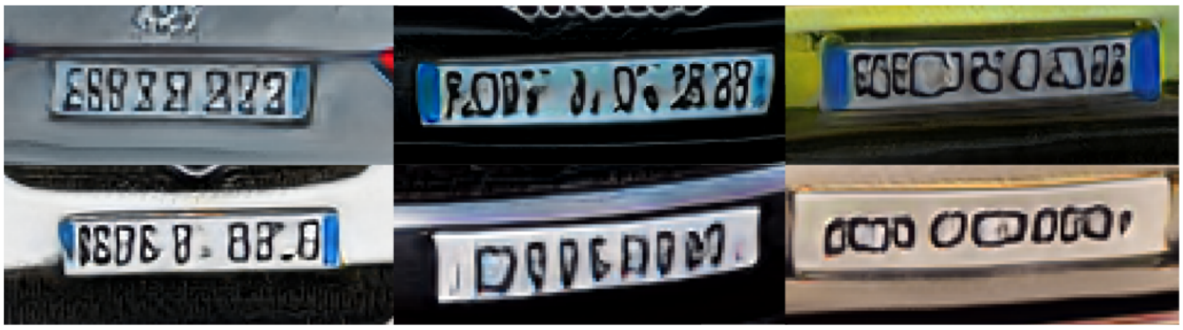

Figure 6.8: Example images of number plates, showing that most of the generated numbers are zeros or eights.

Stochastic Variation

To analyze the influence of the stochastic variation component for the car images, Figure 6.9 shows the standard deviation for 100 images that were generated using the same random input vector. The brighter an area is, the more it got changed by the added noise from the generator.
The standard deviation image indicates that mostly the number plates, the wheels, the edges of the car and the car logos are affected by the stochastic variation.

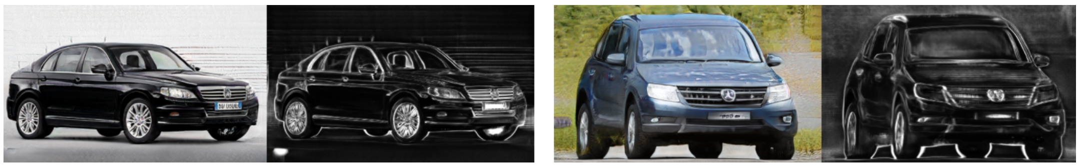

Figure 6.9: Standard deviation of 100 images with the same seed, showing that the random noise mostly affect the edges, the number plates, the wheels and the car logos.

Style Mixing

Like in section 5.2, Figure 6.10 shows the output image when replacing the style input at the AdaIN operations with a different style for one resolution in the generator. Similar to the results from section 5.2, the first layer with a resolution of 4 x 4 changes the orientation of the car, the middle layers with resolutions form 8 x 8 to 64 x 64 change the car shape and the car color and the last layers with resolutions form 128 x 128 to 512 x 512 change small details and the overall brightness of the image.

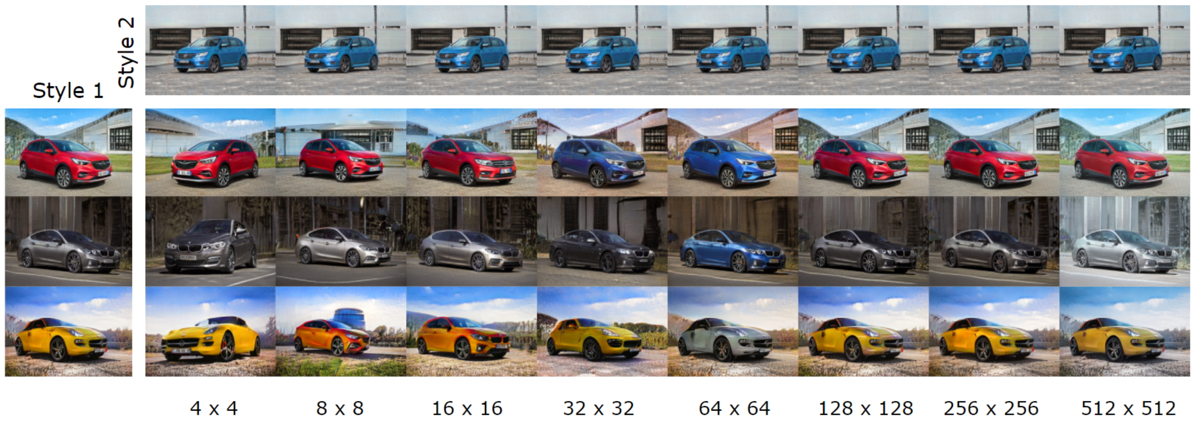

Figure 6.10: Generator output when exchanging the latent vector of style 1 with the latent vector of style 2 for one resolution block. Earlier layers of the generator (4x4 - 16x16) affect the coarse attributes such as car orientation or shape, while later layers (32x32 - 512x512) affect small details and colors.

7 Conclusion & Outlook

After training the StyleGAN with 9 different configurations for a total computing time of more than 79 days, it showed that the StyleGAN is very robust to modifications and does not rely on single components to perform well. This was especially observed after removing important components such as the progressive growing, where the images after the experiment still had a good quality and the FID was relatively low. The most likely reason for that might be the lower output resolution, used throughout all experiments of the ablation study. While the StyleGAN was originally designed to generate images at a resolution of 1024x1024, the experiments were only made on a network with an output resolution of 256x256, which simplifies and therefore stabilizes the training.
Also, in some of the experiments, for example in the experiment for the stochastic variation component, the FID graph of the baseline model and the modified model showed very similar results. In order to tell which one of the models performed better, the experiment should be run multiple times or trained for more training iterations. This although was not possible without compromising other experiments.
The biggest increase in FID was observed after removing the exponential moving average and the the biggest decrease after removing the AdaIN component. For those two components it would also be interesting to perform multiple experiments with only slight modifications, instead of only turning it on or off. The AdaIN component could then, for instance, be analyzed further by replacing the instance normalization with a pixel or batch normalization. Furthermore the exponential moving average component could be examined by selecting different values for the interpolation magnitude β.
Also, training the model leaving out multiple components at once could show interesting results.

Although the training on the car dataset was corrupted by resetting the optimizer after the training crashed, it still showed that the StyleGAN model also performs well on datasets with fewer images that were not as carefully preprocessed, as in the FFHQ dataset.
While the stochastic variation component was specifically designed to generate random features of human portraits, it turns out that the generator also found use for the component when training with the car dataset. Instead of freckles and hair placements the network applied the noise to generate variation in number plates or car logos.
Further interesting studies on the car dataset can be made by training the StyleGAN with labels, since the dataset is annotated with the car model, the car manufacturer and the car color. This would allow the synthesis of images with specific requests.

In conclusion, the ablation study showed that, on the one hand, the StyleGAN model is well equipped with components that help the model to generate images of high quality, whereas on the other hand some components create artifacts and decrease that image quality. In the StyleGAN2 paper [Kar+19], the Nvidia researcher group resolved those artifacts by rebuilding the model's structure and replacing some of the components, which led to significant improvements in FID.
Recent research, however, such as the paper from Joel Frank et al. [Fra+20], has already found new artifacts in the StyleGAN images. This suggest that the model still has a lot of potential and the generated images will become even more realistic in future.

List of Figures

Bibliography

[Ach+18] Dinesh Acharya et al. *Towards High Resolution Video Generation with Progressive Growing of Sliced Wasserstein GANs.* 2018. arXiv: `1810.02419 [cs.CV]`.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis.* 2018. arXiv: `1809.11096 [cs.LG]`.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing).* USA: Wiley-Interscience, 2006. ISBN: 0471241954.

[CV95] Corinna Cortes and Vladimir Vapnik. *Machine Learning.* Kluwer Academic Publishers, 1995.

[Fra+20] Joel Frank et al. *Leveraging Frequency Analysis for Deep Fake Image Recognition.* 2020. arXiv: `2003.08685 [cs.CV]`.

[Gho+18] Arnab Ghosh et al. *Multi-Agent Diverse Generative Adversarial Networks.* 2018. arXiv: `1704.02906 [cs.CV]`.

[Goo+14] Ian J. Goodfellow et al. *Generative Adversarial Networks.* 2014. arXiv: `1406.2661 [stat.ML]`.

[Goo16] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks.* 2016. arXiv: `1701.00160 [cs.LG]`.

[Gui+20] Jie Gui et al. *A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications.* 2020. arXiv: `2001.06937 [cs.LG]`.

[HB17] Xun Huang and Serge Belongie. *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization.* 2017. arXiv: `1703.06868 [cs.CV]`.

[Heu+17] Martin Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium.* 2017. arXiv: `1706.08500 [cs.LG]`.

[Kar+17] Tero Karras et al. *Progressive Growing of GANs for Improved Quality, Stability, and Variation.* 2017. arXiv: `1710.10196 [cs.NE]`.

[Kar+19] Tero Karras et al. *Analyzing and Improving the Image Quality of StyleGAN.* 2019. arXiv: `1912.04958 [cs.CV]`.

[Kha+20] Nour Eldeen M. Khalifa et al. *Detection of Coronavirus (COVID-19) Associated Pneumonia based on Generative Adversarial Networks and a Fine-Tuned Deep Transfer Learning Model using Chest X-ray Dataset.* 2020. arXiv: `2004.01184 [eess.IV]`.

[KLA18a] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks.* 2018. arXiv: `1812.04948 [cs.NE]`.

[KLA18b] Tero Karras, Samuli Laine, and Timo Aila. *Flickr-Faces-HQ Dataset (FFHQ).* 2018. URL: `https://github.com/NVlabs/ffhq-dataset` (visited on 06/10/2020).

[KLA18c] Tero Karras, Samuli Laine, and Timo Aila. *StyleGAN — Official TensorFlow Implementation.* 2018. URL: `https://github.com/NVlabs/stylegan` (visited on 06/10/2020).

[Led+16] Christian Ledig et al. *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.* 2016. arXiv: `1609.04802 [cs.CV]`.

[LKC16] William Lotter, Gabriel Kreiman, and David Cox. *Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning.* 2016. arXiv: `1605.08104 [cs.LG]`.

[Mao+16] Xudong Mao et al. *Least Squares Generative Adversarial Networks.* 2016. arXiv: `1611.04076 [cs.CV]`.

[Met+16] Luke Metz et al. *Unrolled Generative Adversarial Networks.* 2016. arXiv: `1611.02163 [cs.LG]`.

[Mey+19] Richard Meyes et al. *Ablation Studies in Artificial Neural Networks.* 2019. arXiv: `1901.08644 [cs.NE]`.

[MGN18] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. *Which Training Methods for GANs do actually Converge?* 2018. arXiv: `1801.04406 [cs.LG]`.

[Sal+16] Tim Salimans et al. *Improved Techniques for Training GANs.* 2016. arXiv: `1606.03498 [cs.LG]`.

[Sho85] Ken Shoemake. “Animating rotation with quaternion curves”. In: *SIGGRAPH Comput. Graph.* 19.3 (July 1985), pp. 245–254. ISSN: 0097-8930. DOI: `10.1145/325165.325242`. URL: `http://doi.acm.org/10.1145/325165.325242`.

[Sil18] Thalles Silva. *An intuitive introduction to Generative Adversarial Networks (GANs).* 2018. URL: `https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/` (visited on 06/15/2020).

[SZ14] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* 2014. arXiv: `1409.1556 [cs.CV]`.

[Sze+15] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: `1512.00567 [cs.CV]`.

[Wan+18] Ting-Chun Wang et al. *High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs*. 2018. arXiv: `1711.11585 [cs.CV]`.

[Whi16] Tom White. *Sampling Generative Networks*. 2016. arXiv: `1609.04468 [cs.NE]`.

[Yaz+18] Yasin Yazıcı et al. *The Unusual Effectiveness of Averaging in GAN Training*. 2018. arXiv: `1806.04498 [stat.ML]`.

[Yeh+16] Raymond A. Yeh et al. *Semantic Image Inpainting with Deep Generative Models*. 2016. arXiv: `1607.07539 [cs.CV]`.

[Zhu+17] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2017. arXiv: `1703.10593 [cs.CV]`.