Neural Network: Projected GANs Converge Faster

Master in
Engineering in Computer Science

Emanuele Santo Iaia - 1924549
Daniele Calisi - 1752156

April 26, 2022

Contents

Chapter 1

Introduction

The goal of our project was to re-implement ProjectedGan [1], a Generative Adversarial Network with a new training method that achieves State Of Art level FID (metric explained later) in hours instead of days and works on even the tiniest datasets. More specifically, the new training method works by utilizing a pre-trained network to obtain embeddings for real and fake images that the discriminator processes. Additionally, feature pyramids provide multi-scale feedback from multiple discriminators and random projections better utilize deeper layers of the pre-trained network. In this report, we will focus on the implementation, the issues, the results and the general ideas behind them.

Chapter 2

ProjectedGAN

The discriminator in a GAN first projects samples into a shared latent space and then performs classification on their embeddings. The ProjectedGANs aim is to find a way to improve the quality of feedback that the discriminator provides.
The idea is based on 2 points:

1. Using a fixed pre-trained model as an encoder for **features projection**.
2. Using multi-scale discriminators for improving consistency by obtaining feedback at various resolutions.

2.1 Loss Function

The original paper introduces the loss function starting from the general one:

$$\min_G \max_D (\mathbb{E}_x[\log D(\mathrm{x})] + \mathbb{E}_z[\log(1 - D(G(\mathrm{z})))]) \tag{1}$$

Where:

- $D(\mathrm{x})$ is the discriminator's estimate of the probability that real data instance x is real.
- $\mathbb{E}_x$ is the expected value over all real data instances.
- $G(\mathrm{z}))$ is the generator's output when given noise z.
- $D(G(\mathrm{z})))$ is the discriminator's estimate of the probability that a fake instance is real.
- $\mathbb{E}_z$ is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).

The generator can't directly affect the log(D(x)) term in the function, so, for the generator, minimizing the loss is equivalent to minimizing $\log(1 - D(G(\mathrm{z})))$.

Starting from (1) the original paper introduce a set of feature projectors Pl which map real and generated images to the discriminator's input space and formulates the following Loss Function

$$\min_G \max_D \sum_{l \in \mathcal{L}} (\mathbb{E}_x[\log D_l(P_l(\mathrm{x}))] + \mathbb{E}_z[\log(1 - D_l(P_l(G(\mathrm{z}))))]) \tag{2}$$

Where:

- D_l is a set of independent discriminators operating on different feature projections.
- P_l is a set of projectors that map real and generated images to the discriminator's input space.

In the paper [1] the equation (2) has been proven to be consistent.

Chapter 3

Datasets

About the data, we used the few-shots datasets provided by the authors of FastGAN[2]. As the name suggests, few-shots datasets comprehend a series of small size datasets regarding various topics; we needed the datasets for our tests to be small, because of time and computational constraints.

3.1 Augmentation

Since we have chosen to use small size datasets, we needed a way to augment our data. Indeed, the performance of generative adversarial networks heavily deteriorates given a limited amount of training data, mainly because the discriminator is memorizing the exact training set. Therefore we decided to use a Differentiable Augmentation (DiffAugment), which has to be differentiable since gradients should be backpropagated. This augmentation consists in stochastic transformations of brightness, saturation, contrast, translation and cutout. Moreover, it is applied on both real and fake images for both generator and discriminator training, to avoid manipulating the target distribution.

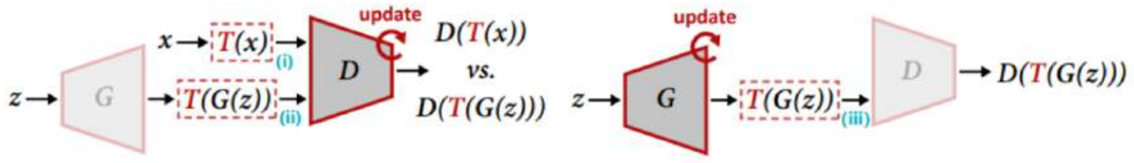


Figure 3.1: Overview of DiffAugment For updating D (left) and G (right). DiffAugment applies the augmentation T to both the real samples x and the generated output G(z). When we update G, gradients need to be back-propagated through T , which requires T to be differentiable w.r.t. the input.

Chapter 4

Architecture

Given the fact that we are dealing with a GAN, our baseline architecture it is characterized by a generator G, which tries to create fake images that are realistic enough to fool the discriminator D, which instead has to decide correctly if its inputs correspond to real or fake images. The three main news of our architecture are: the non-vanilla discriminator which instead is a Multi-Scale Discriminator, the Random Projections and the Pretrained Feature network.

4.1 Generator

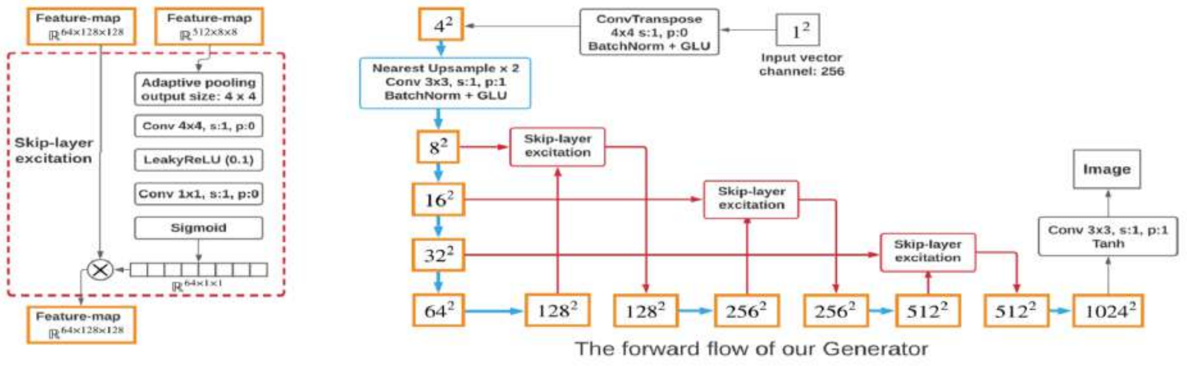

Feature-map $\mathbb{R}^{64\times128\times128}$
Feature-map $\mathbb{R}^{512\times8\times8}$
Adaptive pooling output size: 4 x 4
Conv 4x4, s:1, p:0
LeakyReLU (0.1)
Conv 1x1, s:1, p:0
Sigmoid
Skip-layer excitation
$\mathbb{R}^{64\times1\times1}$
Feature-map $\mathbb{R}^{64\times128\times128}$
4^2
ConvTranspose 4x4 s:1, p:0 BatchNorm + GLU
1^2
Input vector channel: 256
Nearest Upsample x 2 Conv 3x3, s:1, p:1 BatchNorm + GLU
8^2
16^2
32^2
64^2
Skip-layer excitation
Skip-layer excitation
Skip-layer excitation
128^2
128^2
256^2
256^2
512^2
512^2
1024^2
Conv 3x3, s:1, p:1 Tanh
Image
The forward flow of our Generator

Figure 4.1: Generator Model The structure of the skip-layer excitation module and the Generator. Yellow boxes represent feature-maps (we show the spatial size and omit the channel number), blue box and blue arrows represent the same up-sampling structure, red box contains the SLE module as illustrated on the left.

As the above figure shows the generator is the one used in FastGAN, which is built with standard layers like conv2d, convTranspose2d, batchNorm2d and GLU; but also with a new SLE block, that leverages low scale activations to revise the channel responses on high-scale feature-maps. SLE also allows a more robust gradient flow throughout the model weights for faster training.

4.2 Multi-Scale Discriminator

The multi-scale discriminator works on features obtained from several layers of the pretrained encoder that is explained in one of the next subchapters. For each of these layers, a separate discriminator is associated to the respective layer. Each discriminator is defined by a convolutional network with spectral normalization. Moreover, the all the discriminators output logits at 4x4 resolution, this is due the use of a DownBlock made by a conv2d, a batchNorm2d and a ReLU. The total loss is the sum of all logits of all discriminators.

4.3 Pre-Trained Network

The name ProjectedGAN derives by the fact that we are trying to use a pretrained encoder model to improve the GAN training. This means that our multi-scale discriminator will not have directly the real and fake images as input, instead it will have some feature projections that are the output of a pretrained encoder. In our case the pretrained model used was EfficentNets, which are image classification models trained on ImageNet and designed to provide favourable accuracy-compute trade-offs.

4.4 Random Projections

It is hypothesized that deeper layers of the pretrained model are significantly harder to cover, hence the discriminator can just ignore parts of the latent space altogether. Therefore we used two strategies to dilute prominent features, encouraging the discriminator to utilize all available information equally. The first one is Cross-Channel Mixing (CCM) to mix the features across the channels. CCM is implemented as a randomly initialized 1x1 convolution without activation. This random projection is applied at each scale before the features are passed to the discriminator. The other is Cross-Scale Mixing (CSM) to mix the features across the scales. CSM is implemented as a simple U-Net with random 3x3 convolutions and an up sample.

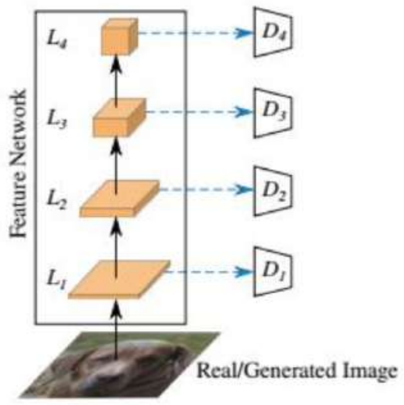


Figure 4.4.1: CCM (dashed blue arrows) employs 1×1 convolutions with random weights.

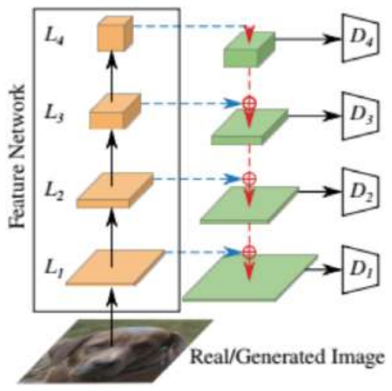


Figure 4.4.2: CSM (dashed red arrows) adds random 3×3 convolutions and bilinear upsampling, yielding a U-Network.

Chapter 5

Experiments

5.1 Metrics - FID

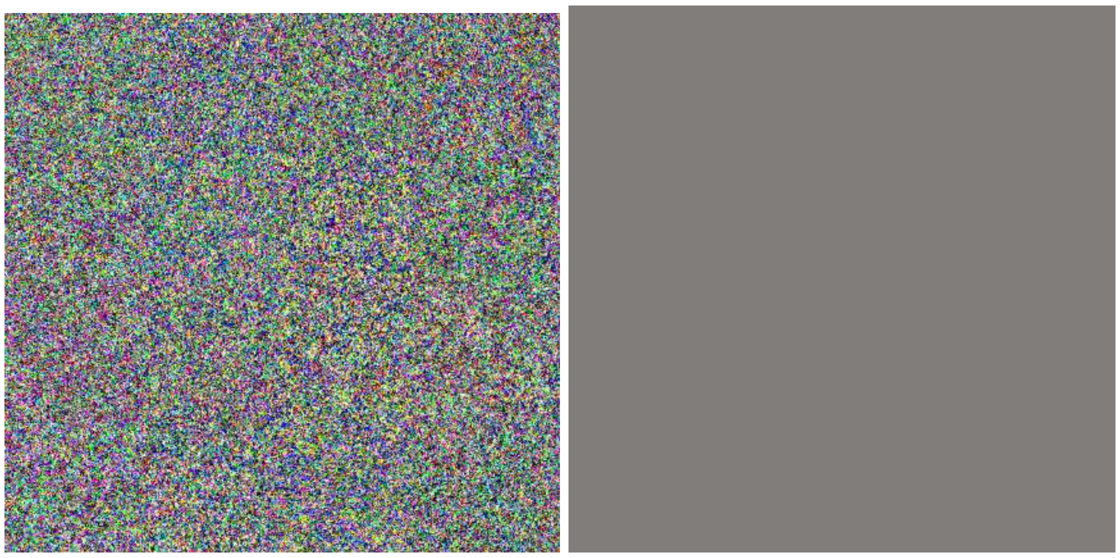
The Fréchet inception distance (FID) [3] is a metric used to assess the quality of images created by a generative model, like a generative adversarial network (GAN). Unlike the earlier inception score (IS), which evaluates only the distribution of generated images, the FID compares the distribution of generated images with the distribution of real images that were used to train the generator.

5.2 Training

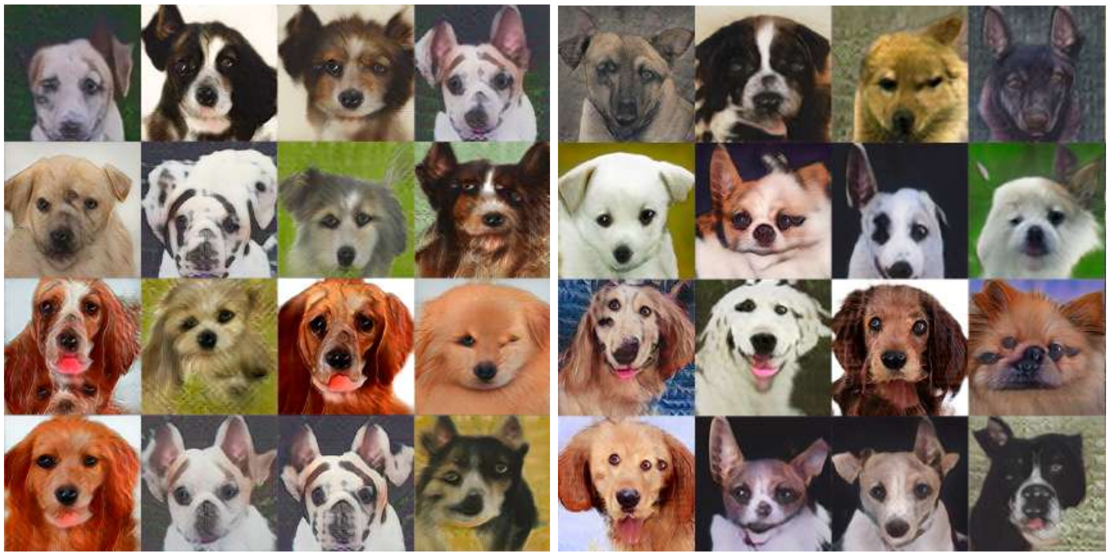
As described in the chapter [4] we carried out experiments using the FastGAN model as a generator and a Multi-Scale Discriminator as a discriminator. We kept in particular the number of layers to 4 using the same hyper-parameters as the paper. The specific datasets that we used from few-shots datasets are: "AnimalFace-Dog" (389 real-life dogs closeups, 256x256), and "AFHQ-Dog" (around 5000 high quality real-life dogs closeups, 512x512).

5.2.1 AnimalFace-Dog fake generated images

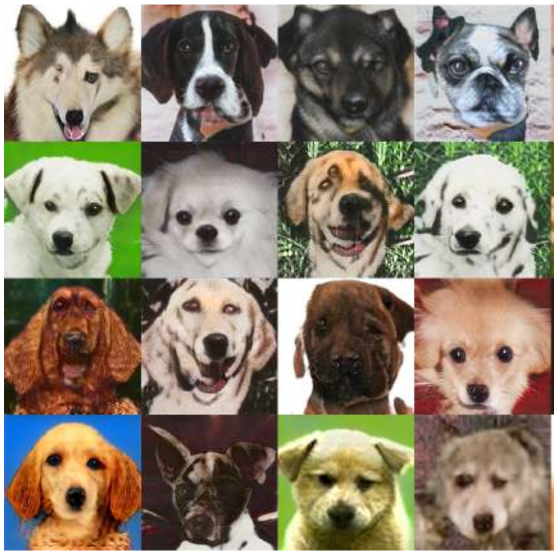
We will now show the results obtained by training over the AnimalFace-Dog dataset. Let's start by showing some fake images generated by our GAN after various epochs of training.



Initial State of fake generated images



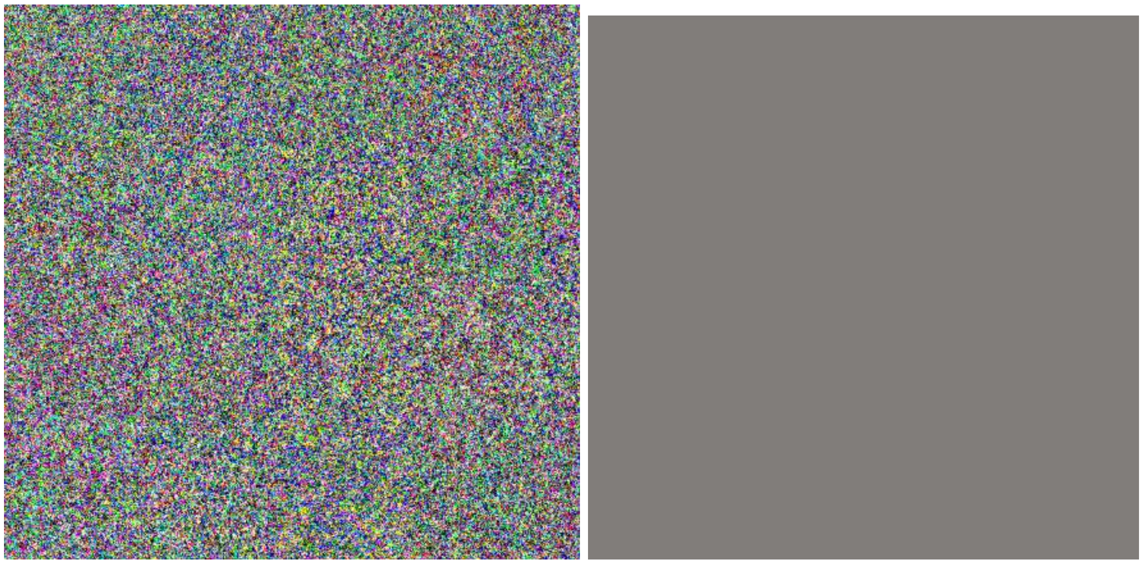
Fake images after respectively 30 and 60 epochs of training

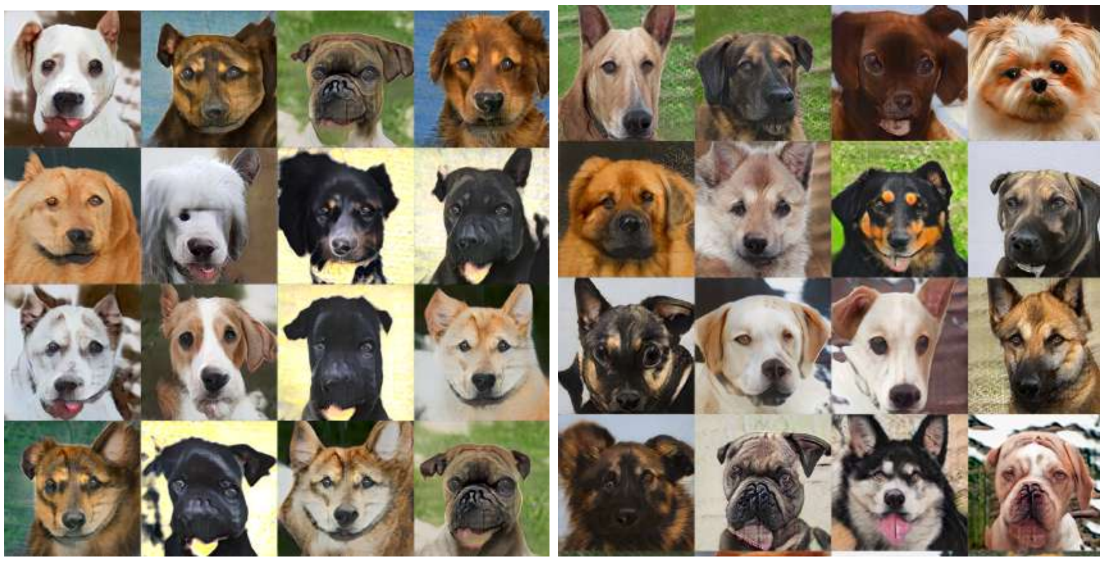


Fake images after 130 epochs of training

5.2.2 AFHQ-Dog

We will now show the results obtained by training over the AFHQ-Dog dataset. Let's start by showing some fake images generated by our GAN after various epochs of training.

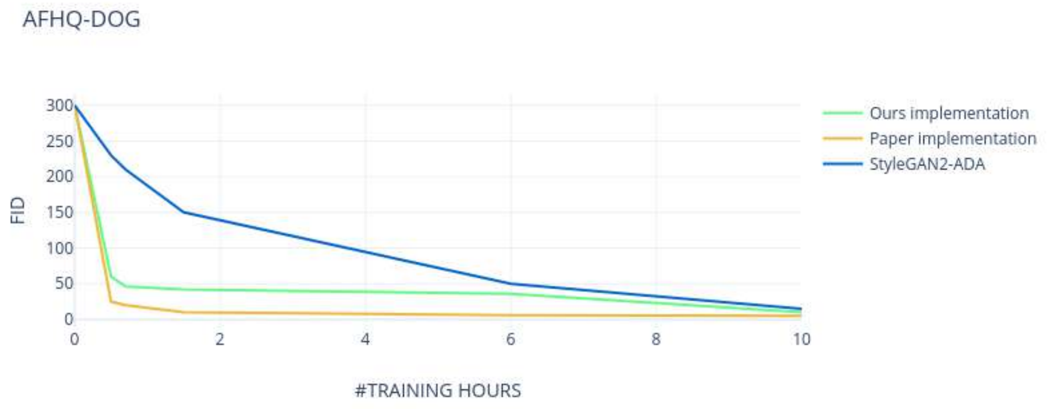

Initial State of fake generated images


Fake images after respectively 30 and 60 epochs of training

Fake images after respectively 90 and 100 epochs of training

5.3 FID Results

Let's show now some plots which intuitively represent the improvement of our GAN throughout the training. Indeed, in the plots is shown the FID descending pattern w.r.t the number of training hours or epochs:



AnimalFace-dog

	FID	IMGS
FastGAN	62.11	200k
ProjectedGAN	58.07	20k
OUR IMPLEMENTATION	50.27	40k

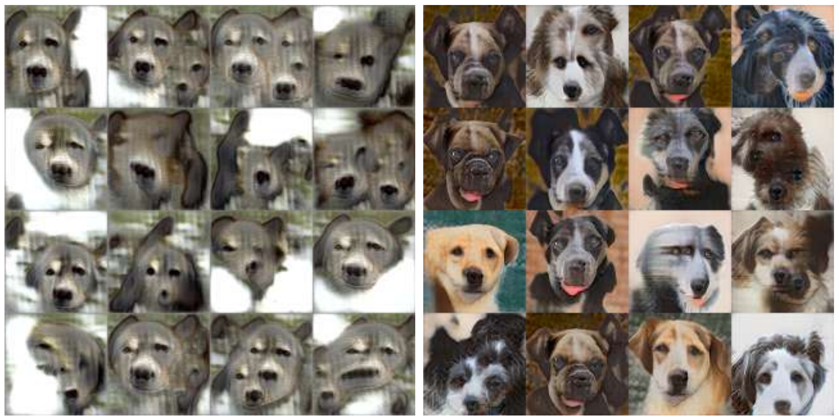
AFHQ-dog

	FID	IMGS
StyleGAN2-ADA [4]	7.4	10M
ProjectedGAN	7.10	900k
OUR IMPLEMENTATION	6.23	1M

5.4 Changing Hyperparameters

Since the paper used the same hyperparameters in all the training runs, to highlight the speed of convergence compared to the other GANs, we decided to carry out some trainings with modified hyperparameters to optimize the results obtained.
Therefore, we trained for 20 epochs changing the hyperparameters suggested by the paper and we got the following results.

5.4.1 Adam Betas = 0.9, 0.999



Fake images from left to right: betas = 0.9, 0.999 VS original paper betas= 0.0, 0.99

The training for betas = 0.9, 0.999 takes around 45 minutes 20 min more than the original one. Moreover, the FID is very high 145.6 (original paper FID after 20 = 53.1)

5.4.2 Learning Rate = 0.001, 0.003

Fake images from left to right: lr = 0.001 VS lr = 0.003

- For **lr = 0.001** we found no improvement both in terms of time and FID.
- For **lr = 0.003** we found an improvement in terms of FID in fact after 20 epochs the FID = 51.96 (with lr=0.002 the FID was 53.1).
  For this reason we decided to carry out another 20 epochs, to confirm the trend. After 40 epochs with lr = 0.003 we found a further improvement with FID = 30.64 (with lr=0.002 the FID was 35.88).

	New FID	Previous FID
Betas = 0.9, 0.999	145.6	53.1
Lr = 0.001	56.35	53.1
Lr = 0.003	51.96	53.1

Once had very interesting results using lr=0.003, we applied the same lr to the other dataset (AnimalFace-Dog) which is smaller than the other. As we can see from the table above, with a smaller dataset the

	New FID	Previous FID
AFHQ-Dog	30.64	35.88
AnimalFace-Dog	55.34	50.27

performance is worse than that of the paper. This leads us to infer that GAN hyperparameters are **dataset dependant**.

Chapter 6

Conclusion

After studying and re-implementing this Generative Adversarial Network we learned how important the discriminator is who is often put in a second place respect to the generator. In fact, we have seen how, by going to work on the discriminator, we can achieve much higher results both in terms of quality and in terms of training time compared to competing GANs that focus on the development of the generator.

Bibliography

[1] Axel Sauer, Kashyap Chitta, Jens Müller, Andreas Geiger (2021) *Projected GANs Converge Faster.* `https://arxiv.org/pdf/2111.01007v1.pdf`

[2] Bingchen Liu, Yizhe Zhu, Kunpeng Song, Ahmed Elgammal (2021) *Towards Faster and Stabilized GAN Training for High-fidelity Few-shot Image Synthesis.* `https://arxiv.org/abs/2101.04775`

[3] Eric J. Nunn, Pejman Khadivi, Shadrokh Samavi (2021) *Compound Fréchet Inception Distance for Quality Assessment of GAN Created Images.* `https://arxiv.org/pdf/2106.08575.pdf`

[4] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila (2019) *Analyzing and Improving the Image Quality of StyleGAN.* `https://arxiv.org/abs/1912.04958`