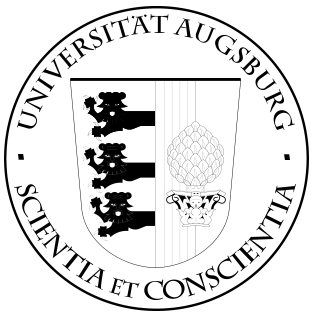
Institut für Informatik
Universität Augsburg



Master Thesis

Conditional Rendering of 3D Objects from 2D Images using Deep Generative Neural Networks

Florian Barthel

Gutachter:	Prof. Dr. Rainer Lienhart
Zweitgutachter:	Prof. Dr. Elisabeth André
Betreuer:	Stephan Brehm
Datum:	March 21, 2022

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

Abstract

In this thesis, we will present a method that allows to conditionally synthesize photorealistic 3D-like 360° views of cars using multi-labeled data. This is done by proposing a multiple modifications to the StyleGAN introduced by Karras et al. in 2019 [Kar+19]. For this purpose, we will first focus on implementing and improving conditional synthesis with the StyleGAN. To this end, we will propose several quality metrics specifically for multi-labeled conditional synthesis and perform a series of experiments with the proposed changes. As a result, we show that our modifications help the model to improve the image quality, especially for label combinations that do not exist in the dataset. This allows to configure the appearance of cars in photorealistic images with about 70 Million different label combinations. Given this capability, we then use a discrete camera location label to synthesize continuous 360° rotations those cars. For this, we perform a series of experiments and compare them along several rotation quality measurements. We then obtain a model that enables to configure any car and rotate it by any given angle. In a last experiment, we also show that the proposed methods scale well with higher resolutions. This was often a drawback of GAN methods from literature that produce images with 3D properties.

Contents

1 Introduction

The goal of synthesizing photorealistic images with machine learning algorithms has become increasingly popular in recent years. It was mainly initiated with the introduction of generative adversarial networks (GANs) by Goodfellow et al. in 2014 [Goo+14]. Since then, significant advancements have been made, which now allow the synthesis of images that are almost indistinguishable from real images for the first time. Due to this capability, GANs have found many applications in photo and video editing [ZKS21], product design [Kat+19], 3D modeling [Cha+21b; Cha+21a; NG21], medicine [Wah+20], and various other fields.

One of the most popular GAN architectures that achieves such photorealistic image synthesis is the StyleGAN introduced by Karras, Laine, and Aila in 2018 [KLA18a]. This architecture improves the image synthesis considerably and is currently considered as one of the best performing GAN frameworks for various image domains [Styb]. It especially stands out for its ability to synthesize images at high resolutions, compared to other GANs.

Although the synthesized images of GANs look impressively realistic, it is still a very challenging task to precisely control specific features of the output images. To tackle this challenge, various methods have been proposed in literature. One of them is the Conditional GAN (cGAN), which was introduced by Mirza and Osindero [MO14] shortly after Goodfellows introduction of GANs. In their paper, they propose a modified GAN architecture that enables conditional image synthesis using labeled training data. For the first time, this modification made it possible to achieve a significantly higher level of control over the output images from GANs.

In this thesis, we will combine the realistic image synthesis of the StyleGAN with the controllability of the cGAN. Furthermore, we will use a multi-label dataset of car images, created by the Chair for Machine Learning & Computer Vision of the University of Augsburg, to manually configure the synthesis of photorealistic car images. To do so, we will propose multiple modifications to the StyleGAN to improve the conditional synthesis even further. In addition to that, we add a camera location annotation to the car dataset, which allows generating car images from eight discrete viewpoints. With this label, we then aim to improve the synthesized images in between those eight discrete viewpoints, with the goal of creating a 360° view of a car. At the end of this thesis, we combine all successful components in one model and test how well the proposed modifications scale with an increased image resolution. The resulting model enables to manually configure the attributes of high resolution photorealistic 360° views of cars.

This thesis is structure as follows:

- First, in chapter 2, a brief overview of Generative Adversarial Networks in general, the StyleGAN model and Conditional GANs is given. Furthermore, the the Fréchet Inception Distance is presented, which measures image quality for GANs.
- Then, in chapter 3, we will present the dataset that will be used for the conditional image synthesis.
- After that, in chapter 4, we will demonstrate several approaches from literature to control the image synthesis of GANs using a StyleGAN model that was trained with the dataset from chapter 3.
- In chapter 5, we will implement the conditional properties for the StyleGAN training and propose multiple modifications to improve the conditional image synthesis even further.
- In chapter 6, we will give an overview of related work from literature that uses GANs to synthesize images with 3D representations of objects.
- In chapter 7, we will create and improve the 360° synthesis of cars with the conditional StyleGAN from chapter 5
- And at last in chapter 8, we will train a model at a higher image resolution in which we combine all successful modifications from chapter 5 and chapter 7 in one model.

2 Related Work

2.1 Generative Adversarial Networks

In the past years, Generative Adversarial Networks (GANs) have become one of the most important machine learning methods in the field of image synthesis. Since the introduction of GANs by Goodfellow et al. [Goo+14], they have been used in a wide range of applications, making it, according to Yann LeCun, one of the most interesting technologies in the past decade [Roc19].
The main idea of a GAN is to simultaneously train two separate deep neural networks with adversarial behavior. One of the networks, the generator, is trained to synthesize fake images from random latent vectors z (noise vectors), whereas the other network, the discriminator, is trained to classify whether a given image is real or was created by the generator. Since the generator never observes the real images during the training, it exclusively relies on the feedback of the discriminator to improve the quality of the generated images. At the beginning of the training, both networks perform very poorly, meaning that the generator produces random images and the discriminator misclassifies real and fake images. However, as both networks improve over the course of the training, the generator starts to produce images that look more similar to the real images provided to the discriminator. This is done, until, ideally, the generated images look as realistic as the training images. An overview of a basic GAN training is visualized in Figure 2.1.

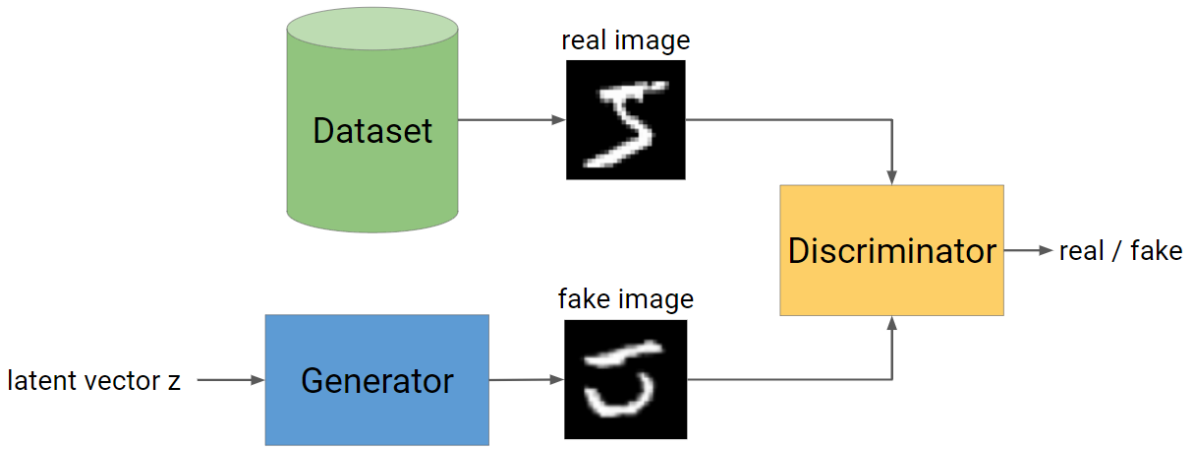


Figure 2.1: An overview of a basic GAN architecture.

This adversarial behavior can be formalized in a non-cooperative min-max game, where the discriminator aims to maximize the probability of correctly distinguishing the fake images from the real images, while the generator aims to minimize the probability that the discriminator detects the fake images. This min-max game can be expressed with the following value function [Goo+14]:

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

Here, $D(x) \in (0,1]$ denotes the prediction of the discriminator, whether a given image is real or fake. If $D(x)$ outputs 0, it predicts that the image is fake and if it outputs 1, it predicts that the image is real. $G(z)$ describes the output of the generator that uses a random latent vector z to synthesize fake images. To minimize the output of the value function, the generator maximizes $D(G(z))$, which increases if the discriminator classifies a fake image as a real image. The discriminator, on the other hand, maximized this value function by correctly identifying the real images, which increases the first term $\mathbb{E}_{x\sim p_{data}(x)}[logD(x)]$ and also by detecting the fake images, which increases the second term $\mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$. This min-max game is played until a Nash equilibrium is reached. A Nash equlibrium describes a state of a non-cooperative game, where both players can not improve their strategy, irrespectively of the actions of the opponent. Ideally, this state is reached when the image distribution of the generator p_g is equal to the image distribution of the training data p_{data} [Goo+14, Theorem 1]. In practice, however, we will only reach a local Nash equilibrium, since the gradient descent algorithm is a local optimization method [Heu+17]. Two examples of a successful GAN training are shown in Figure 2.2. Those models have been trained on the MNIST dataset [Den12] (left) and on the Toronto Faces Dataset [Ban+21] (TFD).

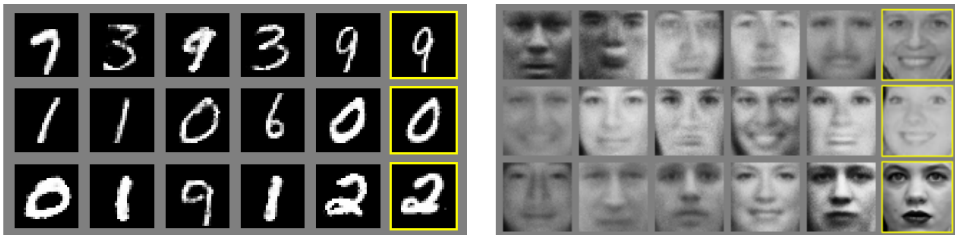


Figure 2.2: Example images generated by a GAN trained on the MNIST dataset (left) and the TFD dataset (right). The yellow highlighted images on the right show the closest real image from the dataset to the images form the previous column. (image source: [Goo+14])

Although in theory the distribution of the generated images p_g should converge towards the training data p_{data} [Goo+14, Proposition 2], in practice, GANs can be challenging to train. The most common reason for this is the training instability between the generator and the discriminator. If for instance, at some point during the training, the discriminator starts to outperform the generator, the generator will receive large loss values. Those loss values in turn produce large gradients, which can cause the generator

to diverge. If that happens, the output images from the generator look very poorly. For that reason, it is essential to keep a balance between the performances of both models, in order to successfully train a GAN.
Another common problem when training GANs is mode collapse. Mode collapse describes a state of a GAN, where the generator produces a very small amount of variation. This problem can arise if the generator successfully synthesizes one specific image that the discriminator classifies as real. The generator will then receive a good feedback for this specific image, which can enforce the generator to exclusively produce this specific image. An example for this effect can be observed in Figure 2.3 which shows the generated images of two GAN trainings with and without mode collapse.

Figure 2.3: An example demonstration of mode collapse. The generated images on the right show no variation. (image source: [Met+16])

2.2 StyleGAN

One of the most popular GAN architectures for deep image synthesis is the StyleGAN introduced by the Nvidia research group Karras et al. in 2019 [Kar+19]. It stands out for its state-of-the-art high resolution image synthesis with photorealistic quality. Some example images, synthesized with the StyleGAN model, are shown in Figure 2.4, demonstrating its (high) capabilities.

Figure 2.4: Example images synthesized by the StyleGAN model (image source: [Styc]), trained with the FFHQ dataset [KLA18b].

In this section, we will give a brief overview of the StyleGAN architecture along with a detailed explanation of two specific components. Those two components, weight modulation and path length regularization, are especially relevant for the experiments in the following chapters. In this thesis, we will only focus on the StyleGAN2 model, which is an improved version of the original StyleGAN [KLA18a] published in 2018. For simplicity, we will refer to the StyleGAN2 model simply as StyleGAN.

2.2.1 Architecture

Similar to conventional generative adversarial networks, the StyleGAN architecture also consists of a generator and a discriminator. While the discriminator is a traditional fully convolutional neural network that receives an input image and classifies whether it is real or fake, the generator of the StyleGAN was completely reinvented. Instead of feeding a random noise vector through a series of deconvolution layers to receive an output image, the StyleGAN starts with a constant input with a resolution of 4×4 and 512 channels. This constant input is then processed by multiple convolution and upsampling layers to produce an output image. Since this output image would be fixed, given that it uses a fixed input, the StyleGAN applies random latent vectors to modify the feature maps of the convolutions. This means that in the StyleGAN, the latent vectors are not used as basis for the convolutions, but instead control the *style* of a fixed image by manipulating the feature maps. Hence the name StyleGAN. The method for this style transfer operation is called weight modulation and will be further explained in the following section. An overview of the StyleGAN generator is shown in Figure 2.5. There it is visualized how the latent vectors z are utilized to manipulate the feature maps of the convolution with the constant input. After each convolution, the StyleGAN also adds stochastic variation to the images by adding mapped noise. The output of each layer is then up-sampled and forwarded to the next convolution layer. This is repeated until the output resolution is reached, after which the channels are reduced via a 1×1 convolution.

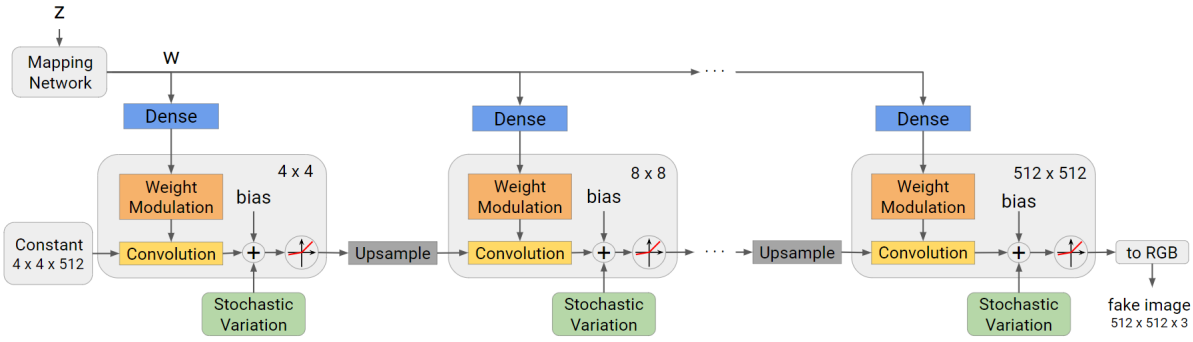


Figure 2.5: An overview of the StyleGAN generator architecture.

Before forwarding the latent vectors z to the weight modulation, they get also mapped by a small network that consists of eight fully connected layers. This means that the latent vectors z that are used to control the style of an output image are drawn from an intermediate latent space. This latent space will be referred to as the disentangled latent space W, as it is designed to spatially separate the image features of the latent vectors

2.2.2 Weight Modulation

The weight modulation component is the core operation of the StyleGAN. It is the only operation that enables the network to produce variation in the images, given that the convolution operations uses a constant input. Even though this input is also a trainable tensor, it is still necessary to input a random latent vector, in order for the GAN framework to work. The weight modulation component utilizes the latent vectors $w \in W$, which have been mapped from an input vector z to modify the feature maps before each convolution operation. This is done by multiplying each input feature map with a single scaling factor. This way, each input latent vector has an influence, how the input feature maps for each convolution operation are weighted to create the output image. In the StyleGAN, this method is not implemented by directly multiplying the feature maps, but instead multiplying each convolution kernel along the axis for the input feature maps. This has the same result, as the multiplications of the convolution are associative. By doing so, the weight modulation can be expressed by the following calculation:

$$w'_{ijk} = s_i \cdot w_{ijk} \tag{2.2}$$

Here, w denotes the original filter weights and w' the modulated. i is the axis of the input feature maps, in which each scaling factor s_i is multiplied, j is the dimension of the output feature maps and k is the combined width and height dimension of a kernel. For the StyleGAN model, k is equal to 3×3 for every layer.
Since, such a modulation operation can significantly change the statistics of the output feature maps, a normalization is applied. Instead of using an instance normalization method that scales the output feature maps based on its statistics, a so-called 'demodulation' operation is used. This 'demodulation' operation scales the output feature maps only based on the statistics of the modulated weights w'. This is done by dividing the output feature maps with the L_2 norm of the modulated weights w'. This way, both the input and output feature maps have the same standard deviation. Like the weight modulation operation, the demodulation operation is also realized by modifying the filter weights instead of the actual feature maps. By doing so, the resulting convolution weight can be expressed as follows:

$$w''_{ijk} = \frac{w'_{ijk}}{\sqrt{\sum_{i,k} w'^2_{ijk} + \epsilon}} \tag{2.3}$$

Here, the L_2 norm is performed for all modulated filters w'_{ijk} along the output dimension j, so that, effectively, each output feature map is divided by the L_2 norm of the filter coefficients. To avoid numerical issues, a small constant ϵ is added.

A visualization of the modulation / demodulation method is shown in Figure 2.6. It illustrates how the scales s_i effectively multiply each input feature map, while the demodulation multiplies each output feature map. For simplicity, we only use one filter in this example.

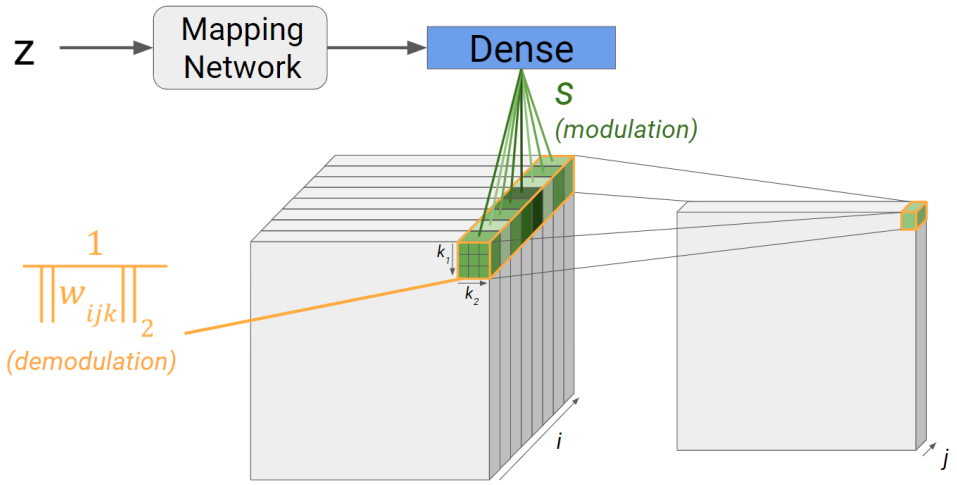


Figure 2.6: An illustration of the weight modulation method, where the convolution kernel is first modulated along the input dimension i and then demodulated along the output dimension j.

2.2.3 Path Length Regularization

The path length regularization is a component specifically designed to smoothen the interpolation paths in the disentangled latent space W. The main idea of this component is to align the distances in the latent space with the magnitude of change in the output images. This means that a small step in the latent space should also cause a small change in the output image and vice versa. This is a desired characteristic, since it helps to produce smooth animations when generating images along an interpolated path in the latent space if 'a fixed-size step in W results in a non-zero, fixed-magnitude change in the image' [Kar+19]. In addition to that, it also helps to locate specific images in the latent space, which will be done in chapter 5. To optimize this during the training, Karras et al. [Kar+19] propose the path length regularization method. This method first generates an image $G(w)$ in the generator, based on a latent vector w from the disentangled latent space W. The output image is then slightly modified by multiplying a random noise image I_{noise}. After that, the gradient of the resulting image to its corresponding latent vector w is calculated via backpropagation ($\nabla_w(G(w)I_{\text{noise}})$). If this gradient has a large magnitude, it indicates that the output image changes a lot when modifying the latent vector w and vice versa. Given that the noise image has a fixed magnitude, it is desired that the magnitude of the gradient is also fixed. In order to penalize the network, if the magnitude of the gradient changes a lot, it is compared to a moving average of all previous gradient magnitudes. This can be expressed with the following term [Kar+19]:

$$\text{Reg}_G = \mathbb{E}_{w, I_{\text{noise}} \sim \mathcal{N}(0,1)} \left(||\nabla_w (G(w) I_{\text{noise}})||_2 - A \right)^2 \tag{2.4}$$

where A is the exponential moving average of the magnitudes from all previous gradients during the training and I_{noise} the noise image that is multiplied to the output image.

2.3 Conditional Generative Networks

Later the same year in which Ian J. Goodfellow introduced the Generative Adversarial Network, Mirza and Osindero published a modified architecture of the GAN that made it possible for the first time to conditionally synthesize images [MO14]. There, they proposed an extension to the min-max value function (Equation 2.1) that incorporates labels. This extension can be expressed as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x,y \sim p_{data}(x,y)}[logD(x|y)] + \mathbb{E}_{z \sim p_z(z), y \sim p_y(y)}[log(1 - D(G(z,y)|y))] \tag{2.5}$$

Here, the generator receives an additional label input y to produce an image with the content of the current class. The discriminator, on the other hand, now predicts whether an image label pair (x, y) or ($G(z, y)$, y) is real or fake, given that it is from the class y. This means that both networks now require an additional label input, as shown in Figure 2.7.

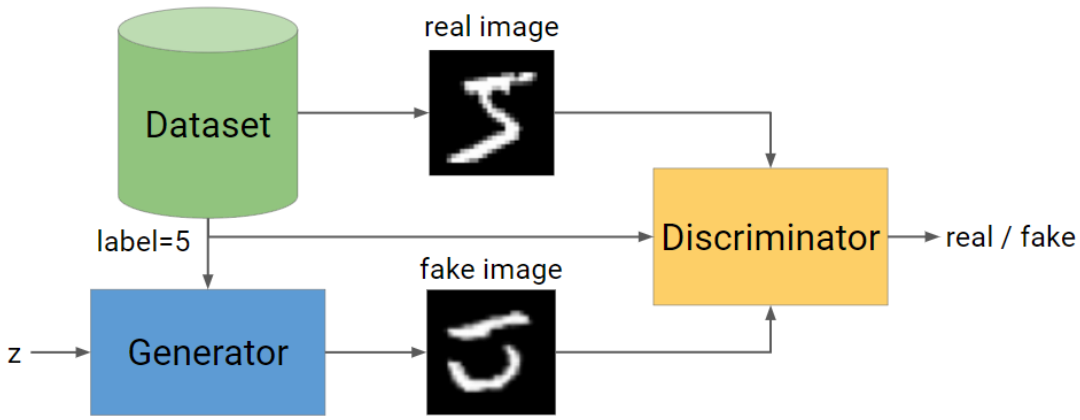


Figure 2.7: Overview of a basic Conditional GAN architecture.

The simplest way to incorporate this label is to concatenate it to the other inputs. Therefore, the inputs are commonly mapped to compatible dimensions using a dense layer [MO14].

This architecture enables the image synthesis with specific class inputs. An example for this is shown in Figure 2.8. There, each row is generated with a different input label.

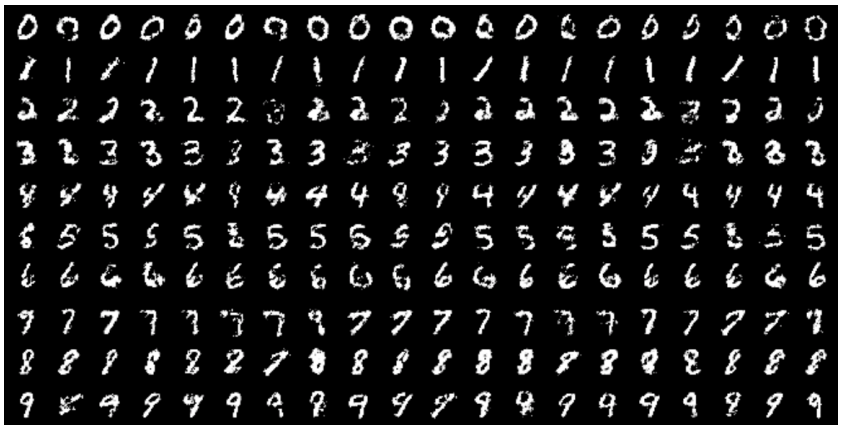


Figure 2.8: Example images generated by a conditional GAN trained on the MNIST dataset (image source: [MO14])

Although this method has shown to be successful, we will apply an alternative conditional architecture introduced by Mescheder, Geiger, and Nowozin [MGN18] for the conditional training with the StyleGAN. In this approach, the label is not directly forwarded to the discriminator. Instead, the discriminator is build to have one output neuron for each class. Then, if the discriminator has to predict whether the input image is real or fake, only the output neuron at the index of the current class is selected. This can be implemented by simply masking the output neurons with the label, given that the label only consist of zeros and ones. This architecture is visualized in Figure 2.9.

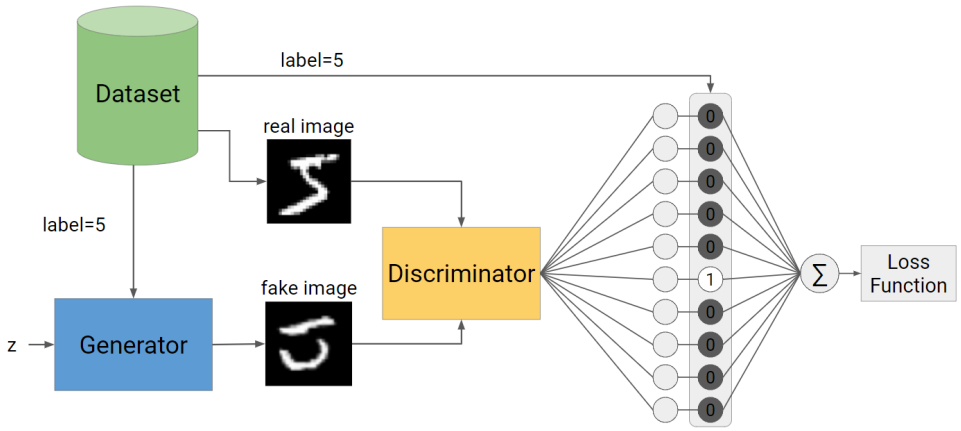


Figure 2.9: Architecture of an alternative conditional GAN by Mescheder, Geiger, and Nowozin [MGN18].

2.4 Fréchet Inception Distance

After successfully training a GAN, we obtain a generator that produces an infinite amount of images. While some of those images will have high quality and look very realistic, others might look very poor. In order to measure this quality quantitatively, it requires a metric that correlates well with human judgment. Therefore, we will use the Fréchet Inception Distance (FID) for this thesis. This metric was introduced by Heusel et al. in 2017 [Heu+17] and is next to the Inception Score [Sal+16] one of the most popular quality measurements for GANs. It uses a pre-trained classifier network (the Inception-V3 [Sze+15]) to calculate a distance between the generated images and real images from the training dataset. To do so, it calculates the feature vectors of 50000 fake images and 50000 real images and compares them with the following expression:

$$\text{FID} = \|\mu_1 - \mu_2\|_2^2 + \text{Tr}\left(C_1 + C_2 - 2\sqrt{C_1 C_2}\right) \tag{2.6}$$

Here, μ_1 and μ_2 are the means, and C_1 and C_2 are the covariance matrices of the feature vectors from the real and fake images. Tr denotes the linear trace operator that calculates the sum of the diagonal entries of a matrix. A low FID output corresponds to two very similar image distributions, which in turn implies that the quality of the generated images is high. This can be demonstrated if we input the same image distributions two times. Then the FID outputs zero, as demonstrated in Equation 2.7.

$$\begin{aligned} \text{FID} &= \|\mu_1 - \mu_1\|_2^2 + \text{Tr}\left(C_1 + C_1 - 2\sqrt{C_1 C_1}\right) \\ &= 0 + \text{Tr}\left(2C_1 - 2\sqrt{C_1^2}\right) \\ &= 0 + \text{Tr}\left(2C_1 - 2C_1\right) \\ &= 0 \end{aligned} \tag{2.7}$$

To underline the effectiveness of this metric, Figure 2.10 shows some examples, where the FID correlates with increasing image disturbance.

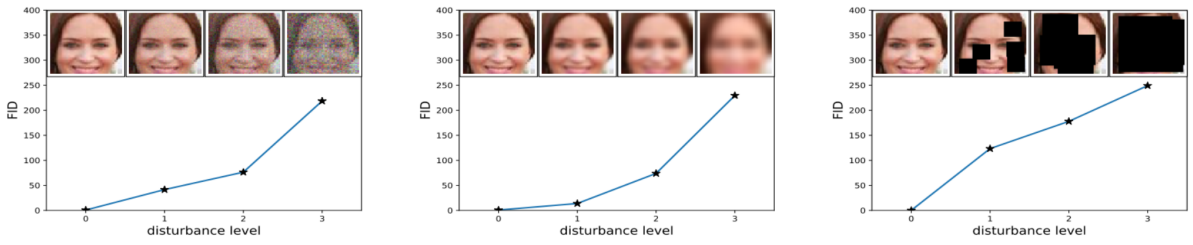


Figure 2.10: A demonstration how the FID correlates with increasing image disturbance (image source: [Heu+17])

3 Training Data

The dataset that we use for all experiments is the car dataset created by the Chair for Multimedia Computing and Computer Vision of the University of Augsburg. To give a brief overview of the dataset, we first describe how the images need to be adapted in order to use them as training data for the StyleGAN model. Afterwards, we give a summary of the labels that we will use for the conditional image synthesis. And at last, we point out some problems and biases of the dataset.

3.1 Image Adaptation

The car dataset consists of about 74000 car images with resolutions ranging from 75×75 to 8688×5792 pixels, and aspect ratios ranging from 1.0 to 3.0 ($\frac{width}{height}$). Due to the architecture of the StyleGAN model, however, both have to be adapted. This is because the generator starts at an internal feature map resolution of 4×4 and then uses multiple up-sampling operations to output an image at a $2^n \times 2^n$ resolution. Therefore, the images of the car dataset need to be adapted to also have a square ratio and a unified resolution. To do this without losing too much of the image content, we first crop the images to the next closest discrete image ratio [1.2, 1.4, 1.6, 1.8, 2.0] ($\frac{width}{height}$) and then pad them with zeros at the top and at the bottom to receive a squared image ratio. At last, we also resize the images to a $2^n \times 2^n$ resolution. Two examples for this editing pipeline are visualized in Figure 3.1.

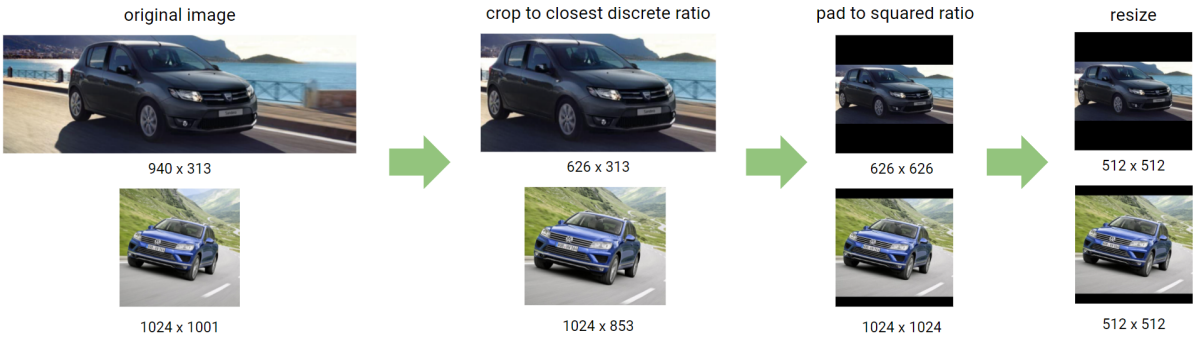


Figure 3.1: Two examples images that are processed by the image editing pipeline, that was used to prepare the images from the car dataset for the StyleGAN training. The images are first cropped, then padded, and at last resized.

This demonstrates how two images with very different aspect ratios can be adapted for the StyleGAN training without losing too much information about the car. Although it might be reasonable to crop the image towards the location of the car, by using a pre-trained detector network, it showed to be sufficient to simply crop the image towards the center, since almost all of the cars are located in the center.
The quantizised image ratios that we choose are based off the distribution of the aspect ratios from the original images. They showed to cover the majority of the ratios, without cropping too much of the image content. The training can also be done without cropping the images at all and instead only padding and resizing them. This, however, would create some images with a very large zero-padded area. In addition to that, using a discrete ratio enables us to create a ratio label, which can be used in the training of the conditional StyleGAN, to control the image ratio of the output images.

3.2 Labels

To train a conditional GAN, as described in section 2.3, we need to provide the network with one-hot encoded labels. For this thesis, we chose the following labels: car color, car model, car manufacturer, car body style, car rotation, image background, and image aspect ratio. While the model, manufacturer, color and aspect ratio labels have already been provided by the car dataset, the rotation, background, and body still had to be added. To do so, we labeled a total of 10700 images (about 600 per class) by hand, using the PicArrange [Jun+21] tool. This tool is designed for quickly finding visually similar images, which facilitated the annotation process considerably. We then used those labeled images to train a pre-trained resnet50 model [He+15] for a duration of 80000 training images. After that, we selected the checkpoint with the highest accuracy on a separate test set (20% of the data) and utilized it to classify the rest of the images from the dataset. In order to obtain a high classification accuracy, we only selected the classifications that had a confidence higher than 80% and filtered those images again by hand. To give a brief overview of all seven labels, we summarize them in Table 3.1. For the three additional labels (rotation, background and body style), we also give further details and examples in the following subsections.

Label	Number of Classes	Dataset Coverage	Annotation Method
Model	67	95%	image search engine keyword
Manufacturer	18	95%	image search engine keyword
Color	12	41%	hand annotated
Body Style	10	53%	classifier + filter by hand
Rotation	8	96%	classifier + filter by hand
Background	6	59%	classifier + filter by hand
Ratio	5	100%	image width / height

Table 3.1: Overview of all seven labels of the car dataset.

Car Rotation

The rotation label is the most important annotation for this thesis, since we will use it in chapter 7 to synthesize a 3D view of a car. It is split into eight discrete classes, distributed uniformly along a full 360° rotation. This means that each of the eight classes describes an angle $\phi \in \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$ as shown in Figure 3.2. While the $k \cdot 90°$ rotations have been annotated very accurately, the others can slightly vary in their rotation angle. For example, the 45° rotation class mainly describes that the car rotation is in between $10° > \phi > 80°$. This is done for the following two reasons. The first reason is that the labels should never overlap each other. This could be the case if all cars with a rotation angle smaller than 22.5° get quantisized to 0°, while all cars with a greater rotation angle get quantisized to 45°. Any car with an angle of about 22° could then be classified with either class, given that the actual rotation angle is difficult to measure in 2D images. The second reason for this decision is that the $k \cdot 90°$ angles act as pivot points at which the appearance of the car changes the most during a rotation. For generating a realistic rotation with a conditional StyleGAN, it proved to be essential that those angles are as precise as possible.

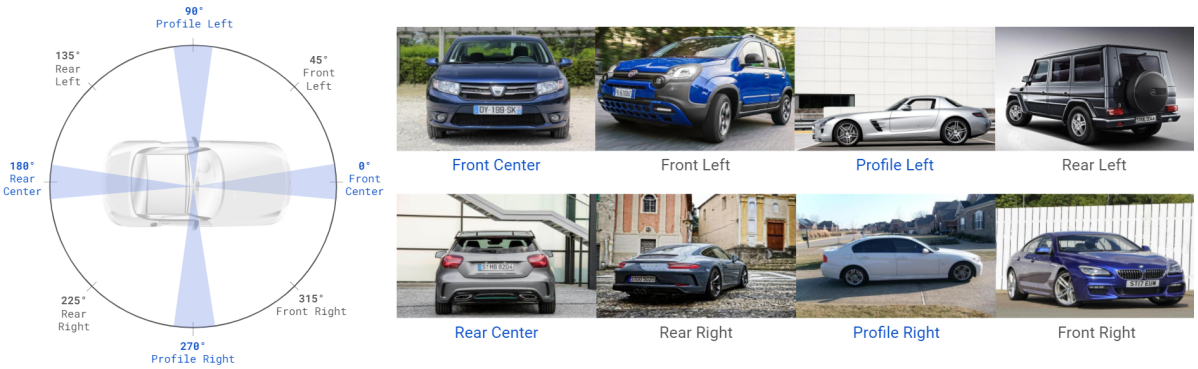


Figure 3.2: Example images for the rotation label. Each class describes a $k \cdot 45°$ angle around the car. The blue and white sections in the rotation circle denote the angle intervals that quantisize the eight discrete labels.

Image Background

The background label consists of about 44000 training examples, which are separated into six classes. Those six classes are: showroom, city, countryside, off-road, white and black. The city label contains all images with buildings in the background, the countryside label all images with nature in the background, and the off-road label contains all images where the car is not driving on an asphalt road.
The histogram of the background label shows that images taken in a showroom or in the city are much more represented in the dataset compared to images with a black or white background. Although such label unbalance can harm the generalization quality of the

model, we expect it to work well, given that images with black and white backgrounds are much simpler to generate than images with complex structures, such as buildings or trees.

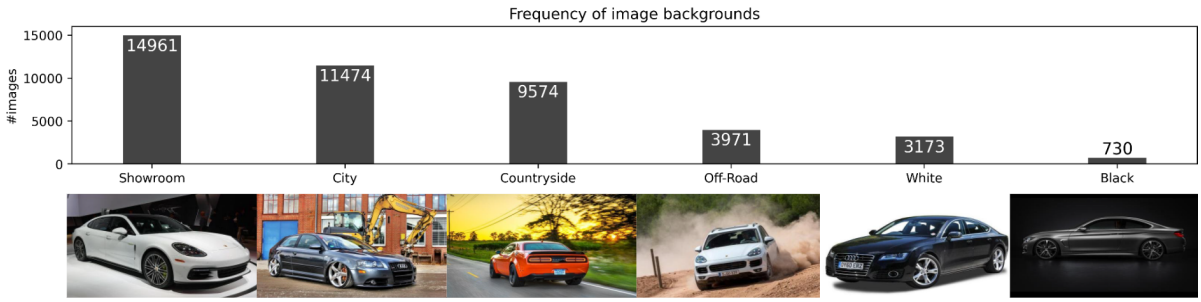


Figure 3.3: The frequency distribution of the image background along with example images.

Car Body

The car body label consists of 39000 training examples that are split into 10 classes. An example for each class is shown in Figure 3.4 along with their frequency in the dataset. Since the coupé and the off-road class can sometimes overlap with other body styles for some cars, we decided that only those images get classified as coupé or off-road that do not fit in any other more specific class. For example, we classify cars that are convertibles and a coupés at the same time only as a convertibles.

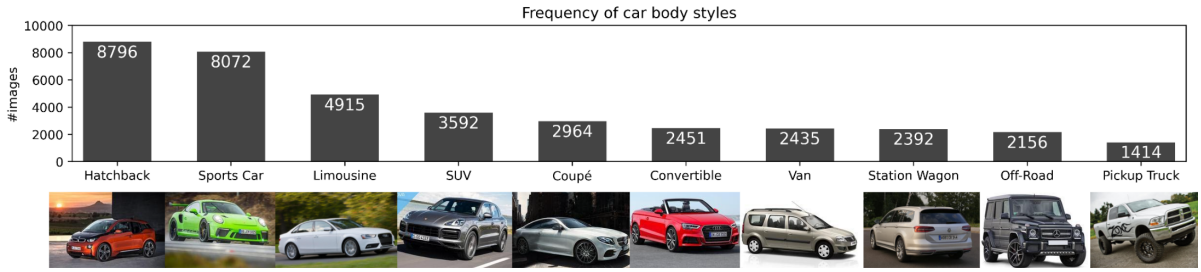


Figure 3.4: The frequency distribution of the car body labels along with example images.

3.3 Dataset Biases

A common problem when dealing with datasets are dataset biases. Dataset biases are characteristics of datasets that can reduce the generalization quality when training a machine learning algorithm [Tom+15]. The two biggest dataset biases that can be found in the car dataset are selection bias and capture bias. Those biases have been

defined by Torralba and Efros [TE11] and describe characteristics based on the method the images were collected and a bias based on how the photos were taken, respectively. The selection bias of the car dataset can, for example, be found in the distribution of the car manufacturers, shown in Figure 3.5. It highlights that German car manufacturers are much more represented than others. While 16% of the cars are BMWs, only 1% are Hyundais. Given this bias, we expect that the generated images show a higher quality for BMWs than Hyundais, as they get sampled much more frequently during training. The capture bias on the other hand can be found in the distribution of the car rotation (Figure 3.5). It shows that most images were taken from the 'front left' or 'front right', whereas only very few were taken from the rear center. This is likely because 'front left' and 'front right' are very common angles to take a photo of a car. Since the rotation label is the most important label for this thesis and some prior experiments with the dataset showed that an unbalance in the rotation can cause major problems in generalization quality, we oversample this label during the training. This means that we sample training examples with less frequent rotation angles more frequently. This is demonstrated in the red graph of Figure 3.5, where each class now has a minimum of 10000 training images. We limit this frequency by 10000 to avoid any mode collapse problems during the training, which is more likely if we duplicate some of the images too often. In order to still balance the 'front left' and 'front right' classes with the rest of the rotations, we randomly remove some of them during the training. This way, we can keep all of the labels, while also none of the classes get oversampled too much.

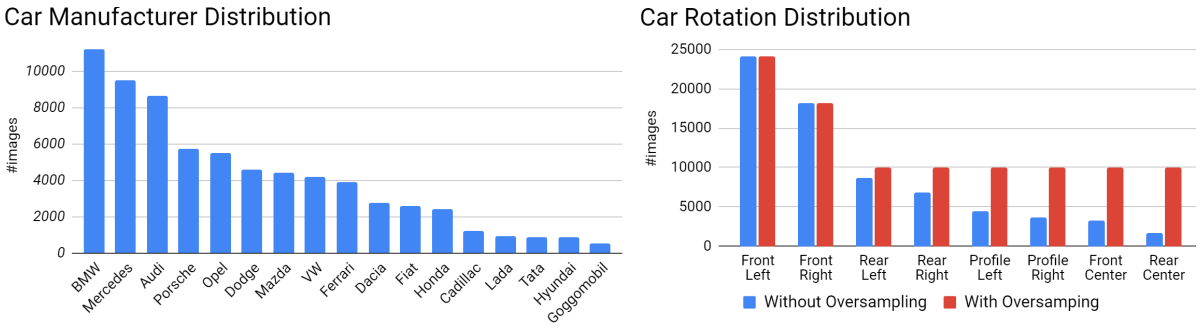


Figure 3.5: The frequency distribution of the car manufacturer classes (left) and rotation classes in the car dataset (right). It shows that some classes are much more represented than others. To encounter this, we oversampled the rotation label during the training, which leads to the distribution highlighted in red.

One further bias that can be found in the labels of the car dataset is the correlation between some of the labels. An intuitive example for this is the color distribution of Ferraris. It showed that about 60% have the color red, whereas only about 1% are green, orange or purple. Given this unbalance, we expect that generated images with red Ferraris show considerably better quality than Ferraris in other colors. This is because those combinations are generated much more often during the training. Similar connections can also be found between other labels. For an example, a car with an off-road body style is more frequently found in an off-road background.

4 Controlling the Image Synthesis of GANs

While GANs have shown a lot of success in generating photorealistic images in recent years, it still is very difficult to control specific attributes in the synthesized images. One major reason for this is the complexity of the generator, which is trained to create realistic images only from high dimensional random noise inputs (or latent vectors). Since this random latent vector is the only controllable input parameter for traditional GANs, it is only possible to customize a generated image with specific attributes (e.g. car color or background), by finding the corresponding latent vector. This, however, can be very exhaustive and time consuming, considering the high-dimensional, infinitely large, latent space. To tackle this problem, this chapter introduces three methods from literature to control the image synthesis with GANs. In order to apply the following three methods, we train the standard StyleGAN as described in section 2.2 with the images from the car dataset. This training was performed at an image resolution of 512×512 pixels for a duration of 8 million training images and achieved a FID of 2.85.

4.1 Latent Space Gradient Descent

The first method for controlling the image synthesis of the generator applies the gradient descent algorithm onto the latent space. The goal of this method is to find the best latent vector that re-synthesizes a given target image. The method was introduced by Karras et al. in the StyleGAN2 paper [Kar+19] and is originally motivated to detect whether a given image is real or was generated by the StyleGAN model. This is an important task in the field of image synthesis, since generated photorealistic images can cause security issues when they are used 'to counterfeit some personal information in social networks' [Li+18]. To counteract this problem, this method is designed to check if a latent vector can be found that exactly re-synthesizes the target image. In such a case, it is very likely that the image is fake. In addition to checking if an image was generated by the StyleGAN model, it can also be utilized to control the output of the generator. To do so, simply a target image is provided to the method, which the model then aims to re-synthesize.
The implementation of this method can be described as follows: First, an image with a random initial latent vector is generated. Then, this image is compared to the target image using a perceptual distance which can be used as a loss to calculate a gradient to

the input latent vector. The latent vector is then updated towards the opposite of the gradient, like in a traditional gradient descent algorithm. This process is repeated for a large number of iterations until the perceptual distance of the generate image and the target image is very low. At this point, both images look very similarly to each other. A demonstration of this method is shown in Figure Figure 4.1, where an image from the training dataset is re-synthesized in 1000 update steps. This takes about 13 minutes using an Nvidia GTX 1070.

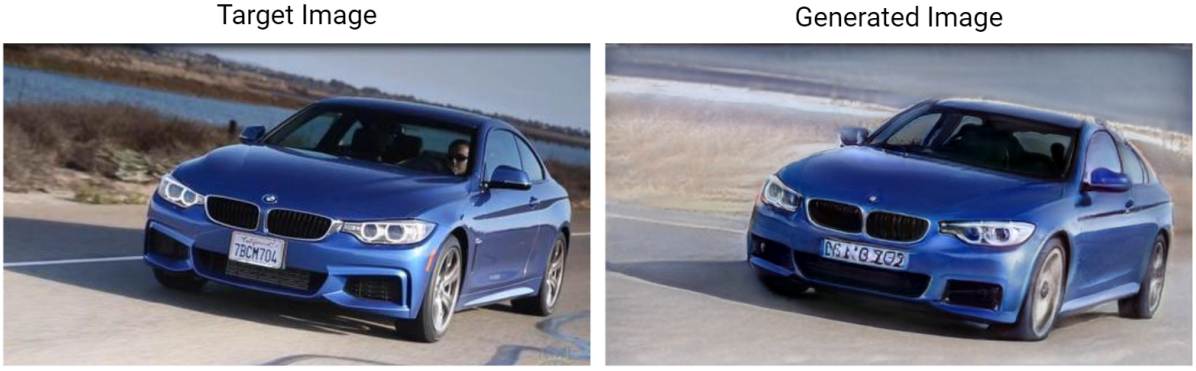


Figure 4.1: A demonstration of a re-synthesized image, using gradient descent in the latent space.

In summary, this method works very well for finding specific images in the latent space. On the downside, however, this method is very computationally expensive. In addition to that, it also does not enable the synthesis of new images, since it always requires a target image. An extension to this method to create new images could be to find the latent vectors of two target images and then generate a third image with the mean of both latent vectors. The resulting output image then might show a combination of both target images. This extension, however, would require even more computations than before.

4.2 Style Mixing

The next conditional synthesis method is style mixing introduced by Karras, Laine, and Aila [KLA18a] along with the first version of the StyleGAN in 2018. Back then, the style mixing method was originally designed as a regularization technique to increase the variation of the generated images during the training. However, it can also be utilized to control the output of the generator. The main idea of style mixing is to combine two latent vectors of two generated images, so that a third image can be generated, that has some attributes from the first image and some from the second. To do this, two latent vectors z_1 and z_2 are first mapped to the disentangled latent space W and then forwarded to the generator at different layers, as visualized in Figure 4.2 (right). The generator then uses those mapped latent vectors to modulate the weights of the

convolutions and therefore manipulate the attributes of the output image. Depending on the layer at which the latent vectors manipulate the weights of the feature map, different attributes are changed in the output image. An example for this is shown in Figure 4.2 (left), where in each column the latent vector w_1 is replaced with w_2, at the corresponding layer. It demonstrates that changing the latent vector of the earlier layers of the generator corresponds to coarse changes in the image, such as rotation, while the latter layers mainly change fine details or colors.

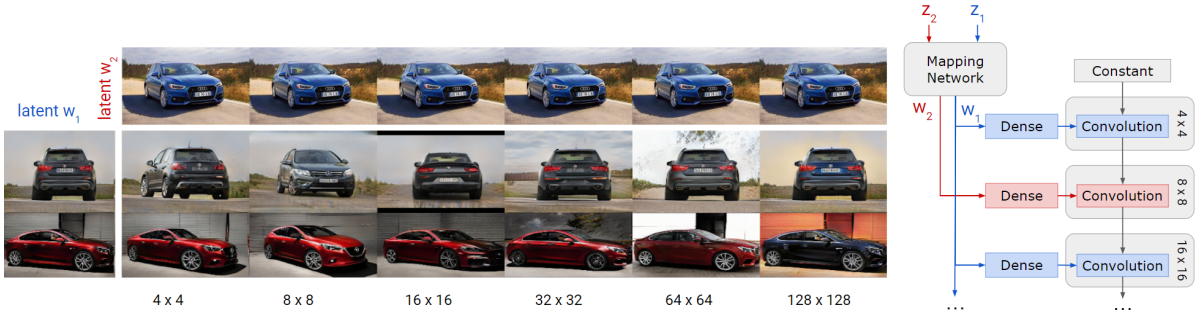


Figure 4.2: A demonstration of the style mixing method (left), where each column denotes the layer at which the latent vector w_1 is replaced by w_2. On the right also an illustration of the generator performing the style mixing is shown.

Now, in order to take advantage of this method and utilize it for conditional image synthesis, multiple latent vectors can be combined at different layers in the generator to obtain the desired output image.
Altogether, this method is very simple to use and does not require a lot of computing time. However, before customizing images by combining the features of different images, the corresponding input latent vectors first have to be found. As previously stated, this can be very difficult. In addition to that, this method does not allow to manipulate any attributes other than those that are provided in the layers of the StyleGAN. Moreover, it is not possible to alter only a single attribute of the image. This, for example, is shown in Figure 4.2 (left), where in addition to the rotation in the second column, also some parts of the background change.

4.3 Principle Component Analysis of the Latent Space

The third method for controlling the image synthesis was proposed by Härkönen et al. in 2020 [Hä+20]. The method utilizes the Principle Component Analysis (PCA) to alter specific attributes in the generated images. To do so, the PCA is calculated with a large number of latent vectors in the disentangled latent space W, to determine the main components of the latent space. With those components, an image can then be manipulated by creating a latent vector from a linear combination of weighted PCA components. It shows that each component mainly corresponds to one specific image attribute. The first few components, with the largest eigenvalues, mainly correspond to

coarse features, such as the rotation of the car, while higher components with smaller eigenvalues, correspond to smaller changes in the output image, such as the color. An example for this is shown in Figure 4.3 and Figure 4.4. Figure 4.3 shows an example, where the second component is multiplied, which caused the generated car to rotate by about 90°. And Figure 4.4 shows an example where the 20th component is multiplied, which changes the color of the car.

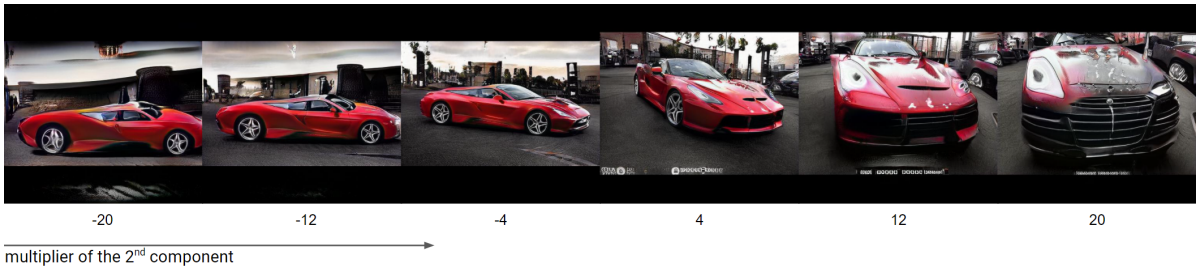


Figure 4.3: An example output sequence when adding the 2nd PCA component, multiplied by values from -20 to 20, to the latent vector. It shows that the car is rotating at 90°.

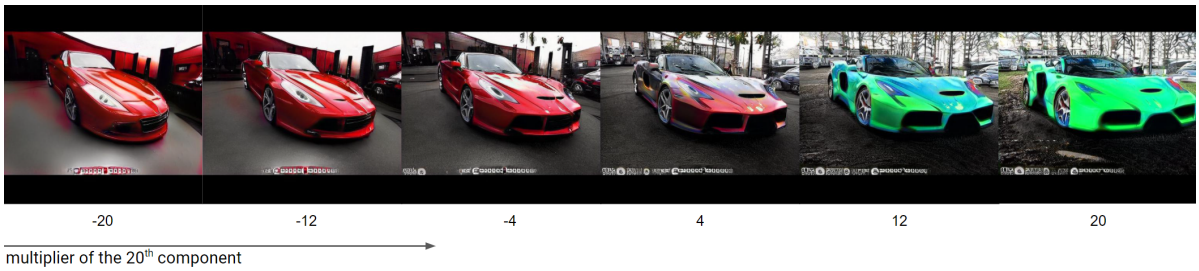


Figure 4.4: An example output sequence when adding the 20th PCA component, multiplied by values from -20 to 20, to the latent vector. Here, it changes the color of the car.

In order to customize an output image with this method, the components that are corresponding to specific attributes can simply be altered. This method is easy to use and computationally cheap. It, however, does not allow to modify any attributes other than those that are encoded in the main components. In addition to that, we also observe that multiple attributes are changed at the same time when multiplying one component. This can be seen in in both examples, where in addition to the color and rotation, the background and the car body also change. This makes it, like the previous method, very difficult to fully customize the output of the generator.

4.4 Summary

While the three presented methods show some capability to modify the content of the generated images, they were all based on finding a specific latent vector in the latent space, either by hand or with the gradient descent algorithm. This, however, as stated before, can be very difficult or computationally expensive, considering the large 512 dimensional continuous latent space. In order to avoid such an extensive search, we apply the characteristics of a Conditional GAN to the StyleGAN model. By doing so, the synthesis of images with specific attributes of the labels that are provided by the car dataset is enabled. In the next chapter, we will implement the conditional behavior in the StyleGAN architecture and perform several experiments with the goal of improving the performance of conditional image synthesis.

5 Conditional StyleGAN Experiments

In this chapter, we will implement and improve the conditional image synthesis for the StyleGAN model. To do so, we will first implement a baseline model that combines the standard StyleGAN with a Conditional GAN. After that, we propose multiple modifications to the baseline model and test them in a series of experiments. At the end of the chapter, we will combine all successful modifications into one model to test how well the proposed modifications work together. In order to compare the performance between each experiment and the baseline model, we also introduce four characteristics for good conditional image synthesis and propose four metrics to measure each aspect quantitatively. Those four characteristics are:

1. High image quality
2. High conditional accuracy
3. Low label entanglement
4. High image quality for unseen label combinations

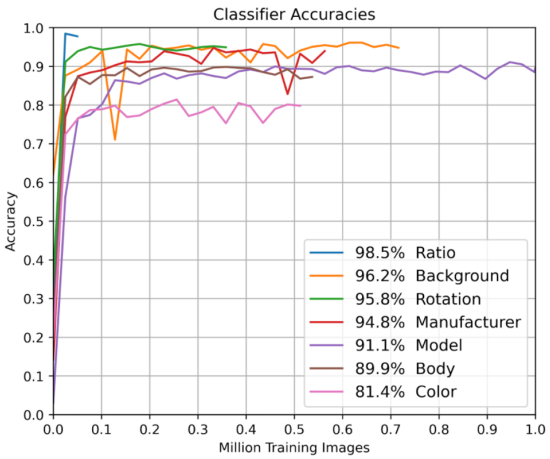
In the following four sections, we will give a more detailed explanation for each characteristic and propose a metric for each.

5.1 Image Quality

Although a generally high image quality is not a characteristic that is specifically desirable when training a Conditional GAN, it is still an essential aspect for image synthesis. To measure it quantitatively, we apply the FID metric from section 2.4 to compare the image qualities of the following experiments. As described in section 2.4, the FID compares the image distribution of the generated images to the image distribution of the training dataset, using a pre-trained perceptual network. A low value states that both distributions are similar to each other, which corresponds to high image quality. Given that we now generate images with an additional label input, we simply draw the labels for fake images of the FID calculation from the dataset. This way, the images are as close to the real data as possible. Apart from that we do not change the FID calculation.

5.2 Conditional Accuracy

When training a Conditional GAN, high conditional accuracy is obviously particularly important. It reflects how well the labels have been learned by the model. To measure this aspect, we utilize multiple pre-trained classification networks to calculate an accuracy between the input labels of the generator and the classifier predictions for the output images. To do this, we train one classifier for each of the seven labels from the car dataset. For each classifier, we use a pre-trained VGG16 backbone [SZ14] with an additional classification head. At the end, we activate the output with a softmax function and calculate a cross entropy loss.
After training each classifier, they score accuracies from 81% to 98% on separate test sets, as shown in Figure 5.1 (left). We decided to train each classifier for a different duration, since it showed that some labels are considerably more difficult to learn. For instance, we performed the longest training with the car model classifier, since this label contains 67 different classes, of which some only differ in small details.



Confusion Matrix of the Color Classifier (in %)

predicted class / target class

	white	silver	gray	black	blue	red	brown	green	yellow	orange	purple	undefined
white	93	4	0	0	0	0	0	0	0	0	0	2
silver	12	67	14	0	2	0	1	1	1	1	1	1
gray	1	10	66	1	2	0	5	2	0	1	0	2
black	0	0	9	75	4	0	3	3	0	0	5	2
blue	1	2	2	2	85	0	1	3	0	0	3	2
red	0	0	0	0	0	87	1	0	0	3	7	1
brown	0	1	4	2	0	2	72	0	2	5	11	1
green	1	1	1	1	0	0	0	94	1	0	0	2
yellow	0	0	0	0	0	0	0	5	90	3	0	2
orange	0	0	0	0	0	9	1	1	1	87	0	2
purple	0	0	0	3	2	2	3	0	0	1	90	1
undefined	5	1	2	1	0	1	0	1	1	1	5	82

Figure 5.1: The classifier accuracies over the training (left). And a confusion matrix of the color classifier, which performs the worst of all classifiers. It shows that the color classifier mainly miss classifies the classes silver, gray, and brown.

The classifier with the overall worst performance is the color classifier. This classifier only scored an accuracy of 81%, whereas the others scored about 90%. To understand this, we generate a confusion matrix using the test set (Figure 5.1 right). This confusion matrix highlights that the color classifier especially struggles to differentiate the classes: silver, grey and black. Given that those colors can be very similar to each other it might explain why the classifier performs this poorly.
The only classifier that we trained with a different network architecture compared to the other classifiers is the image ratio classifier. This is because a fully convolutional network turned out to perform very poorly at classifying global image features such as the image ratio. Instead, we then simply used a smaller network that only consists of one convolution layer with a 1 x 1 kernel and one dense output layer. This network then

showed to classify the images with a 98% accuracy after only five thousand training images.
With the help of all seven classifiers, we can now calculate the average conditional accuracy over all labels as follows:

$$\mathcal{A}_{cond} = \frac{1}{|\text{labels}|} \sum_{l}^{\text{labels}} \frac{1}{N} \sum_{i=1}^{N} equal(C_l(G(z, y_i)), y_i) \tag{5.1}$$

Here, C_l denotes the classifier of the label l, G the generator and y_i the input label. The *equal* function is defined to output a 1 if both inputs are equal, and a 0 otherwise. For this metric, we set N to 5000. This means that calculate the average accuracy for a total of 35000 images.

5.3 Label Entanglement

The next desired characteristic for conditional image synthesis is a low label entanglement. This characteristic describes how many other image attributes change when modifying a label. If we change the color of a car from red to blue, for instance, we do not expect any other attributes (i.e. background or rotation) to change as well. In practice, however, we observe that this is often the case and that some labels have a strong entanglement (or connection). To measure this entanglement quantitatively, we utilize the classifiers from the previous metric to predict the class of each label before and after changing a class of a different label. I.e., we estimate the probability how often the prediction of a label changes when manipulating another label. To do so, we calculate the average number of changes for each classifier. In order to output a single value for this metric, we average the number of changes from all classifiers. This means that a low output indicates a low label entanglement and vice versa. This can be expressed as follows:

$$\mathcal{E} = \frac{1}{|\text{labels}|} \sum_{l}^{\text{labels}} \frac{1}{N} \sum_{i=1}^{N} equal(C_l(G(z_i, y_i)), C_l(G(z_i, y_i'))) \tag{5.2}$$

Here, y and y' are the labels before and after one class of a label (that is different to l) has been modified and C_l denote the classifier of label l. The *equal* function is defined as in the previous metric. For this metric, we choose N to be 10000.

5.4 Image Quality for unseen Label Combinations

The quality of images, synthesized with unseen label combinations, is especially interesting when training a Conditional GAN with multiple labels. This is because GANs allow us to extrapolate the training data in order to create photorealistic images from objects that do not exist in the real world. Now, given the ability to control the output images with input labels, we can observe how the generator handles new label combinations that do not exist in the real world. For example, when generating a green Ferrari with a body of a pickup-truck, which does not exist in the training data or in the real world, we still want to receive a photorealistic output images in which some characteristics of a Ferrari get combined with the body of a pickup-truck. In order to measure this aspect quantitatively, we again use the FID metric from section 2.4. However, instead of drawing the labels from the dataset, like it is done in section 5.1, we now create random label combinations. This way, we measure the image quality for images that were generated from new and unseen label combinations.
For this metric, we have to consider that the output will likely be a higher compared to the FID of section 5.1. This is because, we assume that the real labels from the dataset should help to produce an image distribution that is closer to the real images.

5.5 Implementing the Conditional StyleGAN

In order to train the StyleGAN as a Conditional GAN, we have to perform two modifications. First, we have to integrate the labels into the latent space so that the generator can synthesize the images based on them. And second, we have to adapt the output of the discriminator, so that the discriminator does not solely decide whether the image is real or fake but instead evaluates how well each attribute of the given label has been fulfilled. In the following sections, we describe two methods, how this can be implemented in the StyleGAN. Before that, however, we first give a brief overview of the input label vector.

Input Label

Given that the images in the car dataset have multiple labels, we utilize a vector that concatenates all one-hot vectors from all seven labels for the conditional training. This 127-dimensional vector is visualized in Figure 5.2.

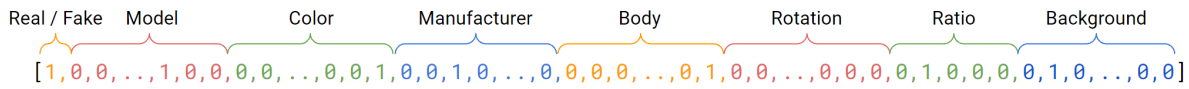


Figure 5.2: The label vector for the conditional StyleGAN training. The vector concatenates all one-hot labels from the car dataset and has a total length of 127.

Since not every image in the dataset is labeled with all seven labels, we simply fill a label segment with zeros, if the corresponding label does not exist. In addition to that, we decided that the first entry of every label vector will be always filled with a 1. This entry will be used as a 'real / fake' label, similar to a traditional GAN, trained without labels. It is added to receive an output from the discriminator independently of the labels, given that not all image features are covered by the labels. This gives the discriminator the ability to decide whether an image was real or fake, without having to specify which of the seven given labels were realistic or not.

Integrating the Labels into the Latent Space

The first StyleGAN modification deals with the integration of the label information into the latent space. The method that we use was first introduced by Cedric Oeldorf and Gerasimos Spanakis [OS19] in 2019.
In the traditional StyleGAN, we only provide the generator with a random latent vector z. In order for us to train the StyleGAN as a conditional network, however, it is required to also add a label. To do so, we multiply the label vector, as described in the previous subsection, by a trainable tensor $T \in \mathbb{R}^{127\times512}$. This tensor T has the shape 127 and 512 which are the dimensions of the label and the dimension of the latent space vector z, respectively. After that, we receive a tensor with an output shape of $N \times 512$, with N being the batch size. The resulting tensor is then concatenated with the latent vector z and forwarded to the mapping network of the generator. After that, the resulting mapped latent vector is used to modify the filter weights of the convolutions in the generator. This method can be formulated as follows:

$$w = f([y \times T]^\frown z) \tag{5.3}$$

where f is the mapping network, z the latent vector and y the label [OS19]. The symbol $^\frown$ denotes the concatenation operation.

Adapting the Discriminator

The second method for adapting the output of the discriminator was first introduced by Lars Mescheder [MNG18] in 2018.
When training the StyleGAN without labels, the discriminator only has to decide, whether the shown image is real or fake. However, when training with labels, this decision does not suffice. Instead, the discriminator now has to differentiate how well the input image fulfills the attributes of the given labels. In order to implement this, we apply an additional dense layer with linear activation at the output of the discriminator, which is equipped with 127 output neurons. Since this is the number of classes in the concatenated label vector, we receive one output neuron for every class. After that,

we use the label vector that only consists of zeros and ones as a mask for the output neurons, since we are only interested in the output of the neurons that correspond to the current label. At the end, we calculate the sum and forward the result to the loss function. This can be expressed as follows:

$$D = \sum_{i}^{127} D'_i \cdot y_i \tag{5.4}$$

Here, D' is the output layer of the discriminator, which has one neuron for each for each class in the label vector y [OS19]. A high output of D corresponds to high activations at the correct label positions. Like in a traditional GAN, the discriminator then aims to maximize the output of D for real images and minimizes it for fake images, whereas the generator aims to maximize this output for fake images. Before forwarding the output of D to the loss function, we also apply a sigmoid function. This way, the output values are between 0 and 1 before calculating the logarithm in the loss functions.

5.6 Baseline Model

In our first experiment, we train the StyleGAN model as a conditional adversarial generative network, using the labels from the car dataset described in chapter 3. This experiment functions as a baseline model for the following experiments that propose several modifications based on this model. Before presenting the training results, however, we give a more detailed description of the training configuration and the loss function of the StyleGAN.

Loss Functions

The loss functions for the generator and discriminator are based on the min-max value function from section 2.3. This value function suggests that the generator minimizes the term $log(1-D(G(z,y),y))$, while the discriminator maximizes both $log(1-D(G(z,y),y))$ and $logD(x)$. The most straightforward way to express these goals with two loss functions would be as follows:

$$\mathcal{L}_G = \frac{1}{N} \sum_{i=1}^{N} \log\left(1 - D(G(z_i, y_i))\right) \tag{5.5}$$

$$\mathcal{L}_D = -\frac{1}{N} \sum_{i=1}^{N} \left[\log\left(D(x_i, y_i)\right) + \log\left(1 - D(G(z_i, y_i), y_i)\right)\right] \tag{5.6}$$

Here, simply the value function is inverted for the discriminator loss. This way the term can be minimized instead of maximized. N denotes the batch size. While this loss is also used for the StyleGAN discriminator, a slightly modified loss is chosen for the generator. Instead of minimizing the term $\log(1 - D(G(z, y)))$, a non-saturating loss is utilized [Kar+19]:

$$\mathcal{L}_G = -\frac{1}{N}\sum_{i=1}^{N} \log\left(D(G(z_i, y_i))\right) \tag{5.7}$$

This loss has the advantage that the absolute value of the loss does not converge to 0 as the generator decreases in performance. It is avoided, given that $-\log(D(G(z, y)))$ diverges to ∞ as the fake images achieve low activations in the discriminator:

$$\lim_{D(G(z,y))\to 0} -\log\left(D(G(z, y))\right) = \infty \tag{5.8}$$

As a result, the chance for diminishing gradients is reduced.

Training Details

For all following experiments, we perform a training with a duration of five million training images, shown to the discriminator. This takes about 7 to 9 days, using two Nvidia GTX 1080ti GPUs, depending on the training configuration. The whole model has about 59 million trainable parameters that are evenly balanced between the generator and discriminator. Although the car dataset provides images at a high resolution, we set the output resolution to 256×256 pixels for all the following experiments. This resolution is selected since it offers a good trade-off between training time and image detail. Later, in chapter 8, we also train a model at a 512×512 resolution, in which all successful components from this chapter and the next chapter get combined. During the training, the FID metric is evaluated every 80 000 training images and a network snapshot is saved. For all experiments the Adam optimizer is used and none of the hyperparameters, such as the learning rate or the batch size, are modified. During the training, we also augment the images from the dataset by randomly flipping them horizontally.

Training Results

After training the baseline model for a duration of 5 million images, shown to the discriminator, we measure a lowest FID of 4.46, as shown in Figure 5.3 (left). This value is relatively low and indicates that the generated image distribution is close to the image distribution of the training data. This is also confirmed when inspecting some of the generated images in Figure 5.3 (right). Those images look very realistic, as they show a lot of detail and variation.

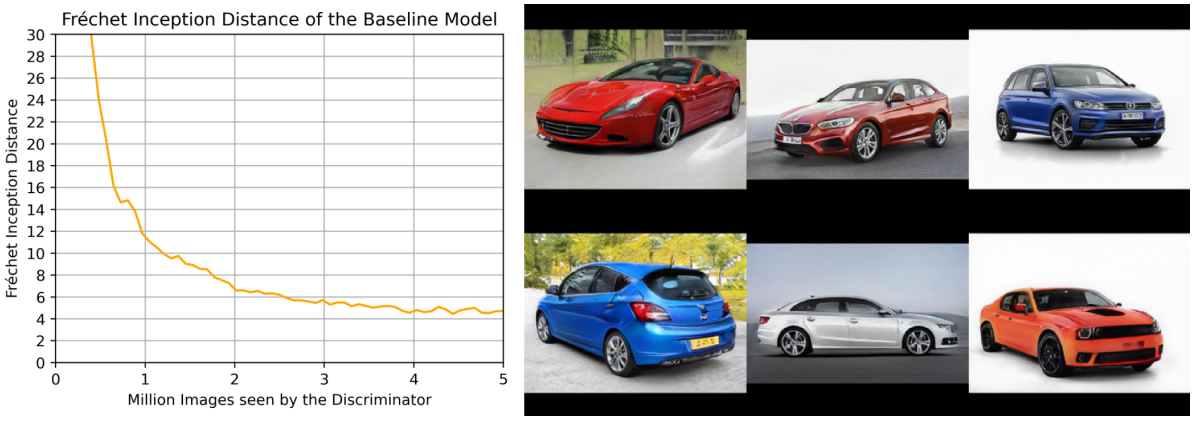


Figure 5.3: The FID graph of the baseline training and curated example images.

Poor Quality Examples

Next to the good results from Figure 5.3, some images also look very poor and sometimes even fully collapse. Collapse in this context means that the images show random shapes and colors, which do not correspond to the appearance of a car. Generally, it showed that the image quality decreases, if we either generate with labels that are rare in the dataset (Figure 5.4 left), or if we use label combinations that do not exist in the real world (Figure 5.4 right).

Figure 5.4: Example images with low quality. The images on the left were generated with rare labels from the dataset and the images on the right with new label combinations.

In order to measure the quality of the images that have been generated from random label combinations, we calculate the FID using randomized input labels. While the traditional FID with real labels scored 4.46, we receive a considerably higher FID of 8.2 when using randomized labels. Although we expect this value to be higher, given that the traditional FID uses the real labels to produce a closer image distribution to the real images, such a discrepancy still underlines that the quality of randomized label combinations is significantly worse.

Conditional Accuracy

In addition to the overall image quality, we also evaluate how well the model learned each individual label. To do so, we calculate the conditional accuracy metric from section 5.2 during the training. As shown in Figure 5.5 (left), the conditional accuracy gradually increases over the course of the training. While simple labels such as the image ratio are already learned within few training steps, other labels such as the model or the manufacturer take considerably longer.

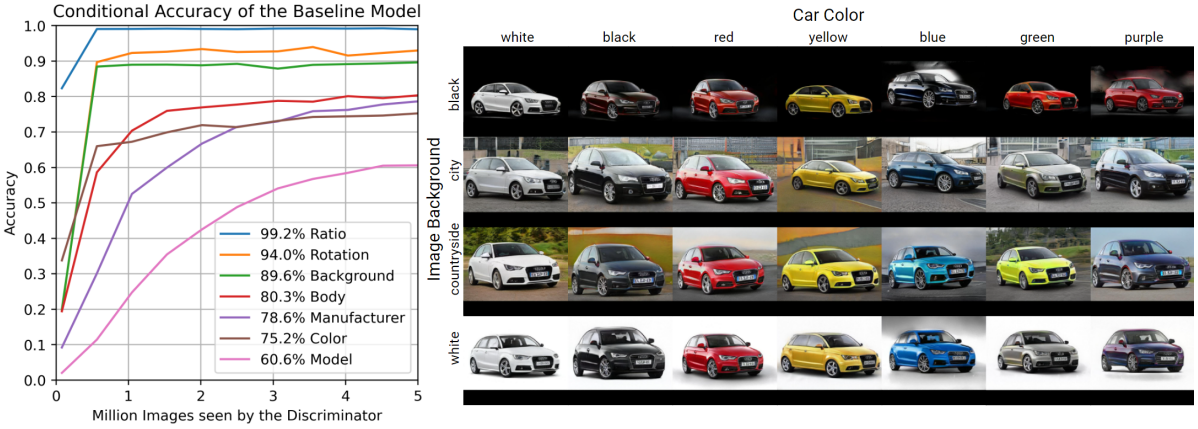
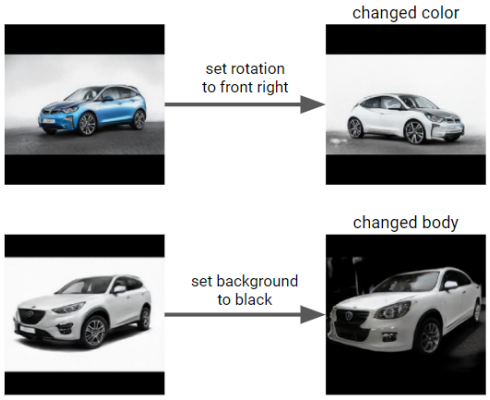


Figure 5.5: The output of the conditional accuracy metric during the training (left) and a matrix of some of the color and background labels (right). It shows that especially green and purple, which are rare colors in the dataset, show incorrect results.

To demonstrate the conditional capabilities of the model, we create a matrix in which we create the same car with different backgrounds and colors (Figure 5.5 (right). It shows that the model has successfully learned most of the labels and is also able to synthesize images from new label combinations. While most colors worked very well, green and purple did not. Instead of green or purple, the model sometimes produced red or silver cars. Given that purple and green are very rare in the dataset, it indicates that less common classes are more difficult to train. It might also explain the low conditional accuracy for the color label, which only scored 75.2%, as shown in Figure 5.5 (left). Another explanation for this low accuracy might also be the lower performance of the classification network, which showed problems differentiating between silver, gray and black.

Label Entanglement

Another problem that arises when generating images with the baseline model, is label entanglement. Label entanglement describes, how much the prediction of a classifier changes if an unrelated label gets changed. An example for this is shown in Figure 5.6, where we change the rotation of a car, which also causes the color to change. To measure this quantitatively, we calculate the label entanglement metric from section 5.3. The result of this metric can be visualized in a matrix as shown in Figure 5.6. In this matrix, each row corresponds to one of the seven classifiers and each column corresponds to the label that is being modified. The diagonal is not filled with values, as we expect the prediction of the classifier to change if a class of the same label gets changed. Generally, a high value states that the classifier prediction changes often when changing a different label, which indicates that those labels are strongly entangled.



changed label

classifier	Model	Color	Manu-facturer	Body	Rotation	Ratio	Back-ground	Mean
Model		53.8	83.4	63.3	56.4	51.7	55.8	60.8
Color	35.6		31.6	33.2	31.9	32.9	36.4	33.6
Manufacturer	51.4	36.1		41.1	34.8	31.1	34.7	38.2
Body	42.6	32.2	33.6		35.8	25.0	30.4	33.3
Rotation	11.0	14.9	12.5	13.6		11.4	11.6	12.5
Ratio	2.7	1.5	2.4	2.3	2.1		13.8	4.1
Background	17.0	18.8	18.2	17.3	17.1	15.6		17.3
							Total Average	28.5

Figure 5.6: Two examples where an image attribute changes when changing an unrelated label (left) and the matrix of the label entanglement metric (right). All values are in %. The total average of all entanglement probabilities is 28.5%

In the entanglement matrix of Figure 5.6, it shows that especially the model prediction changes if the manufacturer label is changed. This was the case in 83% of the times, which indicates that those two labels have the strongest entanglement of all labels. This, however, is expected, considering that the model label always implies the manufacturer label in the real data. The prediction of the manufacturer, on the other hand, did not change as much, when changing the model label. This also makes sense, given that many manufacturers in the dataset have multiple car models. Another intuitive label entanglement is found for the image ratio classifier. It shows very low entanglement values for all of the labels, apart from the background label. The background label changed the ratio prediction in 14% of the times, while the other labels changed it in fewer than 2%. Since both the background label and the ratio label describe similar image regions, it could explain that those have a higher entanglement.

Summary

In Table 5.1, we give an overview of the four metric results. Those will be compared to the following experiments in the next sections.

Experiment	FID ↓	FID Random ↓	Cond. Accuracy ↑	Label Entanglement ↓
Baseline	4.46	8.20	82.4%	28.5%

Table 5.1: An overview of the conditional image synthesis metrics measured on the baseline model. (↓: lower is better, ↑: higher is better)

In summary, it showed that the StyleGAN works very well when it is trained as a Conditional GAN. It manages to learn the semantics of all seven labels from the dataset and produces realistic images. On the downside, however, we observed that the images start to look considerably worse as soon as we generated images from rare labels or with an unseen label combinations. In addition, the labels were also found to be highly entangled. After changing one of the classes, often other unrelated attributes changed as well. In order to avoid those problems, we propose three major approaches, how the performance of the conditional StyleGAN might further improve. Those three approaches are:

- *Changing the Label Sampling*: The first approach proposes two alternatives for the label sampling method during the training to increase the generalization capabilities of the model,
- *Separate Label Mapping*: the second approach tests different methods to combine the label with the latent vector in the generator to separate the random input from the label input,
- *Discriminator with Label Information*: and the last approach provides the discriminator with additional information of the labels, so that it can adapt its feature maps depending on the input label.

5.7 Label Sampling

The first experiment targets the variety of the generated images and in particular the quality of the images when generated with label combinations that are not well represented in the dataset. Especially when training a conditional generative network, it is interesting to see how the generator produces images from label combinations that do not exist in the dataset or even in the real world. With the baseline model, however, we observed that images generated from real label combinations yield a much higher quality than images generated from random label combinations. With the goal of avoiding this problem, we propose different label sampling methods. This means that different label distributions for p_y in the min-max value function are tested.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x,y\sim p_{data}(x,y)}[logD(x,y)] + \mathbb{E}_{z\sim p_z(z),\boldsymbol{y\sim p_y(y)}}[log(1-D(G(z,y),y))] \tag{5.9}$$

In the baseline model p_y is set to p_{data}, i. e., the generator receives the real labels from the training data. This, however, might bring the risk that the generator reaches a state at which 'it learns to reproduce each training image based on the conditional data input' [Gau14]. For this reason some literature suggests to use random label sampling [Gau14]. Since, however, some labels from the car dataset have a strong connection, such as the car model and the car body, we hypothesize that the training performance could reduce if all labels were sampled at random. For this reason, we propose two alternative sampling methods that are designed to increase the generalization capabilities of the model, while also maintain some of the relationships between the labels. Both sampling methods are separately introduced in the following two sections, along with their training results.

5.7.1 Soft Label Randomization

The first sampling method for this experiment is soft label randomization. This method aims to combine the random label sampling with the label sampling from the training dataset. To do so, we first draw the labels from the dataset and then randomly replace some classes with a random class. This is done with a probability of $\rho = 25\%$. By doing so, we replace $7 \cdot 0.25 = 1.75$ of the seven labels in the concatenated label vector on average. With this randomization, we produce a considerably larger variety of label combinations, while still retaining some information about the relationships between the labels.

Training Results

After Training the StyleGAN with the soft label randomization method for five million training images, it showed that the FID decreased considerably slower than the baseline

model. While the baseline model reached a FID of 4.46 after five million images, the model with the randomization only measured a FID of 7.63. This suggests that the image quality is much worse with the soft label randomization method.

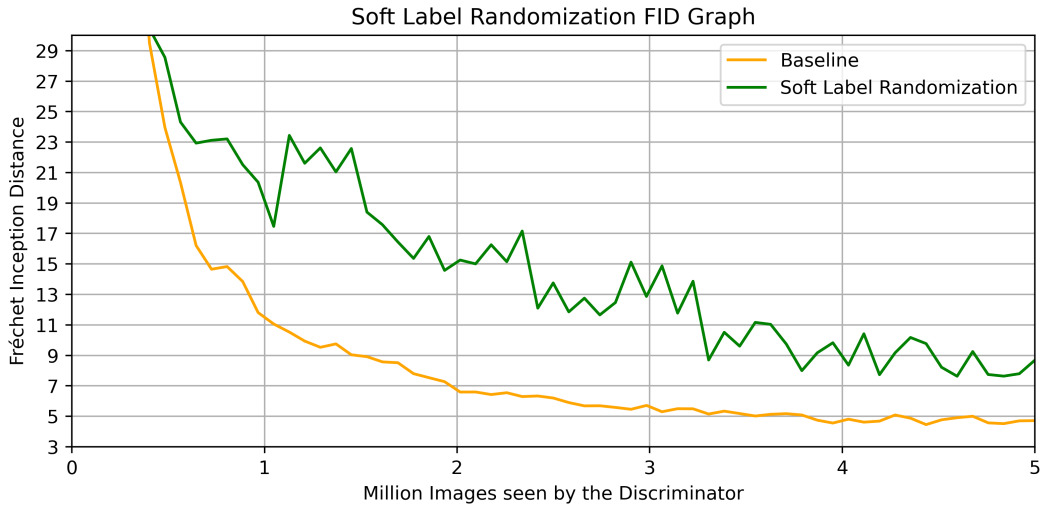


Figure 5.7: The FID graph over the training of the model with soft label randomization.

An even larger difference, however, can be found when synthesizing some example images. This is done in Figure 5.8, which underlines that the images look relatively good when using the label combinations from the dataset (first row), but immediately start to collapse as soon as one of the labels gets replaced by a random label (second row).

Figure 5.8: Example images generated by the model with soft label randomization. The left images were generated with label combinations from the training data and the right images with label combinations, where one class is randomized.

An explanation for this effect might be that the generator now has to simultaneously solve two tasks with very unbalanced difficulty. On the one hand, it synthesizes images from real label combinations and on the other hand it synthesizes images from random label combinations that do not exist in the dataset. Since the discriminator only receives real label combinations for real images from the training data, it is more likely that the generator is able to trick the discriminator with fake images that were generated from real label combinations. If that happens, the generator then receives a positive feedback from the discriminator, which helps to improve the images generated by the real labels,

even further. This could lead to a situation in which the generated images with real labels improve faster than the images with random labels. At the same time, as the discriminator also improves at detecting real and fake images, it then easily detects the fake images from random labels combinations. As a result, the generator then receives large loss values for the random labels, which eventually causes the images to collapse. This would explain why the images, generated by real label combinations from the dataset still produce realistic outputs, while the images with random labels collapse. It also explains why the FID metric is relatively low, although many images only show random outputs. This is because the FID exclusively uses the labels from the dataset. This is changed in the fourth proposed metric, which calculates the FID on randomized labels. While this metric scored a FID of 8.20 for the baseline model, it now scores a FID of 224.49. This value is considerably higher and underlines that almost all of the images have collapsed, when using randomized label combinations.

Figure 5.9: Generator loss graph over the training of the baseline model and the soft label randomization model. It shows that the loss is much more stable in the baseline training.

This theory would also explain the loss graph of the generator over the training (Figure 5.9). While the generator loss is very consistent in the baseline model, it is much more unstable when training with randomized labels. This might be because the loss reaches very low values if the current training batch contains a lot of real labels from the dataset labels, while it gets very high if most of the labels are randomized. Contrary to other machine learning algorithms, a low loss in the generator does not indicate a good training performance. Instead, with GANs, it is aimed to reach an equilibirum, where the generator and the discriminator do not further optimize. Therefore, the lower loss values from the label randomization training do not indicate any improvement compared to the baseline model.

In summary, this experiment showed that it can be very risky to provide the generator with different labels than the discriminator. To avoid this, we propose an alternative

approach in the next subsection that keeps the balance between both models while still increasing the variety of label combinations during the training.

5.7.2 Label Dropout

The second sampling method aims to increase the variety of label combinations by randomly removing some of the labels from the dataset during the training. This way, more distinct label combinations are created while both the generator and the discriminator still receive the same label distribution. This was not possible with the label randomization without creating inconsistent training data. With this experiment, we expect that each label is trained more independently to the other labels, which might improve the quality of the images when they are generated with random labels and also the conditional accuracy. Similar to the previous experiment, we remove 25% of the seven labels in each label vector. This means that the corresponding label vector segment will be filled with zeros.

Training Results

While the previous experiment with the soft label randomization caused the images to collapse for any new label combinations, the label dropout method showed a very good performance. Compared to the baseline model it even improved the lowest FID from 4.46 to 4.18, as shown in Figure 5.10.

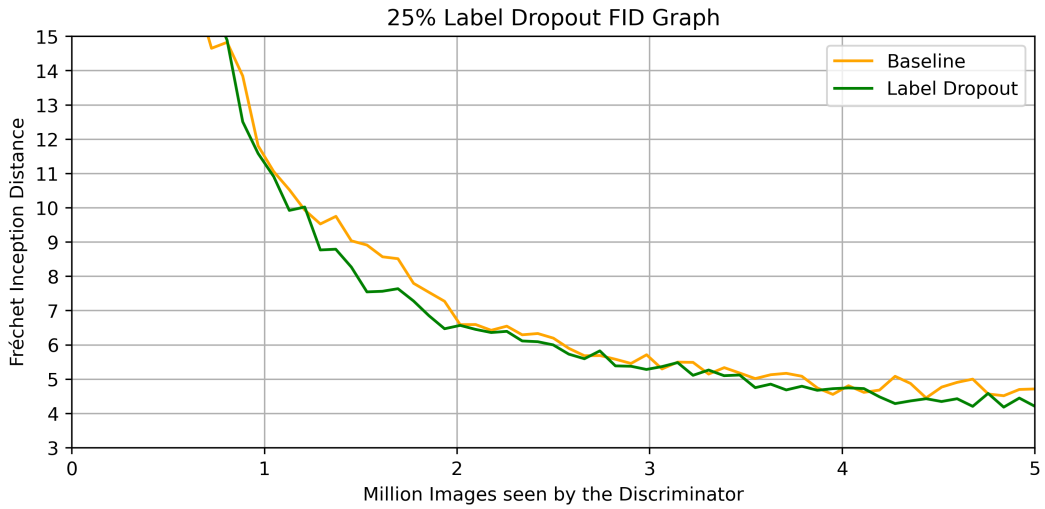


Figure 5.10: FID over the training of the model using label dropout.

Image Quality for unseen Label Combination

The biggest difference, however, is found when calculating the FID with unseen label combinations. The label dropout experiment then scored a FID 6.73, whereas the baseline model only scored a FID of 8.20. This underlines that the images generated from new label combinations, look considerably more realistic compared to the baseline model. This is also demonstrated in Figure 5.11 showing some example images generated from real and randomized labels.

Figure 5.11: Example images generated from real label combinations (left) and unseen label combinations (right).

Conditional Accuracy

Next to the FID metric, we also measure the conditional accuracy. This metric performed slightly worse than the baseline model. This is especially the case for the color and the background labels. While the color label had an accuracy of 75.5% in the baseline model, it only scores 70.1% with the label dropout. The same goes for the background label, which decreased its accuracy from 89.6% to 85.2%. A reason for this might be the number of labels in the dataset. It shows that only 41% of the images have a color label and only 59% have a background label. This is relatively low compared to the other labels, which have a coverage of over 95%. If the labels that already have a low coverage get reduced even more by the label dropout, it then increases the training time for those labels. A possible solution for this problem might be to set the label dropout probability relative to the dataset coverage of each label.
Even though the body label also only has a coverage of 53%, it is hypothesized that this label is still learned well, given that it is semantically connected to the car model and manufacturer.

Experiment	Model	Color	Manufact.	Body	Rotation	Ratio	Backgr.	Average
Baseline	**61.3%**	**75.6%**	78.4%	80.4%	92.9%	**98.9%**	**89.6%**	**82.4%**
Label Dropout	60.1%	70.1%	**81.8%**	**80.9%**	**93.4%**	98.2%	85.2%	81.4%

Table 5.2: The results of the conditional accuracy for the baseline model and the label dropout experiment. It shows that especially the color and the background accuracy decreased with the label dropout method. (higher is better)

Label Entanglement

Although the average conditional accuracy did not improve with the label dropout method, it showed that the label entanglement slightly decreased compared to the baseline model. This is visualized in Figure 5.12.

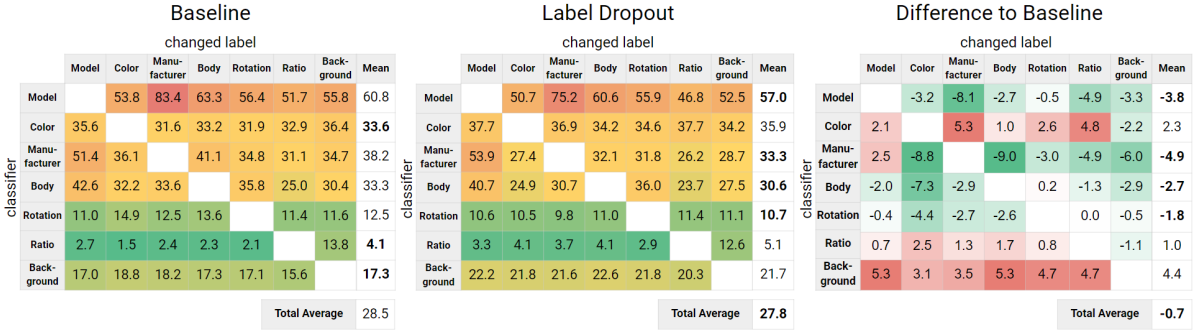


Figure 5.12: The entanglement matrices for the model with the label dropout, compared to the baseline model. Each row represents the classifier and each column the modified label. It shows that the model, the manufacturer, the body and the rotation labels are less entangled using the label dropout. All labels are in %. (lower is better)

There, it shows the label entanglement matrix of the label dropout method compared to the baseline model. The matrix on the right also highlights which dependencies between the labels have increased or decreased the most. This underlines that especially the high entanglement between the model and the manufacturer decreased. Although this value is sill the overall highest, it is considerably smaller with only 75.2% compared to 83.4%. Similar results were also observed for the manufacturer, the body style and the rotation label. On the downside, however, we observe a higher entanglement for the background and color labels. Given that those two labels also reduced in conditional accuracy the most, we hypothesize that a low conditional accuracy implies a high entanglement. Nevertheless, the total average of all probabilities combined still decreases by 0.7%. This indicates that the label dropout helps to decrease the dependencies between the labels.

Training with 50%

Given that this experiment improved both FID metrics and the label entanglement metric, we consider this experiment as successful. For that reason, we perform another experiment with a higher dropout probability. Instead of 25%, we now test 50%.
This training, however, performed very poorly. For the first two million training images, the FID did not decrease below 30, at which point other experiments were already below 10. A reason for this might be that the model receives too few real label combinations from the dataset. This is because, only with a probability of $(1 - 0.5)^7 \approx 0.0078$, non of the labels get removed in a training example. Since the FID is calculated with the label combinations from the dataset, the generated images then look very unrealistic, if

they rarely occur during the training. The FID graph over the training with 50% label dropout is shown in Figure 5.13.

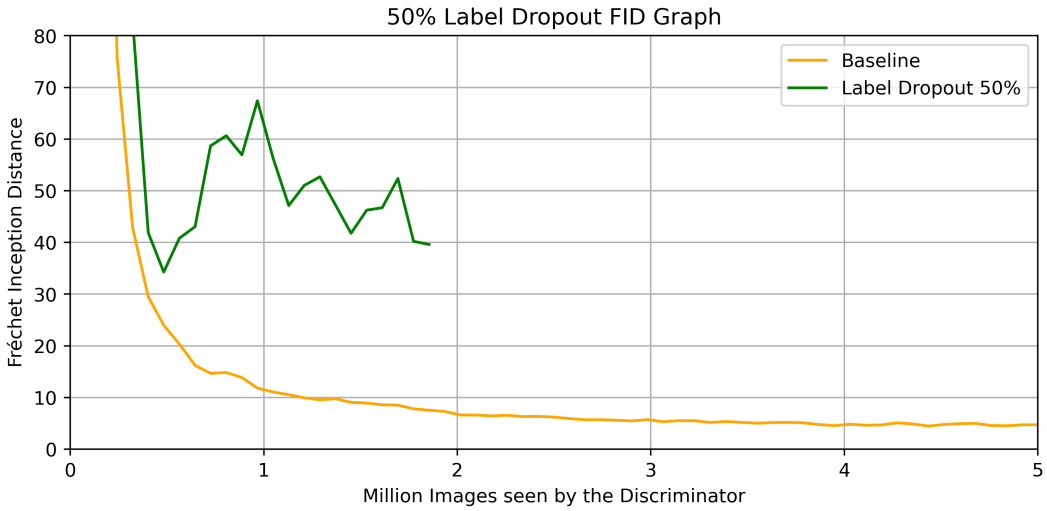


Figure 5.13: The FID graph over the training, when using a label dropout probability of 50%. It shows that the FID does not decrease below 30 after about two Million iterations, which is why the training was interrupted. (lower is better)

Summary

Altogether, the results of the label dropout sampling method showed that the model generally benefits from randomly removing some of the labels. As shown in Table 5.3, this component improved both FID values and the label entanglement. We hypothesize that the decreased conditional accuracy might be explained by the low dataset coverage for the color and background label. Therefore, it might be a good idea to adapt the label dropout probability for each label individually. We hypothesize that this could even improve the conditional accuracy, given that we measured a higher accuracy for the manufacturer and the rotation labels when using this sampling method.

Experiment	FID ↓	FID Random ↓	Cond. Accuracy ↑	Label Entanglement ↓
Baseline	4.46	8.20	**82.4%**	28.5%
Label Dropout (25%)	**4.18**	**6.73**	81.4%	**27.8%**
Label Dropout (50%)	39.60	-	-	-

Table 5.3: An overview of the conditional image synthesis metrics of the label dropout experiment. (↓: lower is better, ↑: higher is better)

5.8 Separate Label Mapping

The next experiment focuses on the method of how the labels are integrated into the latent space. Currently, in the baseline model, the label is first multiplied by a trainable tensor T and then concatenated with the input latent vector z [OS19] [Stya]. The resulting vector is then forwarded through the mapping network to the disentangled latent space W, after which it is used to modulate the weights of the convolutional layers. Although this method has proven to be working well, as shown in the baseline experiment, we test some alternatives to this approach. Instead of combining the label with the latent vector before the mapping network, we experiment how the model performs if the labels are mapped separately to the latent vectors and then combined afterwards in W. This approach is similar to the GAN-Control paper, published in 2021 by Shoshan et al. [Sho+21], which uses a separate mapping network for each individual label and then concatenates each mapped label vector. However, instead of using seven individual networks for each of the labels, we use one combined mapping network. This label mapping network then receives the 127 dimensional concatenated label vector that contains all seven labels at once and maps it to a 512-dimensional vector. An overview of the modified network architecture is visualized in Figure 5.14, showing one mapping network for the latent vector and one for the label. After both the label and the latent vector have been mapped to the disentangled latent space, they are combined and forwarded to the weight modulation component. The method for this combination can range from simply adding or multiplying to more complex operations such as a dense layer. The two combination methods that we test are adding and concatenating. Due to the significantly lower dimensionality of the label vector compared to the 512-dimensional latent vector, the label mapping network only uses 128 neurons, instead of 512, in each of the eight fully connected layers. Apart from that, both network architectures are the same.

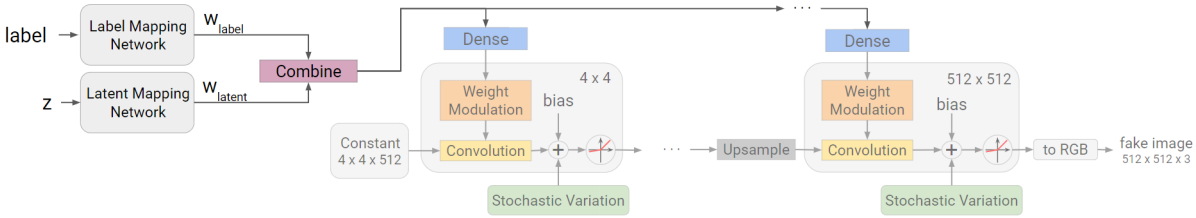


Figure 5.14: The StyleGAN generator architecture with the separate mapping. For the combination method adding an concatenating are tested.

With this modification, we hypothesize that the model might benefit from the additional label mapping network, since it should be easier for the the generator to differentiate the label inputs from the random latent inputs. By doing so it might facilitate a better organization of the latent space W, which in turn could improve the image quality. This is because, according to Karras, Laine, and Aila, "it should be easier to generate realistic images based on a disentangled representation than based on an entangled representation" [KLA18a]. In the following, we evaluate the experiments with both combination types, adding and concatenating, simultaneously.

Training Results

After training the conditional StyleGAN with both variants, it showed similar FID results compared to the baseline model. While the experiment, in which the two mapped vectors are added, performed slightly worse in FID, the experiment with the concatenation method actually improved the FID. This is shown in Figure 5.15, which displays the FID over the training of both models, compared to the baseline model. This hints that the model using the concatenation method slightly benefits from the additional label mapping network.

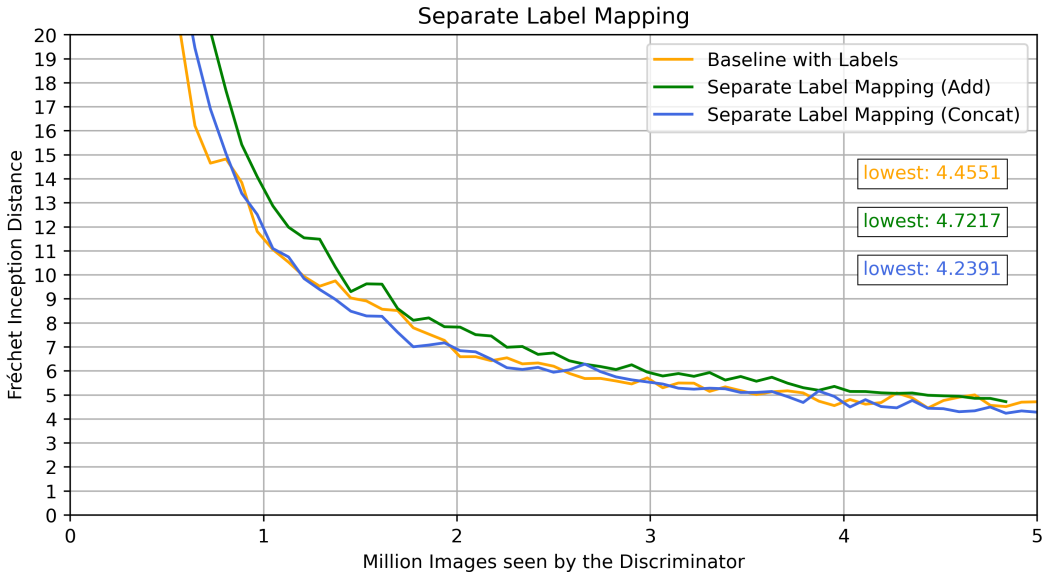


Figure 5.15: The FID results over the training of the models with the separate label mapping network. It shows that the concatenation method slightly improved the FID compared to the baseline model, while the adding method performed consistently worse. (lower is better)

FID with Randomized Labels

A more significant difference between those models and the baseline model appears when calculating the FID with randomized label combinations. There, the baseline model only scored a randomized FID of 8.20, while both new models improved this value, scoring a FID of 6.94 with the concatenation method and a FID of 6.96 with the addition method. Both values are significantly lower, which suggests that the separate mapping network increases the generalization capability and also possibly the robustness against creating collapsed images. Such collapsing images were often observed in the baseline model when generating images from randomized label combinations that did not exist in the dataset.

Label Entanglement

Next to the improved quality for images with unseen label combinations, the label entanglement decreased in both experiments, as well. This is visualized for the model with the concatenation method in Figure 5.16.

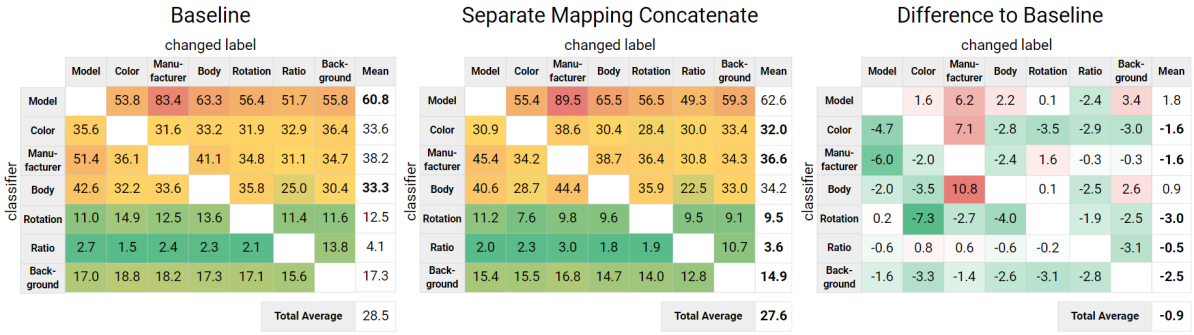


Figure 5.16: The entanglement matrices for the baseline model and the model with the separate label mapping network that concatenates the mapped vectors. Each row represents the classifier and each column the modified label. All values are in %. (lower is better)

It shows that all labels apart from the car model and the car body style decreased their entanglement when using a separate mapping network. Although the entanglement between the body and the manufacturer increased substantially, we observe an overall decrease in entanglement of 0.9%. This underlines that the labels are less connected when using the separate label mapping network. A similar observation was also made with the model, where the mapped label and latent are added. There, however, the label entanglement only decreased from by 0.5%.

Conditional Accuracy

Although the model has improved its label entanglement and therefore possibly created a more organized latent space, no improvement was observed for the conditional accuracy. Instead, both models even slightly decreased their overall accuracies. This is shown in Table 5.4, where only the rotation and the image ratio score better accuracies when using the separate label mapping. All other accuracies, however, scored slightly worse.

Experiment	Model	Color	Manufact.	Body	Rotation	Ratio	Backgr.	Average
Baseline	**61.3%**	**75.6%**	**78.4%**	**80.4%**	92.9%	98.9%	**89.6%**	**82.4%**
Separate Mapping (add)	56.5%	73.3%	76.5%	77.7%	93.3%	99.1%	88.0%	80.6%
Separate Mapping (concat)	58.9%	74.6%	77.2%	77.4%	**94.2%**	**99.1%**	88.3%	81.4%

Table 5.4: The results of the conditional accuracy for the separate label mapping experiments. It shows that the baseline model performs better for almost all labels.

Summary

An overview of all four conditional metrics, measured on the models with the separate label mapping network is shown in Table 5.5.

Experiment	FID ↓	FID Random ↓	Cond. Accuracy ↑	Label Entanglement ↓
Baseline	4.46	8.20	**82.4%**	28.5%
Separate Mapping (add)	4.72	6.96	80.6%	28.0%
Separate Mapping (concat)	**4.24**	**6.94**	81.4%	**27.6%**

Table 5.5: An overview of the conditional image synthesis metrics of the separate mapping experiments. (↓: lower is better, ↑: higher is better)

Overall, it showed that the model with the separate mapping network especially improved the image quality when synthesizing with random label combinations. In addition to that, also the label entanglement slightly decreased for both models, compared to the baseline model. This hints that the separate mapping helps the model to find a better organization for the labels, in which each label semantic is encoded more independently from the other labels. We hypothesize that the label entanglement might improve even further, if we combine this experiment with the label dropout method from the last section. This will be tested at the end of this chapter, where we combine all successful components in one model. Before that, however, we conduct another experiment, where we test, whether the discriminator possibly benefits from a similar label mapping network as well.

5.9 Label Information in the Discriminator

The next experiment aims to improve the performance of the discriminator by utilizing the information of the labels. Currently, in the baseline model, the discriminator receives an input image and produces an output vector that has the same shape as the label. This output vector is then masked by the one-hot encoded input labels to remove all outputs that do not correspond to the classes of the current label. With this method, the model successfully learns the semantics of each class, as demonstrated in the baseline experiment. We, however, hypothesize that the evaluation of the discriminator might further improve if it were able to use the information about the current label already within the feature maps. The discriminator could then focus on different image attributes, depending on the given labels. This idea is similar to an attention mechanism [Vas+17], which is designed to direct the focus of the model towards specific regions of the image. However, instead of using any information from the previous feature maps, like in a traditional attention model (which has already been applied to GANs [Zha+19]), we proposes to use the information given by labels. To do so, we design a modified discriminator architecture, similar to the StyleGAN generator. In this architecture, the discriminator first maps the input label via a separate mapping

network, like in the previous experiment from section 5.8 and then uses the mapped labels to modulate the weights of the convolution layers, analogously to the StyleGAN generator. To do so, we apply the weight modulation component from subsection 2.2.2. An overview of the modified discriminator architecture is visualized in Figure 5.17.

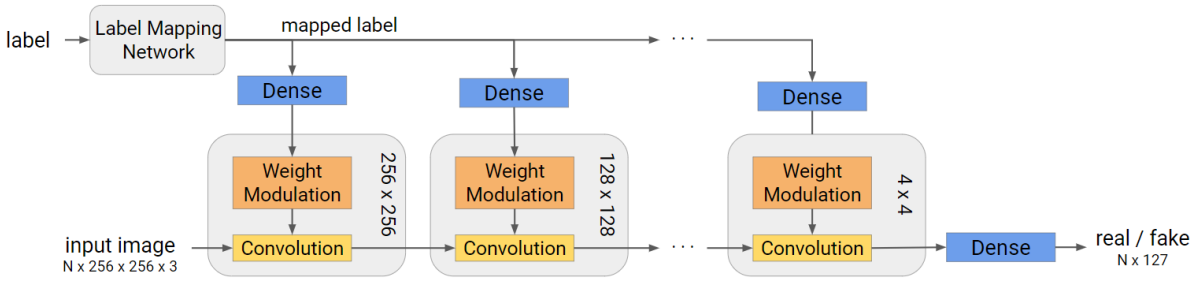


Figure 5.17: The architecture of the modified discriminator that uses an input label to modify the weights of the convolutions with the weight modulation component from the StyleGAN generator.

Training Results

After training the model with the modified discriminator, the FID initially showed very good results. As demonstrated in Figure 5.18, the FID decreases considerably faster when using the additional label information also within the discriminator. Eventually, however, at about 4 million training images, both networks perform very similarly, which is why the both FID scores are very close to each other. Nevertheless, the model with the modified discriminator scored slightly better with a lowest FID of 4.37 compared to the lowest FID of the baseline model, which was 4.46.

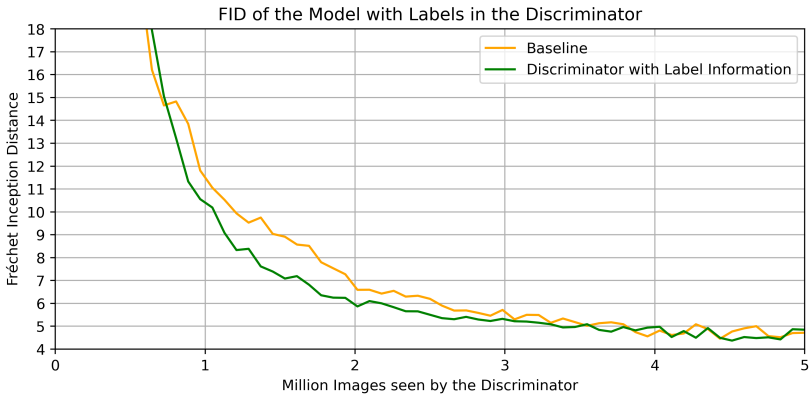


Figure 5.18: FID graph of the training with the additional label information in the discriminator. It shows that the FID decreases much faster at the beginning of the training. (lower is better)

Conditional Accuracy

A possible explanation for this difference in FID at the beginning of the training might be that the model mainly improved its performance for coarse image features. This would explain the results of the conditional accuracy metric, shown in Table 5.6. There, we observe that the model improved its accuracy for the background, the color and the rotation, which are all coarse features that correspond to large areas of the image. Finer details, like the manufacturer or the model, on the other hand, decreased their accuracy considerably. Therefore, also the average accuracy for all labels combined also slightly decreased.

Label	Baseline	Discriminator with Label Info	Difference
Background	89.6%	**91.7%**	+2.1%
Color	75.6%	**77.5%**	+1.9%
Rotation	92.9%	**94.2%**	+1.3%
Ratio	98.9%	**99.1%**	+0.2%
Body	**80.4%**	78.7%	-1.7%
Manufacturer	**78.4%**	74.6%	-3.8%
Model	**61.3%**	55.5%	-5.8%
Average	**82.4%**	81.6%	-0.8%

Table 5.6: The conditional accuracy of the model trained with the additional label information in the discriminator.

Feature Map Analysis

Table 5.6 showed that the accuracy for the background label improved the most. A possible reason for this can be found when inspecting the feature maps of the discriminator using the Grad-CAM method [Sel+17]. This method, introduced by Selvaraju et al., visualizes the feature maps by calculating a linear combination of all channels based on their average gradient to an output neuron. This way it can be inspected, which regions of the image have the biggest influence on the value of an output neuron. Since the StyleGAN discriminator also uses one output neuron for each class, this method can also be applied here. To do so, we first input a real image combined with a label into the discriminator. After that we calculate the gradient from the output neuron that is responsible for the background output to the feature maps. Then, we calculate the channel wise average of each gradient, which is used to calculate a linear combination of all feature maps. This means that we multiply each feature map with its average gradient and then calculate the sum. The output of that can then be used as a heatmap as shown in Figure 5.19.

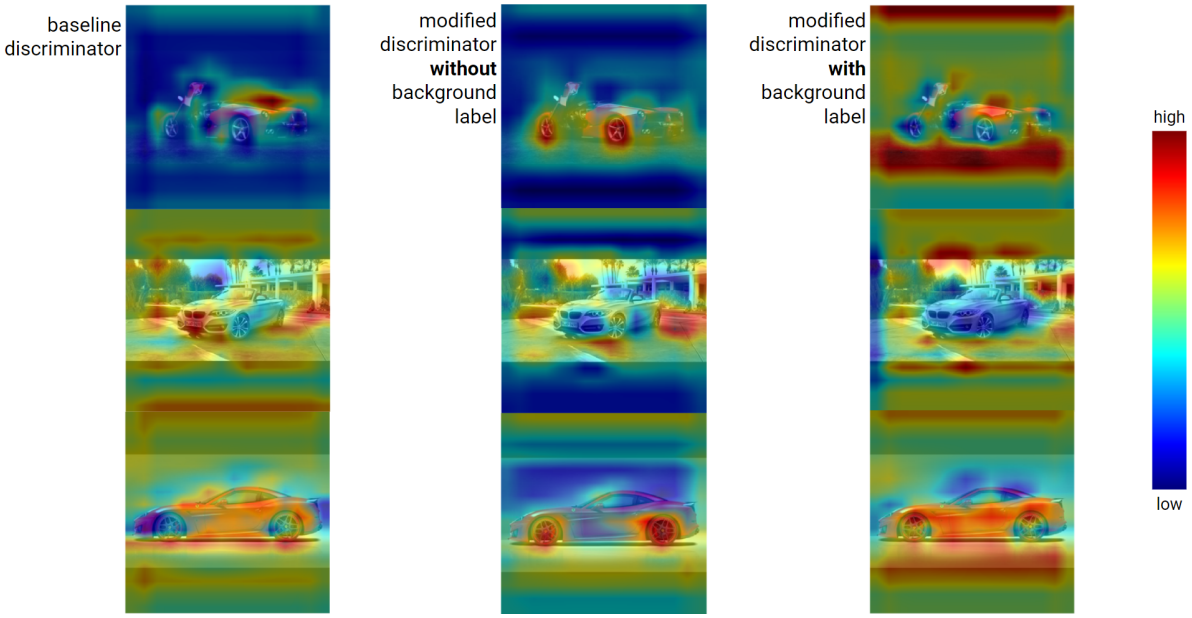


Figure 5.19: The Grad-CAM visualization for the discriminator
Left: The Grad-CAM visualization for the baseline discriminator. Center: The Grad-CAM visualization for the modified discriminator without a background label. Right: The Grad-CAM visualization for the modified discriminator with a background label. In this visualization, regions of the background are highlighted the most.

There, it shows the Grad-CAM visualizations for three example images, using the gradient from the feature map of the 16×16 layer of the discriminator to the background neuron. The first column shows the Grad-CAM visualization using the baseline discriminator, the second column shows the visualization for the modified discriminator, however, without a background label and the third column shows the visualization for the modified discriminator with a background label. It shows that the last column, where we use the modified discriminator with a background label, highlights the background the most, while also removing some focus from the car. This indicates that the discriminator utilizes the background label to modify the convolution filters so that more regions in the background have a higher influence on the output.

Although the examples from Figure 5.19 might look convincing, this observation was not consistent for all images. This is likely because the discriminator sometimes predicts the background of a real image as fake, in which case we observe a low activation with the Grad-CAM method. Nevertheless, it gives an idea of how the discriminator changes its activation based on the input label. However, in order to measure the effect of the additional label input for the feature maps more quantitatively, we forward a large number of real images to the discriminator and calculate the average intensity difference in the feature maps with and without the label input. By doing so, it underlines which labels cause the largest intensity change in the feature maps. This is done for each layer individually, so that we can also observe at which layers the additional label has the

biggest influence. The result of that analysis is shown in Figure 5.20. There, each cell corresponds to the average feature map intensity difference of 1000 real images with and without the label input. It highlights that the intensity is changed much more in the layers that have a smaller resolution. This, again, hints that the label information is mostly used to improve coarse image features. In addition to that, it also shows that the background label causes the larges intensity difference of all labels. This correlates with the increased conditional accuracy (Table 5.6) for the background label.

	Model	Color	Manufacturer	Body	Rotation	Ratio	Background
Layer 256 x 256	1.1	1.1	1.1	1.1	1.1	1.0	1.1
Layer 128 x 128	9.7	10.1	10.0	9.7	9.7	10.0	10.9
Layer 64 x 64	12.4	12.7	13.1	12.3	12.4	12.5	14.4
Layer 32 x 32	23.7	24.5	25.0	23.1	24.4	23.3	28.5
Layer 16 x 16	46.3	47.0	49.5	44.9	49.9	44.9	56.6
Layer 8 x 8	75.4	74.7	77.0	72.9	75.7	72.7	88.6
total proportion	13.86%	13.99%	14.45%	13.49%	14.24%	13.52%	**16.45%**

Figure 5.20: The average intensity difference for each layer in the discriminator when removing a label. All values are multiplied by 1000.

Summary

Altogether, it showed that the additional label information in the discriminator helps the model to produce better coarse attributes such as the image background or the rotation of the car. This was shown with the conditional accuracy metric that improved for the background and rotation label but decreased for the car model and manufacturer label. It especially improved the FID at the beginning of the training and also lowered the overall FID. In addition to that, also the label entanglement slightly increased. We, however, hypothesize that this is connected with the decreased conditional accuracy.
Given that the modification brings some additional complexity to the model, it might benefit from a longer training. Especially, because we observed that all models first focus on coarse attributes and later on finer attributes. This could mean that this component might also benefit the smaller image features, if we use a longer training duration. However, in order to test this, we would also have to increase the length of all other experiments as well, so that the experiments are still comparable to each other. Therefore, we will not increase the training length for this experiment.

Table 5.7 shows the conditional metric results for the experiment with the additional label information in the discriminator.

Experiment	FID ↓	FID Random ↓	Cond. Accuracy ↑	Label Entanglement ↓
Baseline	4.46	8.20	**82.4%**	**28.5%**
Discriminator with Label Info	**4.37**	8.20	81.6%	29.0%

Table 5.7: An overview of the conditional image synthesis metrics measured on the baseline model. (↓: lower is better, ↑: higher is better)

5.10 Combination Experiment

In a last experiment we test how well the proposed modifications work together. To do so, we select the best configuration of each experiment and combine them into one model. Those modifications are the label dropout component, the separate label mapping with the concatenation method, and the additional label information in the discriminator. To give a brief overview of their results, Table 5.8 summarizes the metrics for each of the three experiments compared to the baseline model. Generally, it shows that both FID results have improved in all experiments, while the conditional accuracies decreased. The label entanglement, on the other hand, improved for the label dropout and the separate mapping, while it scored worse for the discriminator with label information.

Experiment	FID ↓	FID Random ↓	Cond. Accuracy ↑	Label Entanglement ↓
Baseline	4.46	8.20	**82.4%**	28.5%
Label Dropout (25%)	**4.18**	**6.73**	81.4%	27.8%
Separate Mapping (concat)	4.24	6.94	81.4%	**27.6%**
Discriminator with Label Info	4.37	8.20	81.6%	29.0%

Table 5.8: An overview of the conditional image synthesis metrics all successful components from the last sections. (↓: lower is better, ↑: higher is better)

Training Results

After training the model with all three components combined, it showed very good but also very poor results. Very good results can be found when inspecting the FID graph over the training in Figure 5.21. There it shows that the training reaches considerably lower FID results compared to the baseline model. While the baseline model scored a lowest FID of 4.46, the combination model now reaches a FID of 3.95. This is also lower than any other previous experiment in this thesis, underlining that the image quality significantly improves when using all components at once.
The same holds true for the FID that is calculated with randomized label combinations. There, the combination model scored a 5.50 FID, which is also an overall best for this thesis. This indicates that the model is less likely to collapse when it is given a label combination that did not exist in the dataset. This is can also be observed when generating example images. As shown in Figure 5.22, the image quality is considerably better when sampling unseen label combinations with the combination model. For comparison, we use the same label combination in each column in Figure 5.22 and generate two images with each model. In the third column it shows the biggest quality discrepancy. There, we generate a car with the manufacturer 'Opel', the model 'Audi A3' and the body 'sports car'. This results in a fully collapsed image for the baseline model, while we observe a relatively good quality for the combination model. Although, some bad examples can also be found in the combination model, we observe that the images are consistently better than in the baseline model.

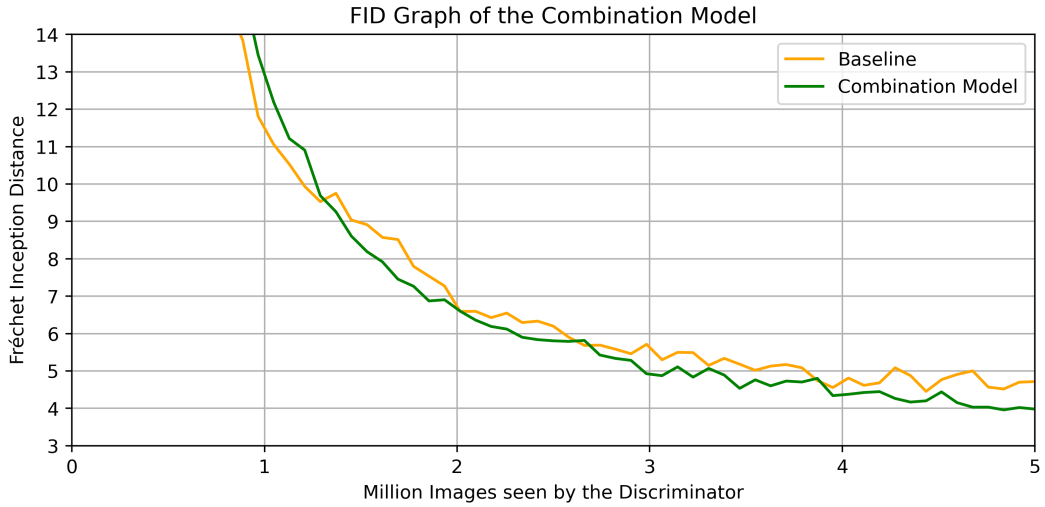


Figure 5.21: FID Graph of the combination training compared to the baseline model.

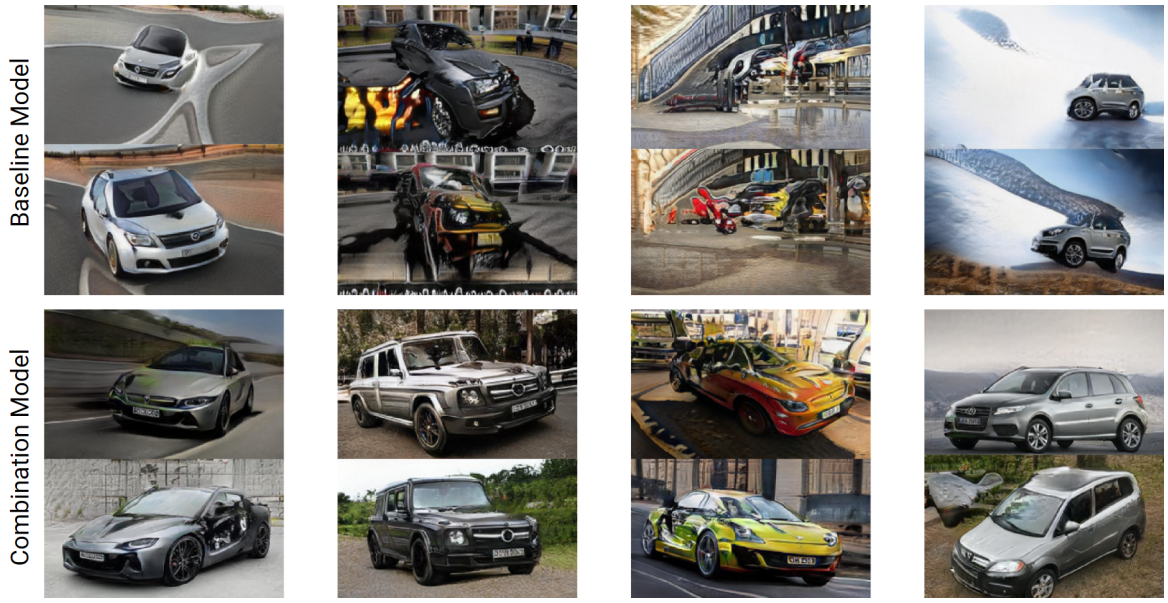


Figure 5.22: Example images from the baseline model (first row) and the combination model (second row). The label combinations are the same in each column and did not exist in the dataset.

On the downside, however, the conditional accuracy scores much worse results. This is shown in Table 5.9, where the accuracy is lower for every label when using the combination model. This might indicate that the model improves the image quality at the cost of its conditional capability.

Experiment	Model	Color	Manufact.	Body	Rotation	Ratio	Backgr.	Average
Baseline	**61.3%**	**75.6%**	**78.4%**	**80.4%**	**92.9%**	**98.9%**	**89.6%**	**82.4%**
Combination	44.9%	69.1%	71.8%	76.2%	91.5%	98.0%	85.6%	76.7%

Table 5.9: The results of the conditional accuracy for the baseline model and the combination experiment. It shows that all accuracies are consistently lower in the combination model. (higher is better)

We believe that this is a rather fundamental problem of the model, which is related to our conditional loss function in combination with the input label vector. In section 5.5, we defined the output of the discriminator as:

$$D = \sum_{i}^{127} D'_i \cdot y_i \tag{5.10}$$

Here, we use the input label as a mask for the output neurons of the discriminator (D') and calculate the sum. After that we apply the sigmoid function and forward the result to the loss function. Although this method has shown good conditional results in the previous experiments, it might bring a major disadvantage, since it allows the model to exclusively improve over few labels. This is because we calculate the sum of all discriminator outputs. Therefore, it might enforce the generator to produce a very good feedback in the discriminator for simple image features such as the image ratio or the background, while it neglects more difficult labels such as the car model. This in turn would increase the conditional accuracy for some labels but decrease it for the others. In the combination experiment, however, we observed that none of the accuracies have improved. This likely has to do with the 'real / fake' label that we append to the input label. As described in section 5.5, we add a fixed 1-entry to the label vector, so that the discriminator can decide whether a image is real or fake independently to the given labels. This was done, since the seven labels from the car dataset do not cover all image features. We, however, hypothesize that this 'real / fake' label, in combination with the sum of the discriminator outputs, causes the network to mainly improve over the image quality task, while the conditional task gets neglected.
This theory is also supported when inspecting the average activations that the generator causes in the discriminator. This is shown in Table 5.10. While the output for the 'real / fake' neuron is at 0.33, all other outputs are negative. This indicates that the generator has the highest success for this output neuron compared to the others. Therefore, we assume that the 'real / fake' label dominates the parameter optimization for the generator.

Real / Fake	Model	Color	Manufact.	Body	Rotation	Ratio	Backgr.
0.33	-0.96	-0.24	-0.77	-0.94	-1.68	-2.29	-0.91

Table 5.10: The average discriminator output for 100 fake images before calculating the sum. It shows that the real / fake entry causes the largest activation in the discriminator.

Summary

Altogether it showed that the combination experiment improved the image quality considerably. Especially when synthesizing images from random label combinations, we observe a consistently better image quality compared to the baseline model. This improvement, however, comes at the cost of the conditional accuracy. We hypothesize that this is due to the formulation of the loss function in combination with the 'real / fake' label. Although this issue should have also affected all previous experiments, we believe that the combination of all modifications amplifies this problem. Therefore, we assume that the modifications from this chapter, in general, help the model to optimize its parameters, even though the loss function might not be suitable. Nevertheless, given that this model produces an overall best FID for randomized label combinations, we still consider this experiment as successful. Therefore, we assume that the components should also improve the performance of a model that uses an alternative loss function. Since this problem was found at a very late stage of the experiments, we will not repeat all of the experiments with a changed loss function. For the same reason, we will also use this loss function for the experiments in chapter 7. There, however, this should not affect the results as much, given that we focus on synthesizing a 3D rotation, for which we will design an additional loss.

5.11 Summary

In this chapter, we performed a total of eight trainings to test the performance of the three main approaches. Those three approaches were: A modified label sampling method during the training, a separate label mapping network in the generator, and a discriminator modification that allowed to use the information of the labels in the feature maps.

- The first approach showed that the model generally benefits from additional variation in the label combinations, although it is detrimental to provide different labels for fake images and real images. In addition to that we also observed that the model does not converge if we remove too many labels in order to create more variation.
- The second approach underlined that a separate label mapping network can increase the image quality for randomized label combinations and also decrease entanglement between the labels.
- The last approach showed that additional label information in the discriminator can help synthesizing coarse image features, which reduces the FID especially at the beginning of the training.

After that, we also performed an experiment in which we combined all modifications. This experiment scored the best FID results for both real and randomized labels. It was especially observed that the generated images were considerably less likely to collapse compared to the baseline model. Therefore, we conclude that the combination of all three components helps the model to significantly improve the image quality. On the downside, however, the conditional accuracy decreased in this experiment. We hypothesize that this is because of the formulation of the loss function in combination with the 'real / fake' label. For future work, we will test how the model performs with alternative loss functions. For the rest of this thesis, however, we will keep using the same loss function. A summary of all results is shown in Table 5.11. This, again, underlines that we receive the best image quality with the combination model, although the conditional accuracy decreases.

Experiment	FID ↓	FID Random ↓	Cond. Accuracy ↑	Label Entanglement ↓
Baseline	4.46	8.20	**82.4%**	28.5%
Label Randomization (25%)	7.63	224.59	-	-
Label Dropout (25%)	4.18	6.73	81.4%	27.8%
Label Dropout (50%)	39.60	-	-	-
Separate Mapping (add)	4.72	6.96	80.6%	28.0%
Separate Mapping (concat)	4.24	6.94	81.4%	**27.6%**
Discriminator with Labels	4.37	8.20	81.6%	29.0%
Combination	**3.95**	**5.50**	76.7%	30.3%

Table 5.11: The metric results of all experiment in this chapter. (↓: lower is better, ↑: higher is better)

In the following chapters, we will test if we can utilize the rotation label in order to create a 3D rotation of a car. This is motivated by the observation from section 4.3, where we were able to rotate a car using the main components of the PCA. Given that we now have a considerably better control over the car rotation using the conditional StyleGAN, we test how well we can utilize the discrete rotation labels to synthesize a continuous 3D rotation. Before that, however, we will first give a brief overview of other approaches from literature to create 3D models with GANs.

6 3D Image Synthesis with GANs

The idea of synthesizing 3D objects with GANs has become increasingly popular in recent years. Numerous research groups have proposed various methods to achieve this. To give a brief overview of the manifold methods, this chapter introduces three main approaches from literature.

6.1 3D Output Space

The first approach aims to synthesize 3D objects by training a GAN that outputs a 3D voxel space. One example for this approach is the 3D-GAN introduced by Wu et al. [Wu+17]. The 3D-GAN utilizes 3D training data, such as the ShapeNet [Cha+15] or the IKEA dataset [LPT13], to train a GAN that outputs a 3-dimensional object. In order to do so, the generator uses 'five volumetric fully convolutional layers of kernel size $4 \times 4 \times 4$' [Wu+17] that map a random latent vector z to an output space with the shape 64 x 64 x 64, as shown in Figure 6.1. The discriminator of the 3D-GAN has a mirrored architecture to the generator and classifies whether the given 3D objects are real or fake.

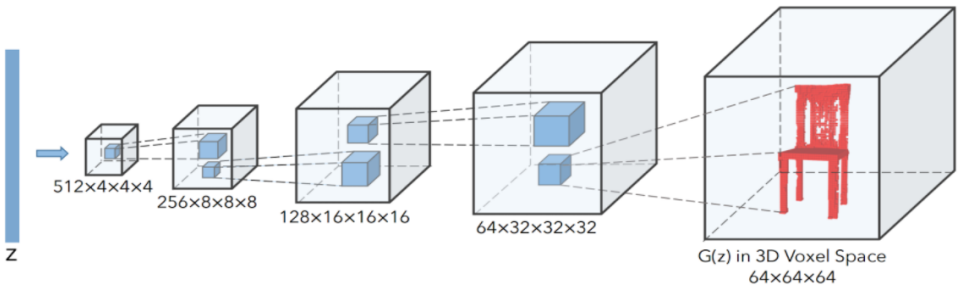


Figure 6.1: The architecture of the 3D-GAN generator that maps a random latent vector to a 3D voxel space. (image source: [Wu+17])

While the 3D-GAN and other similar architectures have shown some success in generating 3D objects, they still face of two major problems. The first problem is that they require 3D training data, which is very limited and often based on simulations (e.g., the CARLA dataset [Dos+17]). And the second problem is the high computing time and memory cost to train the model. This is because of the expensive volumetric convolutions and their 3D feature maps. For this reason, those methods can not be applied to state of the art high resolution datasets.

6.2 Internal 3D Representation

The second approach is similar to the first, as it, again, maps a random latent vector to a 3D voxel space within the generator. Although, instead of forwarding the generated 3D models to the discriminator, a differential projection module is used, to project the 3D representation to a 2D image from a random viewpoint. The resulting 2D image is then forwarded to the discriminator, which predicts whether it is real or fake. This allows to use 2D training data to generate 3D data, which is a big advantage over the previous approach, given the large amount of available 2D image dataset, compared to 3D datasets. One of the first papers that utilizes this approach is the Projective Generative Adversarial Network (PrGAN) introduced by Gadelha, Maji, and Wang [GMW16] in 2016. Its architecture is shown in Figure 6.2, visualizing how the generated 3D object is projected to a 2D image and then forwarded to the discriminator.

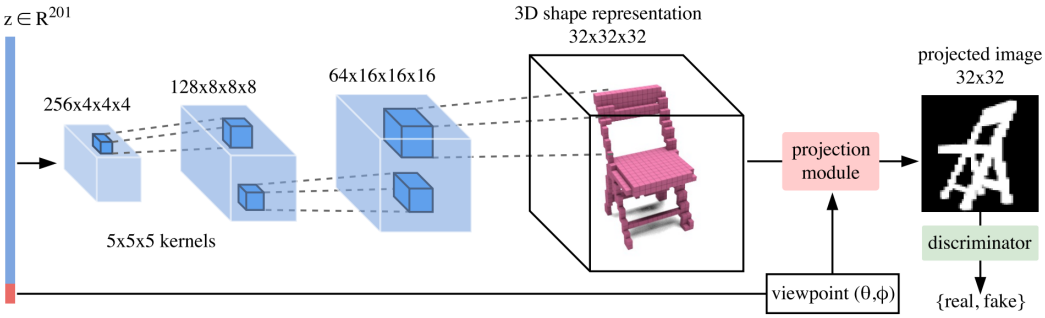


Figure 6.2: The generator architecture of the PrGAN, which synthesizes a 3D object and then feeds a 2D projection of that object to the discriminator. θ and ϕ describe the height and angle of the viewpoint, respectively. (image source: [GMW16])

Although this approach brings a lot of improvements to the 3D image synthesis with GANs, given that any consistent 2D dataset can be used, the problem with long training time and GPU memory consumption persists. For this reason, the method is not applied to high resolution datasets such as the FFHQ dataset [KLA18b], which has a resolution of 1024×1024.

6.3 Alternative Internal 3D Representations

In order to tackle the problem of high computing time and memory consumption, both previous approaches exhibit, some methods in literature propose alternative internal 3D representations. One of those methods is the Efficient Geometry-aware 3D GAN (EG3D GAN) introduced in December 2021 by Chan et al. [Cha+21a]. Similar to both previous approaches, the main idea of the EG3D model is to synthesize an image from an internal 3D representation. This, however, is done without using the expensive 3D convolution

operation. Instead, they design a tri-plane structure of three orthogonally aligned 2D feature maps, as shown in Figure 6.3. This way, any given point in a 3D space can be described by a combination of the values from all three feature maps. Those points are then forwarded to a neural renderer module, where each value combination is decoded and an internal 3D model is produced. This 3D model is then projected onto 2D feature maps, based on extrinsic and intrinsic camera parameters. Since the resulting output feature maps only have a resolution of $128 \times 128 \times 32$, also a super-resolution network is applied, which outputs images at a resolution of $512 \times 512 \times 3$. At last, the resulting high resolution 2D image is then forwarded to the discriminator. In this model, the StyleGAN2 generator and discriminator are used. This is the same version that we use in this thesis.

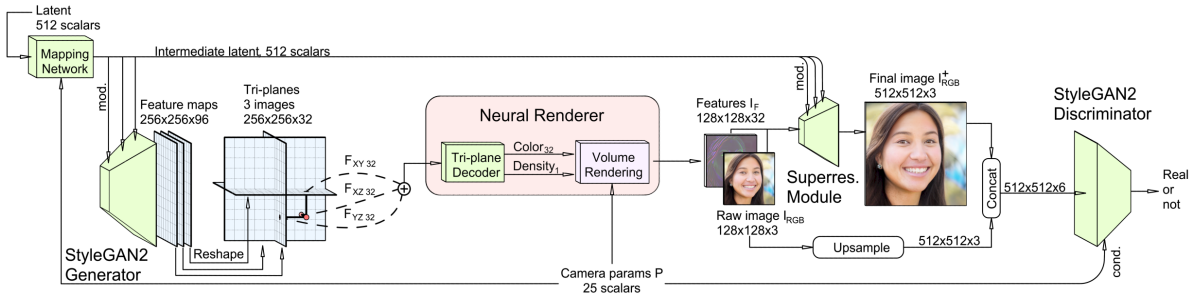


Figure 6.3: The architecture of the EG3D network, consisting of a tri-plane representation of feature maps and a neural renderer module to synthesize 2D images of 3D objects. (image source: [Cha+21a])

The EG3D GAN architecture enables the synthesis of high resolution images with 3D behavior up to 7.8 times faster and with only 6% of the memory consumption [Cha+21a], compared to some previous approaches that used a voxel space. In addition to the computational speed up, it also improves the quality of the output images (measured in FID), compared to similar methods, such as GIRAFFE [NG21] or pi-GAN [Cha+21b]. An example for this is shown in Figure 6.4, demonstrating both the high quality and the ability to rotate the camera.

Figure 6.4: Example images synthesized by the EG3D GAN. (image source: [Cha+21a])

Altogether, it showed that this approach enables the synthesis of state-of-the-art images with 3D properties, while also reducing computing time and memory consumption. The

only downside of this method is that the neural renderer module can be considered a bottleneck for the generator. While the traditional StyleGAN uses 256 feature maps at a 128^2 resolution, the EG3D model only uses 32. This is considerably smaller and could compromise image quality.

6.4 Summary

In this chapter, we presented three methods from literature to synthesize 3D models with GANs. While the first two show significant problems regarding computing time and memory consumption, given that they require expensive 3D convolution operations, the third proposes an alternative internal 3D representation to avoid such problems. The latter successfully produces photorealistic high resolution images with 3D properties in considerably shorter computing time. The only downside of this method is that the neural renderer module poses a bottleneck, as the output feature maps have a significantly fewer channels compared to the traditional StyleGAN. For this thesis, we hypothesize that such a bottleneck can be avoided by using the traditional StyleGAN generator. We believe that a 3D rotation can already be synthesized with the standard StyleGAN, given that we already observed some capabilities of synthesizing 3D-like transformations in chapter 4. The only problems with the proposed methods from chapter 4 were that a specific control over certain image features showed to be very difficult. This was because of the high-dimensional latent space that required an extensive search in order to synthesize images with specific attributes. When training the StyleGAN as a conditional network, however, it is much easier to control the output images, as shown in the previous chapter. This motivates the use of the rotation label from the car dataset to create image transitions with 3D-like behavior. This approach will be tested in the following chapter.

7 360° Rotation View Experiments

In this chapter, we test how well a continuous 360° rotation of a car can be synthesized using only the eight discrete rotation labels from the car dataset. To do so, we propose multiple modifications to the baseline model from chapter 5 and test them in a series of experiments. Before that, however, we first define three characteristics for good 360° rotations.

7.1 Rotation Goals and Metrics

The main target of this chapter is to synthesize a 360 degree view of a car. This means that on the one hand, a car can be synthesized from any given rotation angle and on the other hand, a smooth animation of a rotating car emerges, if the images are sampled in a consecutive order. To further specify this goal, the following three desired characteristics for a generated 360° view of a car are defined:

1. High image quality throughout the whole rotation
2. Consistent rotation speed
3. Correctly synthesizing cars at the input angle

To evaluate how well the following results of the experiments fulfill those characteristics, other than by inspecting the animation of a rotation by hand, we propose four metrics.

7.1.1 Rotation Image Quality

The first rotation metric focuses on the image quality throughout the whole rotation. To do so, the metric calculates the FID for images generated at random rotation angles. Moreover, instead of only sampling discrete rotation angles, like in the dataset, where the images are annotated with multiples of 45°, any continuous angle from 0° to 360° degrees can be sampled. The method for synthesizing such continuous angles is explained in the first two experiments, since this changes depending on the type of experiment. Like in the conventional FID metric, 50000 images are generated and compared to 50000 images of the dataset.

7.1.2 Rotation Linearity

The second rotation metric focuses on the consistency of the rotation speed. To measure this, a perceptual image distance is measured after rotating a car at a small fixed angle. This is done by calculating the euclidean distance of the feature vectors from the VGG16 [SZ14] network. In detail, first, an image is generated at a random rotation angle. After that, a second image with the same label and latent vector is generated, which, however, is rotated by 2.25° in a random direction. Both images are then forwarded to the perceptual network, which outputs a feature vector for each image. The distance of those two feature vectors is then measured using the Euclidean distance. This is repeated for large number of iterations and an a standard deviation of all distances is measured. A low standard deviation then corresponds to an overall more consistent rotation and vice versa. Since the background of an image has a large impact on the perceptual output, this metric only synthesizes images with black or white backgrounds. Those backgrounds have shown to be very consistent throughout a rotation, which allows us to only measure the perceptual distance that is caused by the car. The metric can be formulated as follows:

$$\mathcal{P}_{dist} = || \, VGG(G(z_i, y_i, \alpha_i)) - VGG(G(z_i, y_i, \alpha_i + \phi))||_2 \tag{7.1}$$

$$\mathcal{R} = \sqrt{\sum_{i=1}^{N} \left(\mathcal{P}_{dist} - \mu\right)^2} \tag{7.2}$$

Here, we sample an image in G with three parameters: a noise input z_i, a label from the dataset y_i and a random angle between 0° and 360° α_i. ϕ is a fixed angle that is set to 2.25° and μ denotes the average distance of all rotations so that we can calculate a standard deviation. For this metric we set N to 50000.
One limitation of this metric is, that it never outputs zero, even if the rotation shows an ideal 3D behavior. This is because, a real 3D car rotation from 'front center' to 'front left' might cause more perceptual changes than a rotation from 'front left' to 'profile left'. Nevertheless, this metric is still suitable for comparing the consistency of the rotation speed across different experiments.

7.1.3 Random Rotation Accuracy

The third rotation metric focuses on generating a car at the correct input angle. This metric is very similar to the conditional accuracy metric from section 5.2, where a classifier was utilized to evaluate how well each label was learned by the model. The same is done in this metric, however, instead of sampling real labels from the dataset, each rotation is drawn at random. To do so, the rotation vector from a real label is replaced with a random discrete $k \cdot 45°$ angle. This can be formulated as follows:

$$\mathcal{A}_{rot} = \frac{1}{N} \sum_{i=1}^{N} equal(C_r(G(z, y_i, \alpha_i)), \alpha_i) \tag{7.3}$$

Here, we sample α_i randomly from $\{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$ and C_r is the classifier for the rotation label. For this metric we set N to 10000

7.1.4 Continuous Rotation Distance

Like the previous metric, the final rotation metric measures how well the model is able to synthesize a car at a given input angle. However, instead of calculating an accuracy for the discrete rotations, we now measure how well the model synthesizes cars at continuous angles. To do so, we use the 3D bounding box estimator introduced by Arsalan Mousavian [Mou+17]. This estimator is able to predict the position and orientation of cars. It was trained on the KITTI dataset [GLU12], which contains about 7000 images of cars in traffic that have accurate annotations for position and rotation. The 3D bounding box estimator from Arsalan Mousavian first uses a region proposal network to detect the cars in the image, and then forwards the cropped car regions to a regression model. This regression model then predicts the angle of the car, which in turn is used to create a 3D bounding box. An example for this is shown in Figure 7.1.

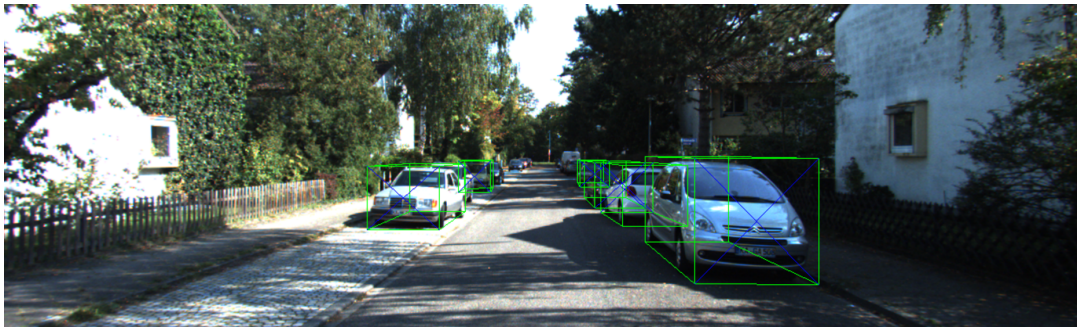


Figure 7.1: Examples for 3D bounding box estimations on the KITTI dataset (images source: [3db])

Since it requires some information about the camera calibration matrix to create a 3D bounding box in a 2D image, a bounding box, as shown in Figure 7.1, can not be created for generated images. However, in order to only measure the continuous angle of a synthesized car, it is sufficient to observe the output of the rotation regression model. Although the 3D bounding box estimator model was originally designed to predict the rotation angles of cars in traffic images, it also showed to perform well with the images from the car dataset. This was tested by hand using a large amount of images from the car dataset. When predicting the angle of synthesized images, however, it occurred multiple times that the estimator was off by about 180°. This is likely due to the fact that some cars can look very similar from opposing sides. Since the purpose of this metric is,

to correctly identify continuous rotation angles of a car, those errors are corrected. To ensure that the GAN still correctly synthesizes the car from all eight discrete rotation angles, we can apply the previous rotation metric in addition to this metric.
The output for this metric is the median of distances between target rotations and predicted rotations calculated with 5000 example images. A low output value corresponds to a low overall distance, which implies that the angle of the generated car is close to the input angle. We chose the median instead of the average, since some predicted angles showed large outliers.

$$\mathcal{D}_{rot} = \text{median}(\{|\alpha_i - \mathcal{E}_{rot}(G(z_i, y_i, \alpha_i))| \mid i \in [1, N]\}) \tag{7.4}$$

Here, $\mathcal{E}_{rot}$ stands for the continuous rotation estimator that outputs an angle from 0° to 360°. For the absolute distance, we consider the angle is periodic.

7.2 Problems with the Baseline Model

When using the baseline model from chapter 5, we are able to synthesize a 360° view of a car by interpolating in between all eight discrete rotation labels in consecutive order. To do so, we simply perform linear interpolation in between all neighboring rotations and append the resulting images to a sequence. The resulting image sequence shows an animation of a rotating car, as demonstrated in Figure 7.2. There it shows the rotation from the position 'Front Right' to 'Rear Right', which is generated by a linear interpolation from the 'Front Right' label (a) to the 'Profile Right' label (b) and then from the 'Profile Right' label (b) to the 'Rear Right' label (c). It shows that, instead of simply switching from one fixed orientation to the next, the car slowly turns around. To make the rotation look as consistent as possible, all other labels, the random latent vector, and the random added noise in the generator are fixed. This allows us to generate a car at any continuous angle. For instance, in order to generate a car at a 4.5° angle, the corresponding label vectors for the rotation $l_0 = 0°$ and $l_{45} = 45°$ are selected and an interpolated vector $v_{\text{int}} = 0.9 \cdot l_0 + 0.1 \cdot l_{45}$ is calculated.

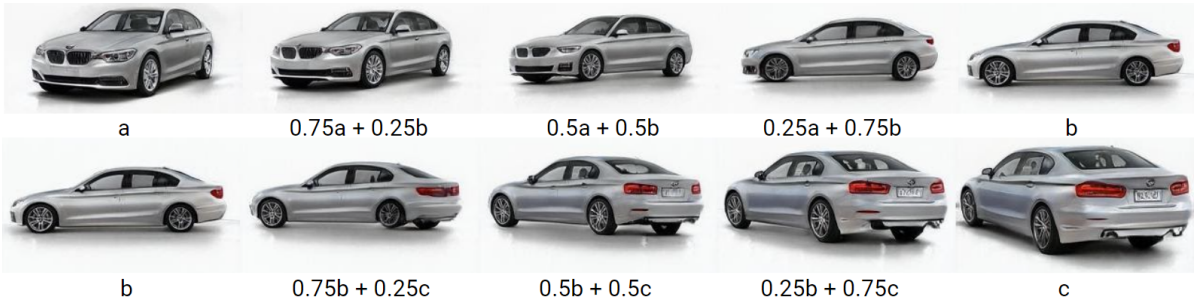


Figure 7.2: Example rotation, generated by interpolating from 'front left' (a) to 'profile left' (b) and then from 'profile left' (c) to 'rear left'.

The reason why the car is rotating, instead of simply fading from one location to the next, is most likely due to the characteristic of the generator to organize its own latent space. Organized, in this context, means that visually similar output images originate from latent vectors that have a close distance and vice versa. In an ideal training, the generator fills out the whole continuous latent space so that each vector corresponds to a realistic image. This implies that the generator is pressured to fill the spaces of the latent space in between the discrete rotations with realistic car images. Since a rotation is the simplest solution for a possible transformation from one viewpoint to the next, without collapsing the content of the image at any point, the generator is enforced to learn the 3D properties of the generated objects in the images. This was also observed in section 4.3, where we trained the StyleGAN without any labels and explored the latent space using the PCA. Also then, we observed that the images showed 3D properties when interpolating along the direction of the second component. Given that we can now take advantage of the rotation label, it makes the 3D properties considerably easier to control.
Although the image sequence from Figure 7.2 already shows a realistic looking rotation of a car, we find some problems when inspecting the rotation at a higher sampling frequency. In general, we found the following tree major problems:

- A bad transition quality from the front/rear view to the side views of the car,
- collapsing car tires in some intervals,
- and an overall inconsistent rotation speed.

In the following sections, we demonstrate all three problems using same the rotation from Figure 7.2.

Front / Rear Transition Problem

The first problem that we find, is the bad transition quality when rotating from the front or rear views towards one of the sides. Instead of a smooth 3D transition, the car slightly collapses for a short period and then reappears at a changed angle. This is demonstrated in Figure 7.3.

Figure 7.3: Example rotations around the front and the rear of a car, highlighting the bad transition quality.

A reason why especially this part of the rotation looks very unrealistic, might be that in those sections the appearance of the car changes the most. Almost no car parts that are visible from the side view are also visible from the front or rear. We hypothesize that this makes it very challenging for the model to find a realistic transition in between those different viewpoints. Another explanation for this problem might be that there are considerably fewer distinct training examples that show the front or rear of the car. Even though we balance the rotation classes during the training using oversampling, there are still a lot less individual training examples from the front or rear compared to the other views.

Collapsing Car Tires

When looking at the same sequence from Figure 7.2 in smaller angle steps, it shows that the car seems to jump from one position to another, at about 73°. Especially the front wheel starts to collapse at 71°, which then reappears at about 75° in a changed location.

Figure 7.4: The same rotation as in Figure 7.2, at a higher sampling rate. It shows that the wheel of the car collapses at about 73°.

This problem is also highlighted when calculating the perceptual (VGG16) distances between each rotation step of the interval 45° to 90°. This is done in Figure 7.5, where each value on the y-axis corresponds to the perceptual distance between the current image and the next image that is rotated by 0.45°. There it shows that the resulting graph has a spike at the same angle that the car tire collapsed in Figure 7.4.

Inconsistent Rotation Speed

In addition to the collapsing images along the rotation path, we also observed that the rotation speed is very inconsistent throughout the rotation. Near discrete rotation vectors, every 45°, the car seems to be almost static, whereas it turns very quickly in between. Since this is very difficult to demonstrate on paper, the rotation estimator from the continuous rotation distance metric is used. By plotting the input angle against the predicted angle of the rotation estimator (Figure 7.6), it shows that the predicted angle increases quickly in some areas and slowly in others. This underlines the observation that the rotation speed is very inconsistent. In addition to that, the graph also shows a spike at about 70° which corresponds to the collapsed car tire, described in the previous paragraph.

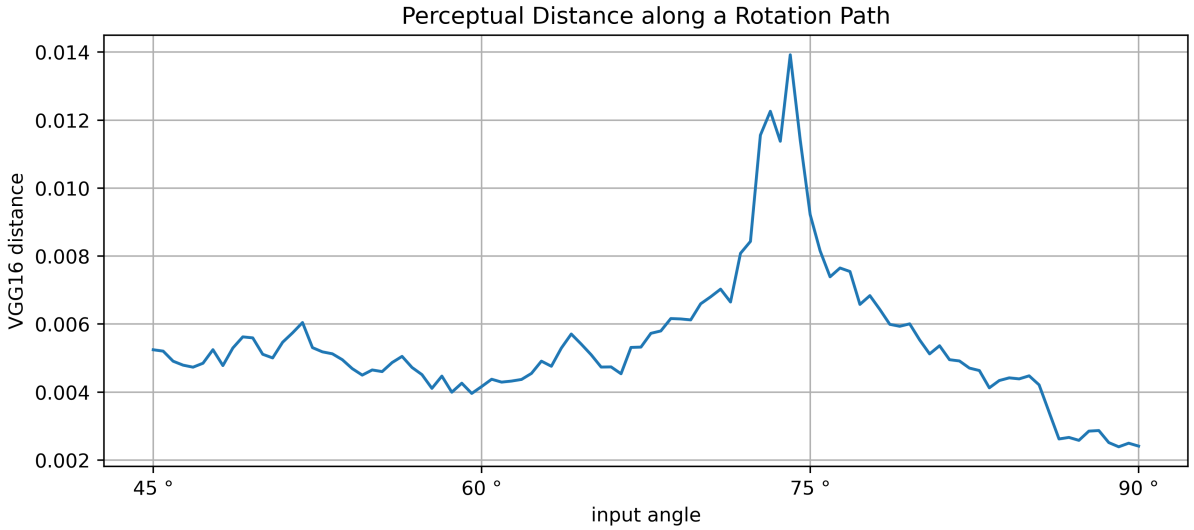


Figure 7.5: The perceptual distance (VGG16 [SZ14]) of the generated images along the rotation path of Figure 7.4 from 45° to 90°. This graph shows a spike at 73°, which corresponds to the angle at which the car tire collapsed.

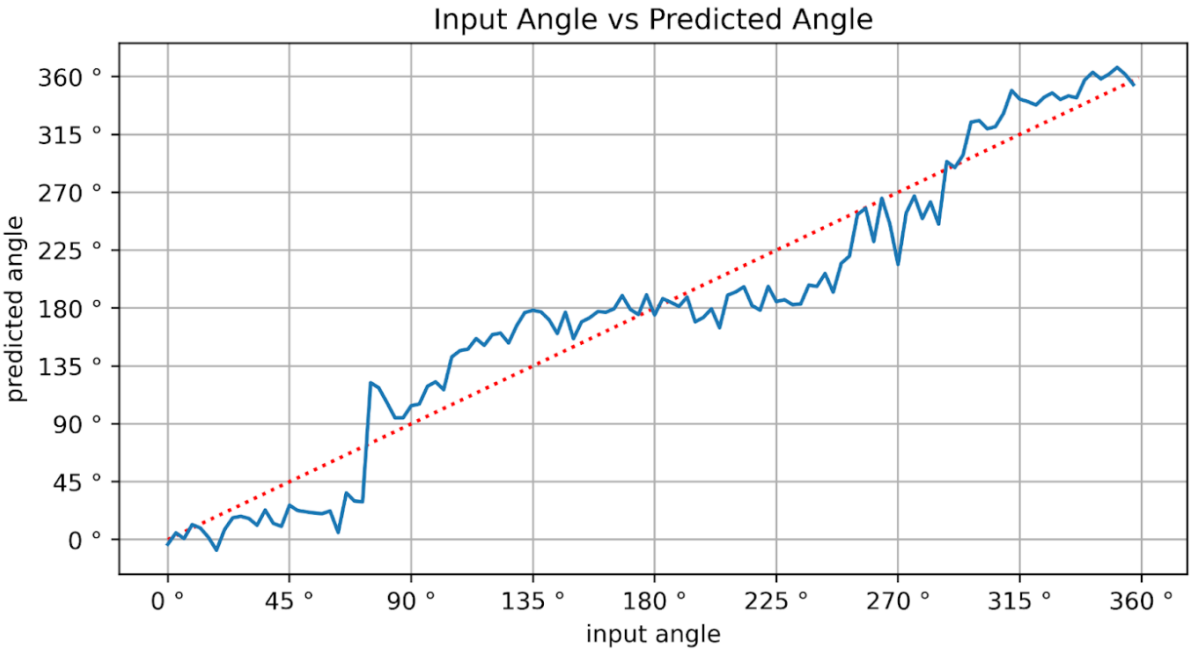


Figure 7.6: The predicted angle of the rotation estimator network against the target angle given to the generator. It shows that the rotation increases very inconsistently. The red dotted line corresponds to an ideal rotation, in which all input angles are equal to the predicted angles.

Rotation Metrics

In Table 7.1 we give an overview of the rotation metrics, from section 7.1. It shows that the rotation FID, calculated with continuous rotation angles, outputs a slightly higher distance compared to the traditional FID that exclusively uses the discrete rotations from the dataset. This indicates that the image quality is worse along the rotation paths, compared to the images at discrete $k \cdot 45°$ angles. This corresponds to the observed problems where car parts collapsed during a rotation.

Experiment	FID ↓	FID Rot. ↓	Rot. Accuracy ↑	Rot. Linearity ↓	Rot. Distance ↓
Baseline	4.46	5.53	90.5%	0.0118	19.87°

Table 7.1: An overview of the rotation metrics measured on the baseline model from the previous chapter. (↓: lower is better, ↑: higher is better)

The continuous rotation distance metric outputs a distance of 19.9° for the baseline model. This value is very high, considering that the maximum distance from any point on the circle to the closest discrete $k \cdot 45°$ angle is 22.5°. We, however, hypothesize that the actual continuous rotation distance is better than the metric suggests, since we observed that the rotation estimation network sometimes performed very poorly. Nevertheless, we still assume the output of this metric to be suitable for comparing the performance between different experiments.

Summary and Experiments Overview

In conclusion, we observed that the baseline model from the previous chapter is already able to create some sort of rotation when interpolating in between all eight discrete one-hot rotation vectors. This rotation, however, shows several problems. It is very inconsistent in its perceived rotation speed, the transitions around the rear and the front of the car look very poor, and the car tires sometimes collapse in between discrete rotations. With the goal of avoiding those problems, we propose the following three approaches in the next sections:

- *Sine / Cosine Rotation Label*: Our first approach replaces the one-hot rotation label with a more suitable two-dimensional vector that describes the position of the camera around the car.
- *Training with Continuous Rotations*: The second approach uses the continuous angles during the training to avoid any collapsing images along the rotation path.
- *Perceptual Rotation Regularization*: The last approach proposes a regularization to increase the consistency of the rotation speed.

7.3 Sine / Cosine Rotation Label

The first experiment focuses on improving the 360° view by revisiting the rotation label. Currently the rotation is encoded as an eight-dimensional one-hot vector, where each dimension represents a discrete 45° angle. We assume that this discrete encoding is ill-suited in order to create a smooth 360° rotation of a car. Instead, we propose a two-dimensional vector that describes the position of the camera with a vector on a unit circle, as shown in Figure 7.7. Therefore, we simply encode the sine and cosine of each angle. With this changed label type, we hypothesize that it should be easier for the model to learn the relationships between the rotation labels, given that adjacent rotations are closer on the unit circle, which was not the case with the one-hot encoding. There, each spacial distance between all rotation vectors was the same. Although some methods in the literature [Sho+21] propose to use the degree of the angle as rotation label, it is reasonable to use the sine and cosine in order to take advantage of their periodic characteristic.

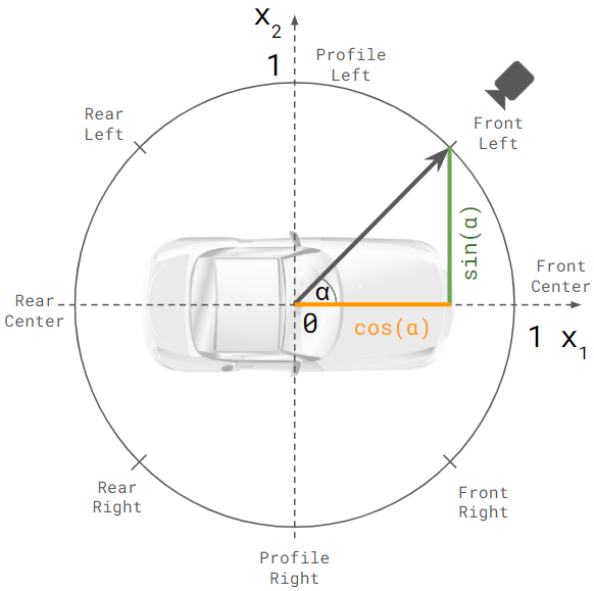


Figure 7.7: A visualization of the new rotation label encoding, where the position of the camera is described by the sine and cosine.

Model Adaptations

In order to train the StyleGAN with the new rotation label, we have to perform some adaptions in the loss functions and at the output of the discriminator. This is because we now design our discriminator to predict the current rotation of a car, in addition to predicting whether the image is real or fake. This means that the discriminator now receives two losses. An adversarial loss and a rotation loss. This rotation loss

is a regression loss, which is calculated by the squared euclidean distance between the rotation vector of the real image $r_{\text{real}} = (\cos\alpha, \sin\alpha)$ and the rotation prediction of the discriminator $D_r(x) = (o_{\cos}, o_{\sin})$ (Figure 7.8 (right) green line). To predict the rotation of a given image, we equip the discriminator with a separate linear output for both the sine and cosine $(o_{\cos}, o_{\sin})$, in addition to its adversarial output. To balance both loss values, a hyperparameter γ is introduced.

$$L_{\text{Disc}} = L_{\text{adv}} + \gamma \| D_r(x) - r_{\text{real}} \|_2^2 \tag{7.5}$$

Next to the discriminator, we also adapt the generator for the new rotation label. In addition to its adversarial loss, it now also receives a rotation loss. This loss gets calculated by the squared Euclidean distance between the desired rotation from the given label r_{fake} and the rotation prediction of the discriminator $D_r(G(r_{\text{fake}}))$ for the generated image (Figure 7.8 (right) red line).

$$L_{\text{Gen}} = L_{\text{adv}} + \gamma \| D_r(G(r_{\text{fake}})) - r_{\text{fake}} \|_2^2 \tag{7.6}$$

In summary, we now train the discriminator to predict the rotation vector of the real data, while the generator is trained to synthesize images, which produce a rotation prediction from the discriminator that is close to the input rotation. An overview of the changed GAN training is shown in Figure 7.8 (left).

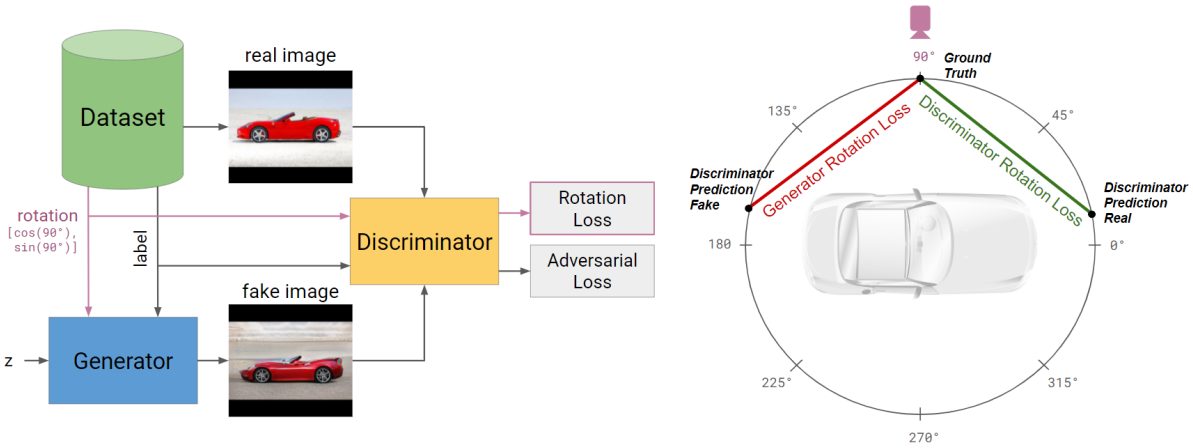


Figure 7.8: On the left, an overview of the changed GAN training, using the sine / cosine labels. And on the right, a visualization of the rotation loss of the generator and discriminator. To calculate the loss as described in Equation 7.5 and Equation 7.6, the distances are squared and then multiplied by a balancing parameter γ.

Classification Performance of both Label Encodings

Before training the StyleGAN model with the new rotation label, we first train the discriminator in isolation to ensure that it has no disadvantage when learning a regression task over a classification task. To do so, we compare two experiments where we train the discriminator as a classifier and then as a regression network. For the classifier we use the one-hot labels and a cross-entropy loss. And for the regression network we use the new sine / cosine label with the loss as described in section 7.3. After that, we compare the performance of both models by calculating an accuracy metric on a separate test set. To do so, we use the closest discrete rotation class for the regression model, since it outputs a vector instead of a discrete class.

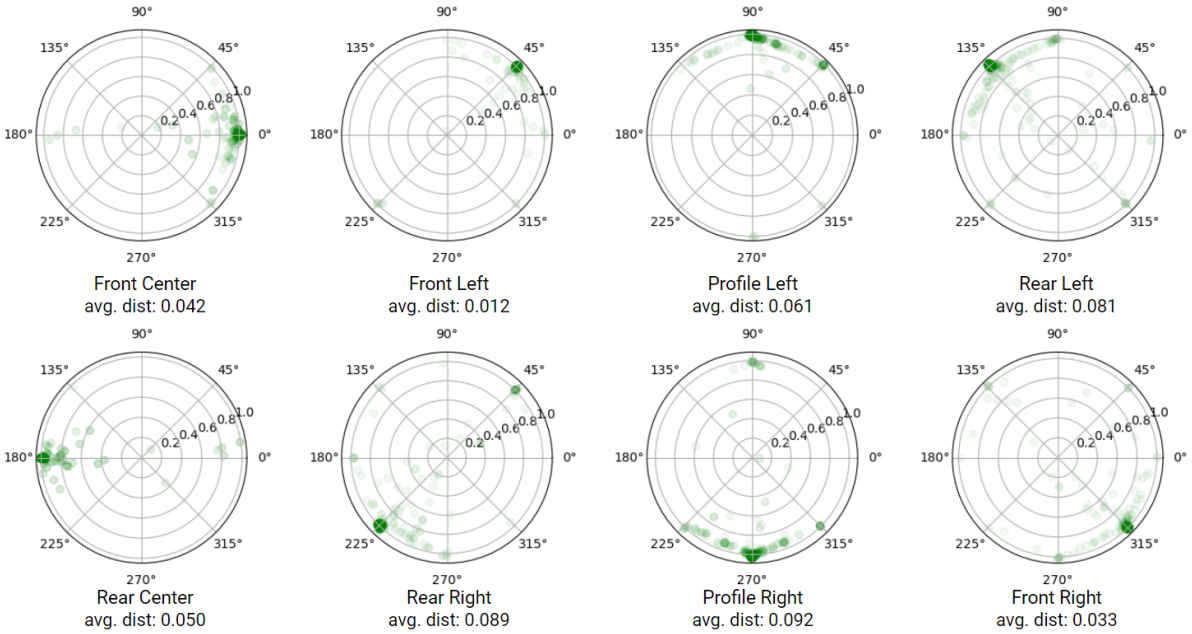


Figure 7.9: Visualization of the rotation prediction using the regression model with the sine / cosine label. For each rotation class, a prediction over 1000 samples from the test set is performed. It also shows the average distances, which is the lowest for the 'front left' label and the highest for the 'profile right label'.

After training the two models for a duration of one million training images, both showed very similar performance. The regression model performed slightly better with 95.9% accuracy compared to 94.9%. This already indicates that the model has no disadvantage when using the sine / cosine label instead of the one-hot label. In order to validate, if the predictions of the regression model are also close to the target vector, we measure the average Euclidean distance between the prediction and the target vector for 1000 test images. This is also visualized in Figure 7.9. It showed that the average distance of all predictions is 0.06. Considering that the distance between two discrete rotation labels is 0.77, it indicates that the training was successful. Figure 7.9 highlights that most of the predictions are located at the correct position, while some few spread towards the

two neighbors or the opposite side. While the 'front left' and 'front right' labels have the lowest average distance of 0.012 and 0.033, the 'profile left' and 'profile right' labels have the highest. They scored the largest average distance to the target rotation with 0.061 and 0.092.

Given that both models perform very similarly to each other in terms of accuracy, it motivates the use of the new rotation label for the StyleGAN training. This is for two reasons. The first reason is that the model might learn the semantics and the relationships of the rotation labels easier, when it is already provided with an arrangement in a two dimensional space. In this arrangement, the adjacent rotation labels are close to each other, whereas all distances of the eight-dimensional one-hot vectors are equal. The second reason is that the new label encoding facilitates the synthesis of continuous rotation angles, since they can simply be calculated by the sine and cosine of any given angle, instead of having to interpolate in between two of the eight one-hot vectors. This especially facilitates the design and implementation of further modifications to the model.
In the next section, we will train the conditional StyleGAN with the sine / cosine label.

Training with Sine / Cosine Labels

After training the StyleGAN with the sine / cosine labels for 5 million training images, it shows that the model is able to successfully learn the semantics of the rotation using the sine / cosine labels. Both the rotation distances of the generator and the discriminator decrease over the course of the training, as shown in Figure 7.10 (left). This indicates that the discriminator learned to correctly predict the rotation angle of the training images and the generator was able to produce images at the correct angles. To demonstrate this, we generate an image at every discrete $k \cdot 45°$ rotation angle in Figure 7.10 (right).

Figure 7.10: Left, shows the Euclidean rotation distance of the generator and the discriminator during training. The rotation loss is calculated with the squared rotation distance, multiplied by 10. Right, shows example images from all discrete rotation angles.

Rotation Accuracy and FID

Next to the successful examples from Figure 7.10, it also shows that the rotation accuracy, measured by a pre-trained classification network, has significantly increased. While the baseline model only scored an average rotation accuracy of 90.5% the new model with the sine / cosine label scored 99.7%. A comparison of the accuracies for each rotation class is shown in Figure 7.2. It demonstrates that no rotation class scores an accuracy worse than 99%, while in the previous baseline model some rotations only reach 87%. On the downside, however, the FID metric scores considerably worse than in the baseline model. As shown in Figure 7.11, the baseline scored a lowest FID of 4.46, whereas the new training scored a much higher FID of 7.27. We hypothesize that there is a trade off between image quality and rotation accuracy, controlled by the loss balancing parameter γ. Given that the FID is worse, while at the same time the discrete rotation accuracy is at 99%, it suggests that a lower magnitude for the rotation loss weight might benefit the image quality. This, however, was tested in prior experiments, which showed a significantly worse rotation results. Although those experiments, initially, showed very good FID values and good rotation accuracy, at some point during the training the generators only focused on the adversarial task. Then, the adversarial loss of the generator dominated the rotation loss, which caused the rotation loss to increase again. This in turn reduced the accuracy of the rotation label and also caused very poor looking rotation animations. In order to avoid such behavior in later training stages, we decided to increase the rotation loss weight from 1.0 to 10.0. With this magnitude, it is ensured that the adversarial loss does not dominate the rotation loss, even though it increases the FID.

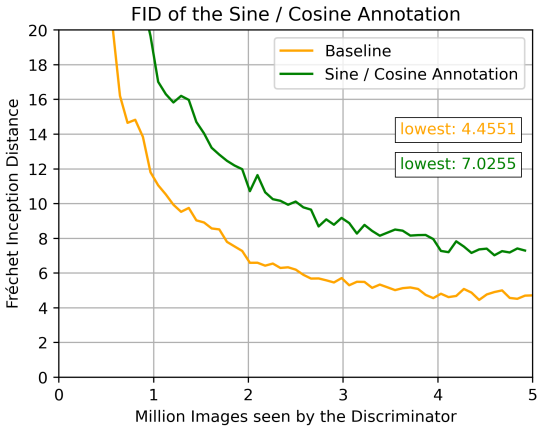


Figure 7.11: FID over the training of the baseline model with the one-hot rotation labels and the new model with the sine / cosine label.

Rotation	One-Hot	Sin / Cos
Front Center	88.5%	99.5%
Front Left	94.7%	100.0%
Profile Left	86.9%	99.3%
Rear Left	87.2%	99.2%
Rear Center	95.5%	100.0%
Rear Right	89.4%	100.0%
Profile Right	90.4%	99.9%
Front Right	91.4%	99.3%
Average	90.5%	99.7%

Table 7.2: Rotation accuracies compared to the baseline model with one-hot rotation labels. It is calculated with 10k generated images at random 45° angles.

Rotation Improvements

The most noticeable difference between the baseline model and the new model with the sine / cosine label appears when synthesizing a full rotation. With the baseline model, those rotations showed three major problems. Firstly, the transitions from the front or rear view to the side of the car looked very poorly. Secondly, the car tire collapsed at some positions. And at last, the overall rotation speed was very inconsistent. Although, we still observe some rotations in which the car tire collapses, the others problems significantly improved. This can be demonstrated by synthesizing a rotation from the front view to the side view of the car, as shown in Figure 7.12. While in the baseline model, the rotations consistently collapsed at that rotation angle, it now smoothly turns using the sine / cosine labels. This indicates that the new label helps the model to find a better relationship between neighboring rotations classes.

Figure 7.12: Two comparable rotation sequences of the baseline model and the new model from 12° to 42°. It shows that the car slightly collapses at 24° when using the baseline model and smoothly rotates with the new model.

The improved rotation is also underlined by the rotation linearity metric, which measures the average standard deviation of the perceptual image distances along a rotation path. While the metric measured a standard deviation of 0.0118 for the baseline model, the new model only measures 0.0069. This result is more than 40% lower and indicates that the perceptual rotation speed is much more consistent. An example for this can again be visualized by plotting the input angle against the predicted angle of the continuous rotation estimator. This is shown in Figure 7.13, where two similar car rotations from the baseline model and the new model are compared. It shows that the predicted rotation angles of the new model are more linear and that the angles increase more consistently. This observation is also underlined by the continuous rotation distance metric. There the baseline model achieved a median distance of 19.9°, whereas the new model only measures 16.4°.

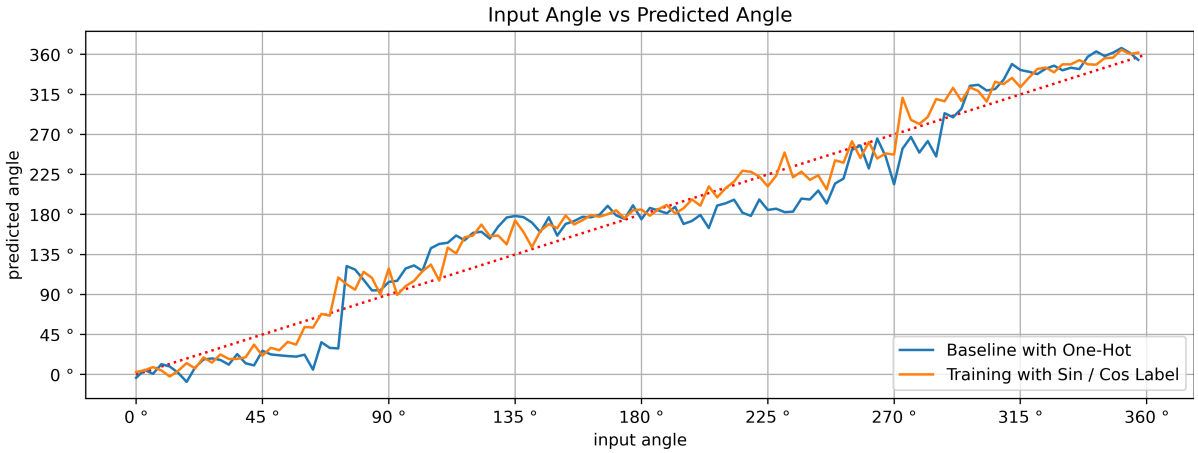


Figure 7.13: The graph plots the input angle of the generator against the continuous angle prediction from the rotation estimation network. Here, two similar rotations from the training with one-hot labels and sine / cosine labels are compared.

Rotation Problems

Although all the above rotation metrics suggest that the rotation is better when using the sine / cosine labels, some images along the rotation paths show that the car tire still collapses at some points during the rotation. An example for this is demonstrated in Figure 7.14. This shows that the model still struggles to create smooth rotations with 3D properties. On the upside, however, the difference between the FID calculated on discrete rotation labels and the FID calculated on continuous rotation angles is considerably smaller when using the sine / cosine label. For the baseline model the FID increases from 4.46 to 5.53, when using continuous rotation angles, while in the new model the FID only increases from 7.27 to 7.46. Although both FID values are higher with the new rotation label, it might still might hint that the images are less prone to collapse in between discrete rotations. This is because we assume that any collapsing image should increase the FID marginally. Since, however, the FID only increases by 0.19 using the sine / cosine label, we hypothesize that fewer images collapse during a rotation, compared to the baseline model.

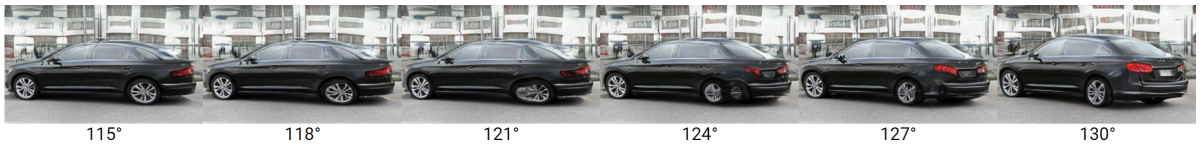


Figure 7.14: An example rotation sequence, where the car tire collapses at 121° similar to previous experiments.

Summary

In summary, this experiment showed that the training with the sine / cosine labels significantly increase the quality and consistency of the rotation. This was shown in every rotation metric, apart from both FID metrics. However, given that both FID values are very close to each other, we hypothesize that this also indicates an improvement compared to the baseline model, in which the two FID values were further apart.
Although the FID might improve even further if we lower the loss balancing parameter for the rotation loss, we do not modify this value for the rest of the thesis.

Experiment	FID ↓	FID Rot. ↓	Rot. Accuracy ↑	Rot. Linearity ↓	Rot. Distance ↓
One-Hot	**4.46**	**5.53**	90.5%	0.0118	19.9°
Sin / Cos	7.27	7.46	**99.7%**	**0.0069**	**16.4°**

Table 7.3: An overview of the rotation metrics compared between the experiments using the one-hot labels and the sine / cosine labels.

Due to the overall success of this experiment, we perform all of the following rotation experiments with the sine / cosine rotation label as well. Therefore, this experiment can be considered as a second baseline model for this thesis. We also performed all of the following experiments using the one-hot rotation label, however, since they all showed consistently worse results, they are not analyzed in the following sections.

7.4 Training with Continuous Rotations

In the next experiment we target the collapsing images during a rotation. Therefore, we synthesize car images at continuous angles already during the training. By doing so, we expect that the image quality along the rotation paths improves, given that the generator now also receives a feedback for those images. With this approach we assume that especially those images, which showed a collapsing car tire, should be penalized the most, as they likely produce a high loss in the discriminator. This in turn, could improve the rotation quality significantly.
The experiment about the label randomization in subsection 5.7.1 underlined that it can be very dangerous to provide the generator with different labels other than from the dataset. There it showed that the images collapsed as soon as the images were synthesized by label combinations that did not exist in the dataset. This is also a problem for this experiment, since the training data only provides discrete rotation labels. In order to avoid a situation, where the labels have a very unbalanced difficulty for the generator, we only provide few continuous rotation angles to the generator, while the rest of the rotations are still drawn from the dataset. For the following experiment, we test 20% and 40% continuous angles.
Since the discriminator never receives any real training examples with continuous angles, we expect that the rotation loss from section 7.3 does not produce any meaningful

rotation feedback for the continuous images in the generator. Therefore, we propose two loss variants in the following. In the first variant, the rotation loss is ignored for the continuous angles and in the second variant, we train the discriminator to also learn the continuous rotation angles. In the following subsections, we analyze both variants separately.

7.4.1 Ignoring the Continuous Rotation Loss

As described in the previous experiment, both the generator and the discriminator receive an additional rotation loss next to the adversarial loss, when using the sine / cosine labels. For the discriminator, the rotation loss is calculated by the Euclidean distance between the predicted rotation vector and the rotation vector of the training data and for the generator the rotation loss is calculated by the Euclidean distance between the target rotation vector and the predicted rotation vector from the discriminator. Since in this experiment the generator now also produces images at continuous rotation angles, we have to adapt the loss function. This is because the discriminator only receives real training samples at discrete $k \cdot 45°$ angles during the training. If the discriminator now has to predict the continuous angle of a generated car, we do not expected it to produce a meaningful output. Instead, it will likely predict a discrete $k \cdot 45°$ angle, like in the training examples. This feedback, however, would not be very helpful for the generator. This is why, in this approach, the predictions for the continuous rotations are simply ignored by the generator, when it synthesizes a car at a continuous angle. This means, that the generator only receives an adversarial loss for images generated at continuous angles. With this approach we hypothesize, that the generator should create a better rotation, as it gets penalized if any image within a rotation looks unrealistic. This could then avoid the collapsing tires that were observed in the previous experiments.
In the following two sections, we first evaluate an experiment with 20% continuous rotation angles is evaluated and then with 40%.

Training Results with 20% Continuous Angles

After training the first experiment, in which 20% of the input labels for the generator are replaced with random continuous angles, the model simultaneously showed very bad, but also very promising results. When synthesizing a full 360° rotation with the model, the transitions look very good, in between 0° to 180°, however, start to look worse between 225° and 270°. This observation showed to be consistent for all synthesized rotations with this model. To visualize this, two rotation sequences are shown in the following. Figure 7.15 shows a rotation from 0° to 91°, which looks very good. Instead of quickly jumping from one position to the next, like it was observed in the previous results, the car smoothly turns from one angle to the next. Even though, the rotation from 0° to 7° is especially difficult for the model, given that it has to add car parts, such as the tires, that are only visible from the side of the car, it looks very realistic.

This good rotation quality was observed for almost all generated images. In addition to that, the car tires do not collapse at any point during the first half of the rotation. This is a major improvement to the last experiments in which the tires collapsed for almost every rotation. This hints that the continuous rotation angles help the model to remove unrealistic images along the rotation path.

Figure 7.15: An example rotation from 0° to 91°, demonstrating the good rotation quality when training with 20% continuous rotation angles.

Figure 7.16, on the other hand, shows a rotation from 236° to 266°, which looks very poorly. There, the car collapses for a short interval between 230° and 254°. It shows that the rear left tire transitions to the rear right tire, which causes the front left tire to quickly disappear. This makes the rotation look very unrealistic.

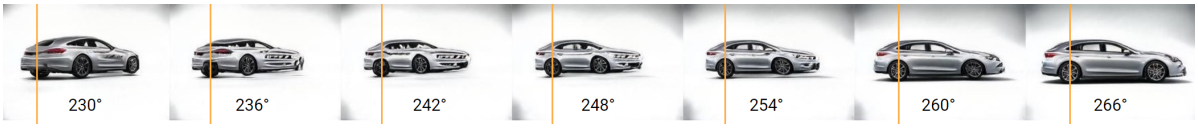


Figure 7.16: A bad rotation example between 236° to 266°. Here the rear left tire transitions to the rear right tire, causing the car to collapse for a short interval. The orange line is placed at the same position for all images, highlighting that the position of the car tire is fixed.

After we tested a large number of rotations by hand, it shows that this behavior is seen in every rotation generated by this model. This indicates that the model favors an alternative transition that does not correspond with the 3D properties of a car. Given that this effect was already visible at the early training stages of the model, it underlines that the generator has learned this incorrect transition right from the beginning. As the generator initially produces images with low quality, it is possible that this transition was not penalized enough by the discriminator at the start of the training. Although, in later training stages, those bad images should produce a higher loss, it might be too late for the generator to correct those transitions.
This bad rotation is also reflected in the results of the rotation metrics, shown in Table 7.4. There it shows that every rotation metric, especially the FID with continuous angles and the rotation linearity, has decreased compared to the baseline model from the previous section. Only the traditional FID metric has improved. This is likely because of the reduced rotation loss, since this loss is now being ignored for 20% of the fake images. This allows the model to set more of its focus on the adversarial task.

Experiment	FID ↓	FID Rot. ↓	Rot. Accuracy ↑	Rot. Linearity ↓	Rot. Distance ↓
Baseline Sin / Cos	7.27	**7.46**	**99.7%**	**0.0069**	**16.4°**
20% Continuous Rot. (ignore loss)	**6.65**	9.53	98.9%	0.0113	18.0°

Table 7.4: An overview of the rotation metrics compared to the baseline model using the sine / cosine labels.

Even though the rotation of this model looks very bad for one section of the rotation, it still shows considerably better transitions than the baseline model for the rest of the rotation. One simple workaround for this problem would be to exclusively use the first half of the rotation and mirror it for the second. This, however, would be a disappointing solution, given that all other features such as the image background would be mirrored as well. And especially after we observed that the rotation improved for the first half of the rotation, it motivates us to find a solution where the second half works as well. Therefore, we perform a second experiment, in which 40% instead of 20% of the the labels have a continuous rotation. With this increased proportion of continuous angles, we hypothesize that the rotation on both sides look better, given that twice as many continuous angles are sampled during the training. This could penalize the incorrect transitions at the beginning of the training even more.

Training Results with 40% Continuous Angles

Contrary to the expectations, the results of the training with 40% continuous angles showed very poor results compared to the experiment with 20%. Instead of enforcing the model to improve the quality for the second half of the rotation as well, the images now fully collapse around 0° and 180°. An example for this is shown in Figure 7.17.

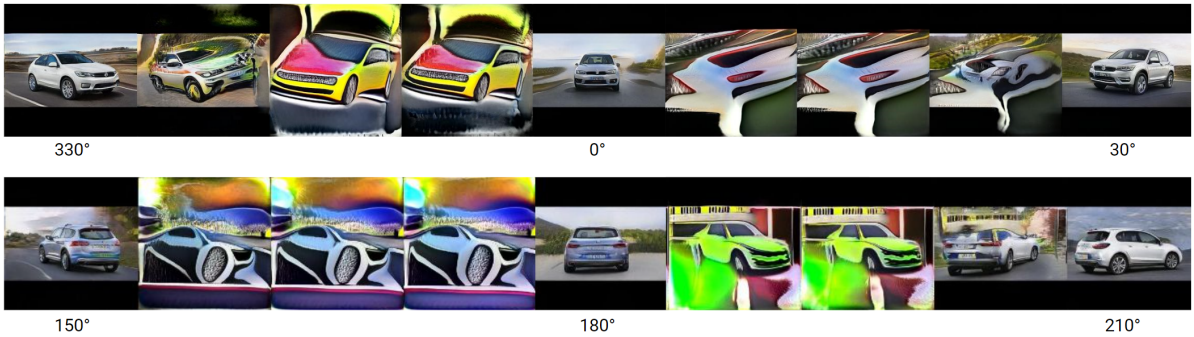


Figure 7.17: An example rotation centered at 0° and 180°, highlighting the collapsing images, when the car makes a transition to the side.

This observation is similar to the result of the soft label randomization experiment from subsection 5.7.1. There, we slightly randomized the labels for the generator during the training, which caused the synthesized images to collapse as soon as a class of one label was changed. Back then, this was explained by an unbalance between the difficulty

between real labels from the dataset and randomized labels. This is likely to be the case for this experiment as well. However, in this case, the difficult labels are the regions near the 'front center' and the 'rear center' view of the car. This makes sense as the appearance of the vehicle changes the most in those regions.

Summary

Overall, the last two experiments have shown that rotation can generally be improved with this method, although it is very difficult to find a suitable proportion between discrete and continuous angles. If too many continuous angles are used during the training, the images start to collapse at the front and the rear view and if too few are used, the generator might not be penalized enough for creating poor looking transitions. Instead of testing further proportions, an alternative approach is tested in the following.

7.4.2 Cooperative Continuous Rotation Loss

The second approach for handling the continuous rotation loss is inspired by a semi-supervised learning algorithm. The main idea of this approach is to teach the continuous rotations to the discriminator by simultaneously training it with the discrete labels from the training data but also with synthesized data from the generator. This means that the discriminator learns the discrete rotation angles from the training data and the continuous rotation angles from the generator. This way, the generator can also receive feedback for the continuous rotation angles. With this approach, both networks have to agree on a representation of a car at a continuous angle, without ever observing such a training example from the real data. With this experiment, it is either expected that the continuous rotations look good and consistent, or very bad. This could be the case, if both models agree on a representation that does not correspond to a 3D rotation. On the other hand, it might as well be very good, considering that the generator already produced 3D-like transformations only using discrete labels. The generator could then receive useful feedback for the continuous rotation views, which would help to generate more consistent rotation angles. With this approach, the resulting loss function of the discriminator looks as follows:

$$L_{\text{Disc}} = L_{\text{adv}} + \gamma \| D_r(x) - \alpha_{\text{real}} \|_2^2 + \gamma \| D_r(G(z)) - \alpha_{\text{input}} \|_2^2 \tag{7.7}$$

Here, $D_r(x)$ and $D_r(G(z))$ are the angle predictions of the discriminator for the real and the fake images, respectively. γ is the balancing weight between the adversarial loss and the rotation loss. Although this value can also be different for both rotation losses, we chose it to be the same for both rotation losses.
Like in the previous experiment, we first test 20% continuous angles and then 40%.

Training Results with 20% Continuous Angles

Other than in the last two experiment, the cooperative approach shows very good results. After testing a large number of rotations, it showed that the problem from previous experiments, where the car tire collapsed, got completely removed. This makes the rotations look much more realistic compared to all previous models. An example for this is shown in Figure 7.18. It shows that none of the images during the rotation collapse.

Figure 7.18: An example rotation of the model with the cooperative rotation loss with 20% continuous rotation angles during the training. The rotation demonstrates that none of the images collapse during the entire rotation.

This rotation improvement is also reflected in the output of the rotation metrics (Table 7.5). It shows that especially both FID values and the continuous rotation distance improved. This underlines that the images are less likely to collapse and that the predicted angles are closer to the target angles.

Experiment	FID ↓	FID Rot. ↓	Rot. Accuracy ↑	Rot. Linearity ↓	Rot. Distance ↓
Baseline Sin / Cos	7.27	7.46	**99.7%**	**0.0069**	16.4°
20% Continuous Rot. (coop)	**5.70**	**6.76**	98.5%	0.0100	**15,4°**

Table 7.5: An overview of the rotation metrics compared to the baseline model using the sine / cosine labels. It shows that especially the FID results and the continuous rotation distance improved.

On the downside, however, the rotation linearity significantly increased by about 60% compared to the baseline model. This indicates that the rotation speed is much more

inconsistent using the cooperative loss formulation. One possible reason for this can be seen in the synthesized rotations. When rotating from the front to the side view, as demonstrated in Figure 7.19, the side of the car suddenly appears at around 21°. As this effect was also found in many other synthesized rotations, it could explain the increased rotation linearity.

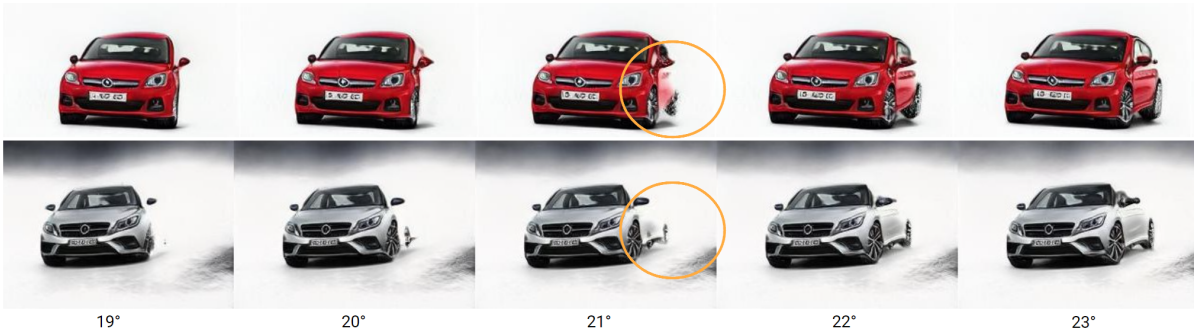


Figure 7.19: Two example rotations, in which the side of the car suddenly appears when interpolating from the front to the side. The orange circles highlight the side of the car at 21°, which was not visible at 20°.

Although this effect mitigates the realism of the rotation, we still consider this experiment as most successful so far. This is because, for the first time none of the car parts collapsed during the rotations. For that reason, we repeat this experiment with a higher percentage of continuous angles. Instead of 20%, we now test 40% in the following section.

Training Results with 40% Continuous Angles

After training the same model as before with 40% instead of 20% continuous angles, the results initially looked very similar. In both experiments, no collapsing car tires were found but the rotation linearity increased. On further examination, however, it shows that especially the 'front center' view sometimes looks very poorly. An example for this is shown in Figure 7.20. There the image at 0° shows an angle of about 340°. This causes the car to skip the 'front center' view and directly fade into the front left view. This creates unrealistic images that display both sides of the car simultaneously, as shown in Figure 7.20 at 5° or 10°.

Figure 7.20: An example rotation around the front view of the car, showing that 'front center' view is skipped.

This problem can especially be highlighted by the rotation accuracy metric, as shown in Table 7.6. There, the accuracy for the 'front center' view is only at 81.5%, while it was 99.5% for the baseline model. This hints that this problem occurs in about 20% of the synthesized rotations. Next to the 'front center' view, also the accuracy for the 'rear center' view has decreased. At the rear, however, no significant artifacts were found after synthesizing numerous rotations by hand.

Rotation	Baseline Sin / Cos	Cooperative Loss 20%	Cooperative Loss 40%
Front Center	**99.5 %**	**91.8%**	**81.5%**
Front Left	100 %	100%	100%
Profile Left	99.3 %	100%	99.6%
Rear Left	99.2 %	100%	99.7%
Rear Center	**100 %**	**96.7%**	**96.3%**
Rear Right	100 %	100%	99.7%
Profile Right	99.9 %	100%	100%
Front Right	99.3 %	100%	100%
Average	99.7 %	98.5%	97.0%

Table 7.6: An overview of the rotation accuracies for each class compared to the baseline model using the sine / cosine labels. It underlines that the 'front center' and 'rear center' views perform worse with increasing continuous rotation angles.

Table 7.6 also shows that in the previous experiment with only 20% continuous angles, the accuracy for the front center view also decreased to 91.8%. This indicates that the same effect might also be found there. However, after further testing with the previous model, it shows that this effect is not found. Instead, it is sometimes observed that the rotation is slightly shifted. This means that the car shows a small rotation to either side at the input angle 0°, while the actual 'front center' view is located at about ±10°. An example for this is shown in Figure 7.21, where the car is slightly rotated at 0°, while the actual 'front center' view is found at 354°. However, since the front view is not skipped like in current experiment, the rotations from the previous experiment with 20% continuous angles look far better.

Figure 7.21: An example rotation, generated by the previous model with 20% continuous angles, where the car is slightly rotated at 0° and the actual 'front center' view is found at 354°.

Summary

Altogether it showed that the training with the cooperative rotation loss significantly improved the rotation, given that none of the car tires collapsed at any point. This makes the rotation look far more realistic compared to the previous experiments. The only drawbacks of this experiment are that the quality of the front and rear view of the car slightly decreases and that the rotation linearity metric performs worse. In order to encounter this problem, the next experiment proposes a regularization technique to stabilize the rotation speed.

7.5 Perceptual Rotation Regularization

The next experiment aims to improve the rotation linearity, by adding a regularization similar to the path length regularization from the StyleGAN paper [Kar+19]. As described in subsection 2.2.3, the path length regularization is designed to align the distances in the latent space with the magnitude of change in the output images. This is done, since it is expected that linear interpolations in the latent space correspond to smoother transitions in the output images [Kar+19], if a fixed step in the latent space causes a fixed change in the output image.
In the StyleGAN, this is implemented by calculating the length of the gradient from a modified image to its original latent vector w, using backpropagation. This length is then compared to its own moving average from all previous iterations. This way, the regularization penalizes the model, if the length of the gradient vector changes, although the corresponding image is modified by a fixed magnitude. The regularization can be formulated with the following expression:

$$\text{Reg}_G = \mathbb{E}_{w, I_{\text{noise}} \sim \mathcal{N}(0,1)} \left(||\nabla_w (G(w) I_{\text{noise}})||_2 - A \right)^2 \tag{7.8}$$

Here, I_{noise} denotes the multiplied noise image that is used to modify the output image by a fixed magnitude and A is the moving average of the length of the gradients.
In this experiment we test, if a similar regularization technique can be applied to smoothen the rotation interpolation paths instead of arbitrary latent space paths. Therefore, we measure the perceptual distance with the VGG16 [SZ14] model when rotating the car by a fixed angle. This distance is then compared to its moving average to penalize the model, if the perceptual distance is inconsistent throughout the rotation. This can be formulated as follows:

$$\text{Reg}_G = \mathbb{E}_{z, \alpha \sim [0,360]} \left(||\text{VGG}(G(z, \alpha)) - \text{VGG}(G(z, \alpha + \phi))||_2 - A \right)^2 \tag{7.9}$$

Here, A is the moving average of all previous distances and ϕ is the fixed constant angle that is added to the random angle α to rotate the car. In order to balance the rotation

loss with the regularization, a weight is introduced that multiplies the regularization result before it gets added to the loss. For the following experiments we set this weight to 10 and 20, while ϕ is set to 4.5°. Unlike the previous experiments, we evaluate the results of both experiments simultaneously.

Training Results

After training the StyleGAN with the perceptual rotation regularization, it shows the following penalty graphs (Figure 7.22). At the beginning, both penalties are very low, as the images show only very little variation. Then, after about 200k training images both penalty terms slowly start to increase. At this point the generator starts to produce more variation. After about one million training images both graphs reach their maximum and slowly decrease from this point onwards. It shows that in the training, which uses twice the regularization weight, the graph decreases slightly faster.

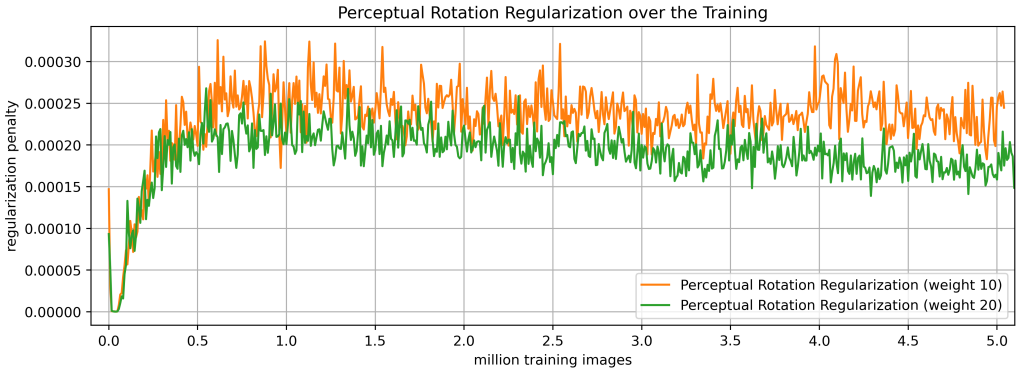


Figure 7.22: The penalty graph of the perceptual rotation regularization during the training. It shows that the experiment with the higher regularization weight decreases slightly faster.

In order to validate if the regularization improves the rotation, we measure all four rotation metrics, as shown in Table 7.7. Those especially underline that the rotation linearity decreased. This indicates that the helped to reduce the perceptual distance between two images along a rotation path. This in turn implies that the rotations are smoother.

Experiment	FID ↓	FID Rot. ↓	Rot. Accuracy ↑	Rot. Linearity ↓	Rot. Distance ↓
Baseline Sin / Cos	7.27	**7.46**	**99.7%**	0.0069	16.4°
Perceptual Rot. Reg. (weight=10)	6.84	7.49	99.4%	0.0047	16,1°
Perceptual Rot. Reg. (weight=20)	**6.67**	8.17	99.6%	**0.0044**	**15,5°**

Table 7.7: An overview of the rotation metrics compared to the baseline model using the sine / cosine labels.

To visualize this with an example, Figure 7.23 shows the incremental perceptual distance for three comparable rotations of all three models. Therefore, each value calculates the distance of the current image to the next image and adds it to all previous distances. By doing so, the gradient of the resulting curves describes the magnitude of change at the current angle. As a result, it shows that the rotations, synthesized with the models that use the regularization, show a lower and also more consistent curve compared to the baseline model. This highlights that the images change a lot less during a rotation when using the regularization.

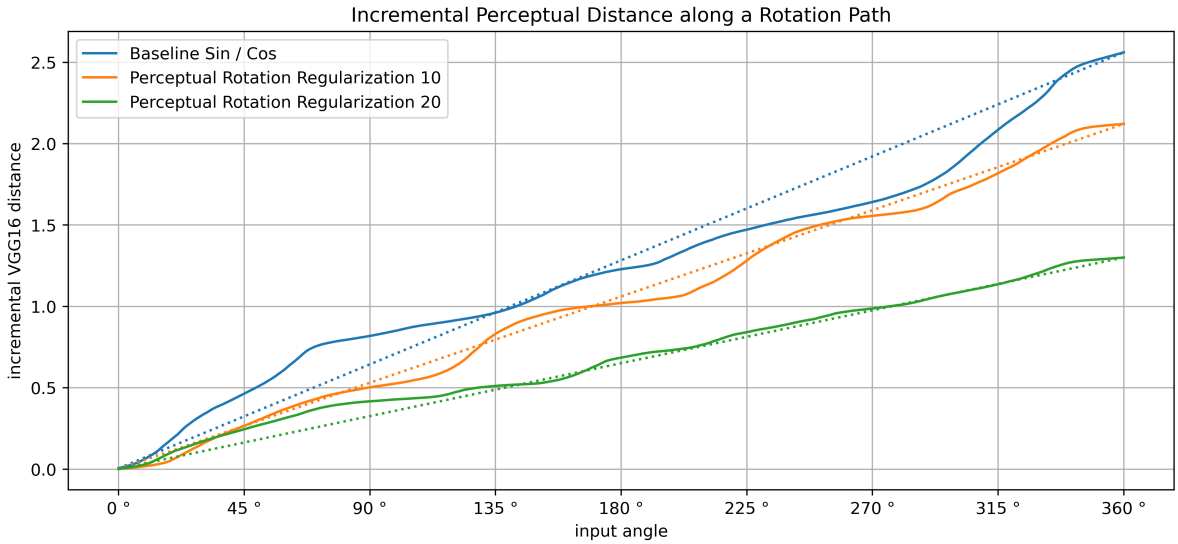


Figure 7.23: An incremental perceptual distance graph, for three comparable rotations using the rotation regularization with different regularization weights. It shows that with increasing regularization weight, the perceptual distances become smaller and more linear.

Next to the rotation linearity metric, also the continuous rotation distance improved. This indicates as well that the rotations show better results when using the regularization. On the downside, however, the FID, measured at continuous angles increased considerably when using the regularization with a weight of 20 instead of 10. A reason for this can also be found when generating some rotations by hand. It then shows that the images often collapse during a rotation at about 100°, as demonstrated in Figure 7.24. This hints that the regularization actually harms the quality of the rotation if it is set too high. An explanation for this might be that in an ideal 3D rotation of a car, the perceptual distances are also not consistent throughout the whole rotation. Instead, it is expected that the rotations at $k \cdot 90°$ angles should have a lower perceptual distance than the rotations at $45° + k \cdot 90°$ angles. Since the regularization aims to equalize all perceptual distances irrespectively to the current angle, it might cause very bad transitions at angles at which the perceptual distance of a real 3D car should be high.

Figure 7.24: A bad example for a rotation from 94° to 110°, showing that the car fades from one rotation to the next instead of turning around.

Summary

Altogether, it showed that the perceptual rotation regularization generally helps the model to create an overall smoother rotation. This was shown by the improved rotation linearity and the smaller continuous rotation distance. If, however, the regularization weight is set too high, it causes very bad transitions during a rotation. This might be because the model then steers away from a 3D rotation and instead creates a transition in which the perceptual distances are smaller. For that reason, we will only use the regularization with a weight of 10 for the next experiment, in which we combine the successful components of all rotation experiment in one model.

7.6 Combining the Components

In the next experiment, we test how well the proposed modifications from the last sections can be combined. To do so, we perform an experiment using the three modifications that achieved the best performance increases. Those are:

- The sine / cosine label from section 7.3, which especially improved the rotations around the front and the rear of a car.
- The training with 20% continuous rotation angles with a cooperative rotation loss from section 7.4 that removed the collapsing car tires during a rotation, however at the cost of the consistent rotation speed.
- And a perceptual rotation regularization with a weight of 10 from section 7.5, which improved the rotation speed.

Training Results

After training the model with all three successful modifications combined, it showed very promising results. As demonstrated in Table 7.8, all rotation metrics, apart from the rotation accuracy improved compared to the baseline model. This already indicates that the resulting rotations look considerably better than in the previous models. It especially shows that the continuous rotation distance and the FID at continuous angles is lower than all previous experiments. This hints that the continuous angles are closer to the input angles and that the rotations are less prone to collapse.

Experiment	FID ↓	FID Rot. ↓	Rot. Accuracy ↑	Rot. Linearity ↓	Rot. Distance ↓
Baseline Sin / Cos	7.27	7.46	99.7%	0.0069	16.4°
20% Continuous Rot. (cooperative)	**5.70**	6.76	98.5%	0.0100	15.4°
Perceptual Rot. Reg. (weight=10)	6.84	7.49	99.4%	**0.0047**	16,1°
Combine	5.8	**6.20**	98.8%	0.0057	**14.4°**

Table 7.8: An overview of the rotation metrics compared to the baseline model using the sine / cosine labels.

The good result from the rotation metrics also correlates with the perceived rotation quality when synthesizing some examples per hand. Figure 7.25, for instance, shows a good example of a rotation from 0° to 170°, in which the car does not collapse at any point during the rotation.

Figure 7.25: An example rotation from 0° to 170°, underlining the improved rotation quality.

Although this rotation looks considerably better than all previous results, it still shows some small problems when rotating towards the rear view of the car. It was sometimes observed that the front tire gets converted into the rear tire, as shown in Figure 7.26. This seems like a logical solution for the network, given that it transitions the left most tire from one perspective to the left most tire of another perspective. However, by doing so, it requires the rear tire from the side view to vanish, which in turn breaks the 3D properties of the car. This was especially observed with cars that were equipped with a large exhaust pipes, like in Figure 7.26. There it showed that the former rear left tire often got converted into the left exhaust pipe.

Figure 7.26: An example rotation from 185° to 189°, in which the rear left tire gets converted to the left exhaust pipe.

Summary

In summary, it showed that the successful components from previous sections also work well in combination. This was shown with the FID metric that uses continuous angles and with the continuous rotation distance metric. Both scored considerably better results than previous experiments. In addition to that, we also observed a good rotation quality when inspecting some example rotations by hand. Although few transitions around the rear of the car still caused problems, the general rotation quality improved a lot compared to all previous results.

7.7 Summary

In this chapter, we successfully synthesized a full 360° rotation of a car only using the eight discrete rotation labels from the car dataset. This was mainly achieved by the changed rotation label that allowed us to encode relationships between each rotation class. As a result, we observed considerably smother and especially more realistic transitions in between each 45° angle. In addition to that, we also proposed a semi-supervised loss function that enables to learn continuous rotations in the discriminator using the generated images as additional training data. This helped to reduce collapsing images during a rotation, as the generator and the discriminator cooperatively learned all continuous viewpoints around cars. This modification, however, came at the cost of a worse rotation linearity. To encounter this, we also proposed a regularization technique that produced considerably smother transitions. At last, we also combined all three components in one model, which resulted in an overall highest rotation quality. In order to test how well those modifications scale with increased image resolution, we will train the model from section 7.6 again at a resolution of 512×512 in the next chapter. In addition to that, we will also apply the modifications from chapter 5 to increase the image quality especially for unseen label combinations.

8 High Resolution Experiments

In this chapter, we will test whether we can further improve the rotation quality by increasing the image resolution. Tod do so, we will train the final model from chapter 7 again at an image resolution of 512×512 pixels instead of 256×256. Furthermore, we will also add some of the components from chapter 5 to take advantage of the faster model convergence and especially improve the image quality for unseen label combinations. In addition, we will increase the training duration from 5 million training images to 10 million, since the images at higher resolutions show considerably more detail, which might take longer to learn.

8.1 Combination Model Configuration

Instead of simply combining all successful components from chapter 7 and chapter 5, we question whether any component from chapter 5 could possibly harm the rotation quality. Consequently, we analyze each of the three conditional components that were used in the last experiment of chapter 5.

Label Dropout

The label dropout component mostly improved the general image quality for unseen label combinations. This was achieved by randomly removing some of the labels during the training. Due to its successful results, it should also be part of this combination experiment. Since, however, the removed rotation labels could affect the rotation loss, we decided to randomly remove all labels apart from the rotation label. This way, we make sure that the rotation components perform as similar as possible to the previous experiment.

Separate Label Mapping

Similar to the label dropout, we showed that the separate label mapping component also brings some image quality improvements when synthesizing with unseen label combinations. In addition, it also slightly reduced the entanglement between the labels. For both reasons it is desirable that this component is also used in this combination

experiment. However, after inspecting some rotation sequences of this model, it showed that the transitions between the discrete rotations looked very poor. An example for this is shown in Figure 8.1 (right). There, it demonstrates that the whole transition from one discrete rotation to the next gets squeezed into a very small interval, at which the car collapses. This observation can also be made with the rotation linearity metric. This metric scores a standard deviation of 0.0181, which is considerably higher than the result of the baseline model that only scored 0.0118. This indicates that the rotation sequence looks very inconsistent. Another way to visualize this is shown in the incremental perceptual distance graph (Figure 8.1 left). This graph adds up all perceptual changes during the rotation. It underlines that the perceptual distance, measured by a VGG16 model, increases very inconsistently, compared to the baseline model. Since this could also harm the rotation quality for the following combination experiment, we decided to completely remove the separate mapping component.

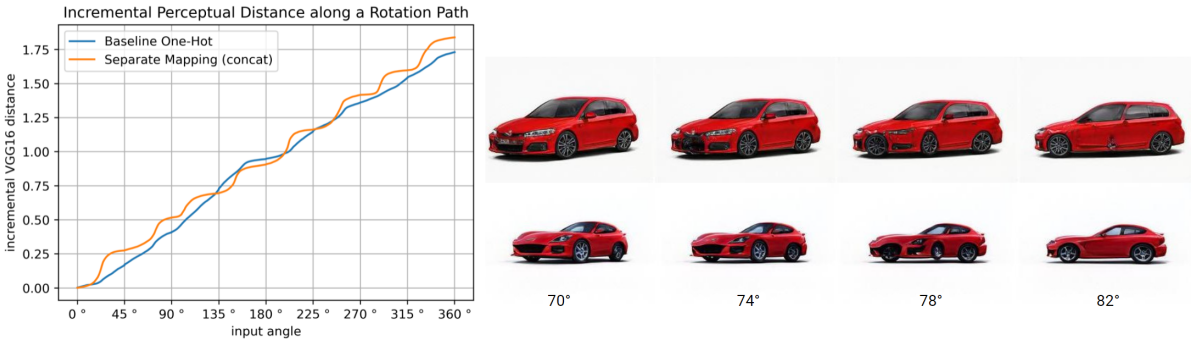


Figure 8.1: A demonstration of the inconsistent rotation speed, when synthesizing a rotation with the separate mapping component. The graph (left) shows the incremental perceptual rotation distance and the images (right) show two example rotations between 70° and 82°.

Label Information in the Discriminator

The last conditional component is the additional label information in the discriminator. This component showed to speed up the convergence of the FID considerably during the training. This led to an overall higher image quality. Therefore, it should be used in the combination experiment as well. However, since the component provides the rotation labels directly to the discriminator, which also has to predict this angle using the sine / cosine labels, we decided to remove the rotation label for this component. Otherwise it might bring the risk of the discriminator relying on the rotation input to predict the rotation angle. In such a situation, the generator would not be able to learn the rotations, as it does not receive any rotation feedback based on the generated images.

Training Configuration

With those modifications, the configuration of the combination experiment can be summarized as follows:

- 25% label dropout (without removing the rotation label) from subsection 5.7.2
- Label information in the discriminator (apart from the rotation label) from section 5.9
- Sine / cosine label from section 7.3
- 20% continuous rotation angles with a cooperative rotation loss from section 7.4
- Perceptual rotation regularization with a weight of 10.0 from section 7.5

8.2 Training Results

Since we changed the image resolution and training length, it is not reasonable to compare the metric results from the following experiment with the previous results. Therefore, we train the baseline model from chapter 5 at the increased resolution. In the following we will first compare both models with all metrics from this thesis and then focus on a qualitative analysis of the rotation improvements that come with the increased resolution.

Metric Results

In Table 8.1, we summarize the rotation metrics and conditional metrics of both models that were trained at an increased resolution.

Experiment	Baseline 512^2	Combination 512^2
FID ↓	**2.28**	3.49
FID Continuous Rotation ↓	**3.34** (+1.06)	3.61 (+0.12)
Rotation Accuracy ↑	92.0%	**97.0%**
Rotation Linearity ↓	0.0141	**0.0093**
Continuous Rotation Distance ↓	17.6°	**16.8°**
FID Randomized Labels ↓	**8.02** (+5.74)	8.33 (+4.84)
Conditional Accuracy ↑	**85.2%**	82.6%
Label Entanglement ↓	**26.0**	27.3

Table 8.1: An overview of the rotation metrics measured on the baseline model from the previous chapter. The second value of the FID metrics denotes the difference to the traditional FID result. (↓: lower is better, ↑: higher is better)

Generally, it shows that the baseline model performs better for image quality and conditional accuracy, while the combination model performs better for rotation quality. This is expected, as we already observed similar results in section 7.3. There, we explained this observation with the high value of the rotation loss balancing parameter that forces the model to focus more on rotation quality instead of image quality. This is likely the case for this combination experiment as well. Although all three FID results are higher for the combination model, we observe smaller increases between the traditional FID that uses labels from the dataset and both FID metrics that use modified labels. This indicates a more consistent image quality for randomized label combinations but also along the rotation paths. This correlates with the observed rotation quality shown in Figure 8.2, where we synthesize a rotation with the high resolution baseline model. It shows that the transitions between the discrete rotation angles look very poor. As this observation proved to be consistent for all rotations, we will, from now on, only focus on the rotation quality of the combination model and compare them with the final model from chapter 7.

Figure 8.2: Rotation examples from the high resolution baseline model. It shows very bad transitions between the discrete rotations, while the car is also rotated to the left at 0°. Both sequences show a rotation from 0° to 50°.

Rotation Improvements

After synthesizing a rotation with the high resolution combination model, the transitions look very good compared to all previous models at low resolutions. Similar to the final model from chapter 7, the rotations look very consistent and do not show any collapsing tires for the whole rotation. In addition, it shows that some of the problems from previous models have been reduced. Back then (in section 7.6), for instance, we observed that the rear car tire sometimes converted into exhaust pipes when rotating form the side view to the rear view of a car. This effect got completely removed using the higher resolution. An example for this is shown in Figure 8.3, which demonstrates that the rear tire slowly disappears instead of being converted into another car part. This makes the rotation look considerably more realistic. A possible explanation for this might be that the effect still exists using the increased resolution, however, at a much smaller level of detail. Therefore, this effect is much less visible and mostly affects smaller car parts.

Figure 8.3: A comparison of a rotation at the rear of the car, generated at low resolution and high resolution. The circle highlights the tire / exhaust pipe.

Another rotation improvement that comes with the higher resolution can be found when tracking the position of small textures, such as the manufacturer logo, during a rotation. There, it often showed that the logo is stuck at a specific position when training with the low resolution. An example for this is demonstrated in Figure 8.4. There it shows that the rings of the Audi logo are stuck in place. During the rotation, the model then appends new rings at one side, while the old rings disappear at the opposite side. This is no longer the case when training at a higher resolution. There, all four rings move along with the car. This problem is also often referred to as "texture sticking" in literature [Cha+21a]. This makes the rotation with the high resolution model look more realistic, as well.

Figure 8.4: A demonstration of the texture sticking effect. It shows that the Audi rings are stuck in place when using the lower resolution, while the move with the car when using the higher resolution.

Rotation Problems

Although the higher resolution generally improves the quality of the rotation, it shows that some problems from the previous models also exist at a higher resolution. For instance, we observe that in some examples, the car is slightly shifted to either side at 0°. An example for this is given in Figure 8.5. Similarly to subsection 7.4.2, we observe that the car is rotated to the right at 0°, while the actual front view is found at 12°. This problem is likely caused by the cooperative loss function, as we observed this effect with its introduction for the first time.

Figure 8.5: An example rotation, where the car is slightly shifted to the right. At 0° it shows the car from the 'front right' view, while the actual 'front center' view is located at 12°.

In some rare examples this problem even causes the car to collapse at the front view. This can be seen in Figure 8.6, where the front of the car is skipped and the rotation directly fades from one side view to the next, creating a very unrealistic rotation. This might indicate that the proportion of continuous angles during the training might still be too high, since such an effect increased using 40% continuous angles instead of 20%.

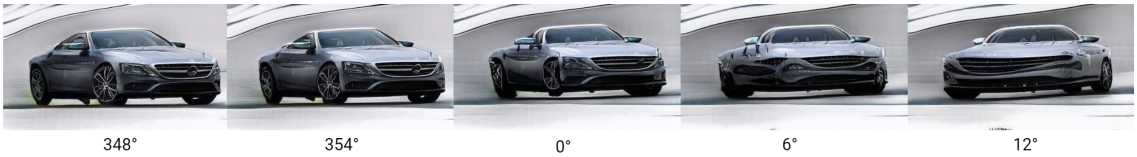


Figure 8.6: An example rotation, where the front of the car collapses.

Another problem that still exists at a higher resolution is the rotation around the rear of the car. Although we showed in Figure 8.3 that we do not observe the problem where a tire gets converted into an exhaust pipe any longer, the rotations still sometimes look very poor. An example for this is shown in Figure 8.7. There, the car tires fade in at about 186° causing the images to look less realistic.

Figure 8.7: An example rotation around the rear of a car, demonstrating slightly worse quality at 186° and 192°.

Summary

In this chapter, we showed that the model produces analogous results when increasing the image resolution. Similar to the experiment from section 7.3, in which we changed the rotation label, we observed that the general image quality was reduced, while the rotation improved. In this experiment we showed that in addition to the image quality, the conditional metrics were also reduced. We, however, hypothesize that the components from chapter 5 still slightly improve image quality for unseen label combinations, as the FID of the combination model increased less when using random label combinations instead of real labels from the dataset.
For the high resolution baseline model, we observed the same rotation problems as for the previous baseline model. It showed that the car collapses during most of the transitions between the discrete 45° angles, making the rotations look very unrealistic. Therefore, we only compared the rotations of the combination model with the final model of chapter 7. This showed that the higher resolution generally benefits the rotation quality. We observed much less problems where some textures were sticking to a specific location or where car tires got converted into other parts of the car. We hypothesize that those problems might still exist at the higher resolution, although they are much less visible. Some problems, on the other hand, were not solved by the higher resolution. For example, we observed similar problems to previous experiments where the front of the car was slightly shifted. Such problems were especially visible when using 40% continuous angles instead of 20%, which indicates that an even better rotation quality might be found if we lowered the proportion of continuous angles. This likely also holds true for other parameters, such as the weight of the perceptual rotation regularization, given that the VGG16 differences might be increased with the higher resolution model, even though we downscale the images before forwarding them to the VGG16 model.

9 Conclusion & Outlook

In this thesis, we proposed various modifications to the StyleGAN to improve conditional image synthesis for multi-labeled car images. In addition, we created a smooth 360° rotation of synthesized cars only using discrete rotation labels. Furthermore, we showed that the proposed methods scale well for high resolutions, providing a big advantage compared to similar GAN methods from literature that synthesize 3D representations of objects.

The two modifications from chapter 5 that have brought the largest improvements for the conditional image synthesis were label dropout and a separate label mapping network. In the label dropout method we randomly removed some of the labels and thereby created a considerably larger amount of distinct label combinations. This showed significant improvements for the image quality when synthesizing from new label combinations that did not exist in the dataset. In addition, the modification also removed dependencies between the labels. For the second modification, we added a second network to the generator that maps the labels independently from the random latent vectors. We hypothesized that this helps the model to find a better organization of the disentangled latent space that spatially separates the semantics of the labels. As a result, we observed a better image quality for unseen label combinations and a lower entanglement between the labels. At the end of chapter 5 we also showed that the proposed modifications work well in combination, although a more suitable conditional loss function for multi-labeled data might improve the conditional image synthesis even further. This is because we observed that the model mostly focused on the general image quality instead of synthesizing the correct conditional semantics. For future work it will be interesting to test how the modifications perform with alternative loss functions.

In chapter 7 we experimented with different approaches to create a 360° rotation with the discrete rotation labels from the car dataset. There, the two modifications that achieved the biggest improvement were the changed rotation label and the training with continuous rotation angles. In the first modification, the changed rotation label, we replaced the one-hot rotation label with a more suitable 2D vector that describes the position of the camera around the car using the sine and cosine of the rotation angle. This enabled us to design a regression loss for the rotation angle in both the generator and the discriminator that can be trained simultaneously with the adversarial loss. This improved the rotation considerably, as less images collapsed, while the transitions became significantly smoother. In the second modification, the training with continuous rotation angles, we used a semi-supervised training approach that enabled us to learn continuous rotation angles in the discriminator. To do so, we used the produced images from the generator as training samples for the discriminator. As a result, the synthesized

rotations were much less prone to creating collapsed images. At the end of chapter 7 we also showed that the proposed modifications work well in combination. There, we observed an even better rotation quality than in all previous experiments.
At last, we combined the modifications that improve the conditional image synthesis with the modifications that improved the 360° rotation view and showed that our methods scale well with higher resolutions. This brings a major advantage compared to other methods from literature, which often struggle with a bottleneck that is caused by an internal 3D image representation that comes with expensive operations.
An interesting future application of this model can be to combine it with the latent space gradient descent method from section 4.1. This could enable to take a real image of a car, re-synthesize it with our model, and then modify specific attributes or rotate it.
Although, we designed the modifications for the rotation from chapter 7 specifically for car images, we hypothesize that they also work for various other domains. Furthermore, we also believe that those techniques can not only be applied for spatial attributes, such as the camera location, but also for any other discrete label for which we would like to synthesize smooth transitions between classes. This enables numerous applications for 3D graphic design or the synthesis of animations from labeled images.

List of Figures

List of Tables

Bibliography

[3db] *3D-BoundingBox 3D-BoundingBox implementation*. https://github.com/skhadem/3D-BoundingBox. Accessed: 2022-02-22.

[Ban+21] Andrea Bandini et al. "A New Dataset for Facial Motion Analysis in Individuals With Neurological Disorders". In: *IEEE Journal of Biomedical and Health Informatics* 25.4 (2021), pp. 1111–1119. DOI: 10.1109/JBHI.2020.3019242.

[Cha+15] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. arXiv: 1512.03012 [cs.GR].

[Cha+21a] Eric R. Chan et al. "Efficient Geometry-aware 3D Generative Adversarial Networks". In: *arXiv*. 2021.

[Cha+21b] Eric R Chan et al. "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5799–5809.

[Den12] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[Dos+17] Alexey Dosovitskiy et al. *CARLA: An Open Urban Driving Simulator*. 2017. arXiv: 1711.03938 [cs.LG].

[Gau14] Jon Gauthier. "Conditional generative adversarial nets for convolutional face generation". In: *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester* 2014.5 (2014), p. 2.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 3354–3361.

[GMW16] Matheus Gadelha, Subhransu Maji, and Rui Wang. *3D Shape Induction from 2D Views of Multiple Objects*. 2016. arXiv: 1612.05872 [cs.CV].

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

[He+15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [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].

[Hä+20] Erik Härkönen et al. *GANSpace: Discovering Interpretable GAN Controls*. 2020. arXiv: 2004.02546 [cs.CV].

[Jun+21] Klaus Jung et al. *PicArrange – Visually Sort, Search, and Explore Private Images on a Mac Computer*. 2021. arXiv: 2111.13363 [cs.CV].

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

[Kat+19] Natsumi Kato et al. "Gans-based clothes design: Pattern maker is all you need to design clothing". In: *Proceedings of the 10th Augmented Human International Conference 2019*. 2019, pp. 1–7.

[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).

[Li+18] Haodong Li et al. "Can Forensic Detectors Identify GAN Generated Images?" In: *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2018, pp. 722–727. DOI: 10.23919/APSIPA.2018.8659461.

[LPT13] Joseph J. Lim, Hamed Pirsiavash, and Antonio Torralba. "Parsing IKEA Objects: Fine Pose Estimation". In: *2013 IEEE International Conference on Computer Vision* (2013), pp. 2992–2999.

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

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

[MNG18] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. "Which Training Methods for GANs do actually Converge?" In: *International Conference on Machine Learning (ICML)*. 2018.

[MO14] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[Mou+17] Arsalan Mousavian et al. *3D Bounding Box Estimation Using Deep Learning and Geometry*. 2017. arXiv: 1612.00496 [cs.CV].

[NG21] Michael Niemeyer and Andreas Geiger. "GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[OS19] Cedric Oeldorf and Gerasimos Spanakis. *LoGANv2: Conditional Style-Based Logo Generation with Generative Adversarial Networks*. 2019. arXiv: 1909.09974 [cs.LG].

[Roc19] Joseph Rocca. "Understanding generative adversarial networks (gans)". In: *Medium, Jan* 7 (2019), p. 20.

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

[Sel+17] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[Sho+21] Alon Shoshan et al. *GAN-Control: Explicitly Controllable GANs*. 2021. arXiv: 2101.02477 [cs.CV].

[Stya] *StyleGAN implementation GitHub (NVlabs)*. https://github.com/NVlabs/stylegan2. Accessed: 7.10.2021.

[Styb] *StyleGAN2 Benchmark paperswithcode.com benchmark*. https://paperswithcode.com/paper/analyzing-and-improving-the-image-quality-of. Accessed: 2022-01-27.

[Styc] *StyleGAN2 GitHub StyleGAN2 implementation*. https://github.com/NVlabs/stylegan2. Accessed: 2022-01-27.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

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

[TE11] Antonio Torralba and Alexei A. Efros. "Unbiased look at dataset bias". In: *CVPR 2011* (2011), pp. 1521–1528.

[Tom+15] Tatiana Tommasi et al. *A Deeper Look at Dataset Bias*. 2015. arXiv: 1505.01257 [cs.CV].

[Vas+17] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[Wah+20] Abdul Waheed et al. "Covidgan: data augmentation using auxiliary classifier gan for improved covid-19 detection". In: *Ieee Access* 8 (2020), pp. 91916–91923.

[Wu+17] Jiajun Wu et al. *Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling*. 2017. arXiv: 1610.07584 [cs.CV].

[Zha+19] Han Zhang et al. "Self-attention generative adversarial networks". In: *International conference on machine learning*. PMLR. 2019, pp. 7354–7363.

[ZKS21] Peiye Zhuang, Oluwasanmi Koyejo, and Alexander G Schwing. "Enjoy your editing: Controllable gans for image editing via latent space navigation". In: *arXiv preprint arXiv:2102.01187* (2021).