

# Assignment 2 - Applied Deep Learning WS 22/23

November 2022

Maximilian Viehauser - 11945353

**Topic of the project:** *Contemporary Art Generator*

**Type of the project:** *Bring your own data AND Bring your own method*

## Changes since ex1:

In exercise 1, one of the focuses was on creating my own dataset on contemporary artists. This turned out to be time intensive and repetitive (e.g. deleting unsharp images, cropping away borders manually for thousands of images). I created a dataset of around 1000 cropped images in a quality of 512x512 of contemporary artists.

As I thought, that for a good art generator a huge amount of images is necessary, which would be incredible time intensive to create, I decided to stop working on this dataset and continue to look for existing ones. Luckily, I found one by wikiart (<https://www.kaggle.com/datasets/ipythonx/wikiart-gangogh-creating-art-gan>) with around 100.000 paintings (37GB) of all kind of epochs. So I decided to change my focus. Take the existing dataset and add my acquired data. Since, I realised that the model will be way better and more creative with such an high number of images. Furthermore, the big dataset can also be just used for pertaining, and specific image groups can be used for specialised training in the end.

## Computational setup:

The idea was to execute the training on a local workstation which I mainly built for deep learning training a short time ago. This workstation contains a GeForce RTX 3070, 32 GB of RAM and a AMD Ryzen 5 3600. One difficult and time intensive part of this project was the setup of this workstation. Drivers for the GPU, CUDA etc had to be correctly installed for the GPU acceleration of the training. This led to several freezings of the system, but was successfully achieved after half a day of work.

## Building the model:

As mentioned in ex1 the goal was to use a ProGAN for this project as it is a good tradeoff between high quality results, but a complexity which is still understandable for me as a Deep Learning beginner. I have found a excellent implementation of a ProGAN by Aladdin Persson (<https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/GANs/ProGAN>) which I decided to use for this project. My goals were to understand the implementation, so that I can adjust parts of it, and apply it on my problem. I initially wanted to code most of the GAN by myself but realised, that it is infeasible in the given amount of time. That's why I decided to use Aladdins code and set my focus on creating a program which produces great art.

## Preprocessing of the data:

A preprocessing pipeline was setup, which crops all pictures in a 512x512 format. Additionally Torch.RandomResizedCrop was used to get different crops, different scales and ratios for data augmentations. This finally led to a dataset of 400.000 images. (100.000 images cropped in 4 different ways)

## Defining an error metric:

For this project it is hard to define an error metric, since the goal is to create art. And especially for abstract art, deciding if a piece of art is good or bad is subjective.

Since the training set is huge and training takes several days for larger images sizes for my setup, I decided to use the following training setup:

Train the model for hyperparameter tuning only on the pictures of the genre „Landscapes“ until a quality of 128x128. Then look at 100 randomly generated samples by the trained network and manually evaluate how many of those 100 images look like paintings of a landscape. This leads to some „accuracy“ rate - e.g. 5 of 100 images look like a painted landscape. I decided that this metric fits as an error metric for my task.

Obviously, this metric is quite subjective and one could dispute with others if an image represents a painted landscape or not, but in the case of this project I think that this error metric was totally sufficient as we will see in later steps.

Also it can be critiqued that, the landscape dataset is not representative for the whole dataset and thus might not be ideal for hyper parameter tuning (e.g. z-dimensions of latent space), still I thought that this error metric and hyper parameter tuning setup was the best factoring in my computational and time restrictions.

So to conclude, the error metric of choice is a ratio: „of 100 randomly generated pictures, how many look like a painted landscape“.

$$accuracy = n_{samples\ look\ real} / n_{samples}$$

## Hyperparameter tuning:

For the parameter tuning I did first setup a baseline. This was the setup which Aladdin Persson did use for his training. Furthermore the authors of the original ProGAN paper did use a similar setup for their training of a face generator. (Main difference was the size of z dimension and in channels was 512 each) So the following baseline was defined.

Standard setup:	
Learning rate	1E-03
Z Dimension	256
In Channels	256
Activation function	LeakyReLU
Epochs	30

Next, over the course of several weeks I changed several parameters of this baseline model and documented the results for the different setups. First, I decided to train until a size of 256x256, but after several tests I realised that this takes very long and a size of 128x128 is sufficient to rate the different methods. Thats why I switched at some point to 128x128.

Below are the results for the different tests. In the appendix you can see some sample images of the four tests were the training worked out without mode collapse.

	Changes:	Accuracy:	Time till 128x128:
Test 1	Standard setting	18/100	NA
Test 2	Changed lr of standard setup to 1e-4	5/100	NA
Test 3	Changed standard setup to dilated convolutions	Training stopped due to mode collapse	NA, but faster than non dilated

	Changes:	Accuracy:	Time till 128x128:
<b>Test 4</b>	Changed standard setup to dilated convolutions and lr to 1e-4	4/100	NA, but faster than non dilated
<b>Test 5</b>	Changed leakyReLU of standard setup to GeLU	Training stopped due to mode collapse	9:20
<b>Test 6</b>	Changed leakyReLU of standard setup to SeLU	Training stopped due to mode collapse	9:20
<b>Test 7</b>	Changed leakyReLU of standard setup to SeLU and lr to 1e-4	4/100	8:20
<b>Test 8</b>	Changed in channels to 512	Training stopped due to mode collapse	16:00

The different tests showed, that ProGANs are very sensitive when it comes to hyperparameters. While the standard setting was not the fastest (exact time missing, since it was trained til 256x256), it produced by far the best accuracy.

#### Notes on time and electricity costs of hyperparameter tuning:

As one can see, the training time for only one hyperparameter test was quite high with around 10 hours per test. I tracked the electricity costs for the hyper parameter training. The costs for the whole hyperparameter tuning were above 50 kWh. This is a high financial cost. Assuming one kWh costs 35 cents, this is more than 17,5 Euros. This is also a high environmental burden, since for every kWh a significant amount of CO2 is emitted. Furthermore, the training is very time consuming.

For that reason it is greatly important to plan hyperparameter tuning in great detail, and design it in an efficient way to avoid unnecessary financial, environmental and time costs!

#### **Outcome:**

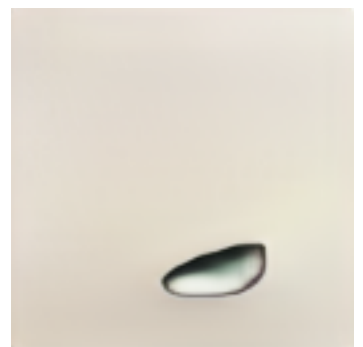
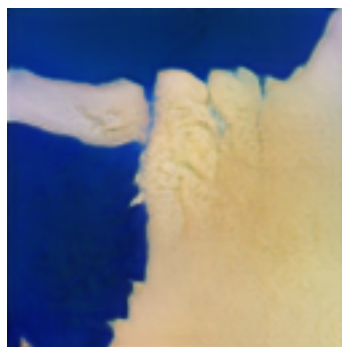
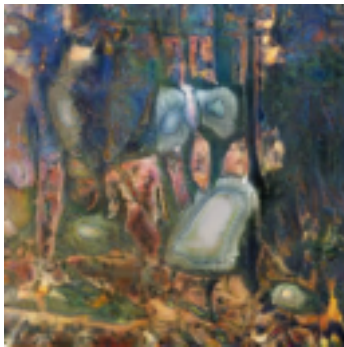
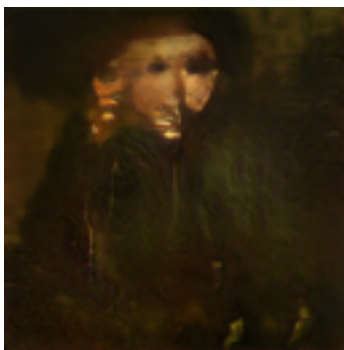
After the hyperparameter tuning I decide to try the baseline setting on the full dataset. Here, it can be critiqued that in-channels and z-dim are rather low for the given task. I decided, to keep those values at 256, since otherwise I would face VRAM issues at some point and the training time would heavily increase. Furthermore, I found a paper which states, that you can also reach good results with a low z-dimension.<sup>1</sup>

So I started to train the model on the augmented full dataset. The first tries with different learning rates led to divergence or mode collapse after some time. Stable training was achieved with the following setup:

```
LEARNING_RATE = 1e-4
BATCH_SIZES = [32, 32, 32, 24, 16, 8, 8, 4, 2]
Z_DIM = 256
IN_CHANNELS = 256
```

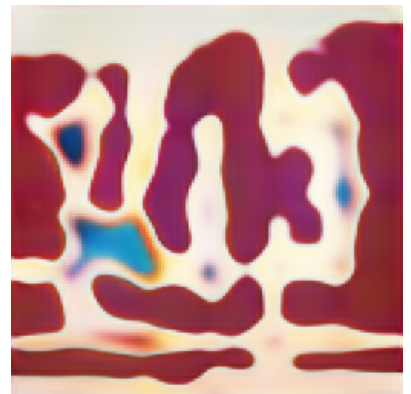
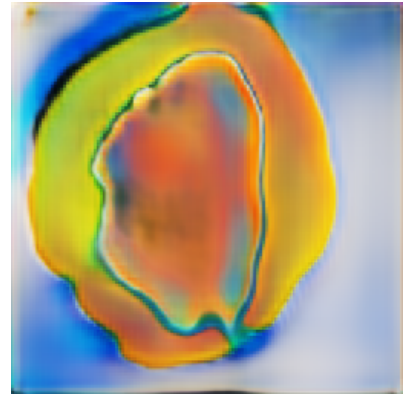
<sup>1</sup> - Marin, Ivana & Gotovac, Sven & Russo, Mladen & Božić-Štulić, Dunja. (2021). The Effect of Latent Space Dimension on the Quality of Synthesized Human Face Images. Journal of Communications Software and Systems. 17. 124-133. 10.24138/jcomss-2021-0035.

At the point of writing this I trained the model until a size of 128x128 (training time: 5days and 13 hours). I will continue the training till 256x256 or 512x512. Here are some sample images of the model I trained on the full dataset:



The results are really interesting. It can be seen that the model is trained on a very big dataset. We can see structures of faces, but also of trees or colourful abstract art pieces. As some of the results during training were already very aesthetic and in my opinion fulfilled the criteria for innovative contemporary art, I think the current results are very successful.

I later tried to run the ProGAN only on the images of the category abstract from the wikiart dataset. The training time was obviously way shorter, since the size was only 15000 images. Still the results were great, as seen below:



For the abstract dataset the variety of the generated images seemed rather low. Similar pictures were appearing often. I think that this issue could have been improved with a bigger dataset, a higher value for the channels or a larger z dimension.

## Challenges and learning:

### My main challenges in this task:

- Sensitivity of GAN to hyperparameters (I encountered mode collapse many times)
- Training time!
- Finding of error measure

Luckily, I was able to solve the mode collapse issues without big changes in the overall model architecture. A different setting of several hyper parameters was enough. Still, I would encourage everyone who wants to build a GAN to consider this issues while planning the architecture of the GAN and read about novel research on how to tackle mode collapse.

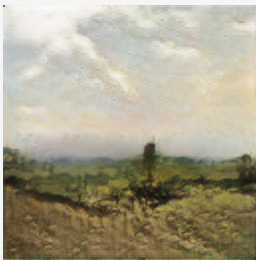
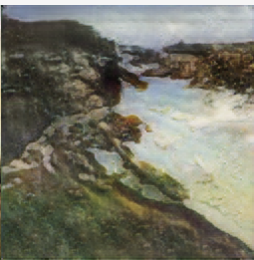
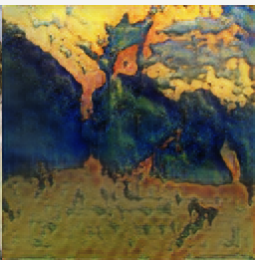
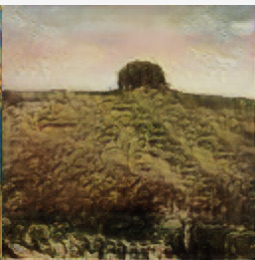



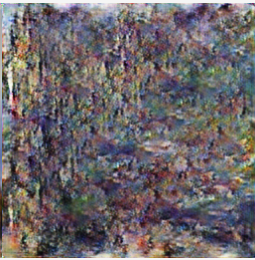




Furthermore, the long training time made the whole tuning process not only time consuming but overall difficult. The different hyper parameter setups should be planned well. Regular checks of the training should be done, that in case of any errors you can stop the GAN and start a new training.

### Work breakdown:

Task	Number of hours
Ex1 - Researching and writing of assignment 1	10
Ex1 - Dataset collection and preparation	15
Ex2 - Understanding and adjustment of ProGAN implementation	5
Ex2 - Setup of work environment (installing Nvidia GPU drivers, CUDA, libraries, remote control of workstation)	7
Ex2 - Hyperparameter tuning and its supervision (!)	20
Ex2 - Documentation/writing report	4
Estimation Ex3 - Building an application to present results	15
Estimation Ex3 - Writing final report	5
Estimation Ex3 - Creation and preparation of Presentation	5



Appendix:

Sample images of the four successful tests:			
Test 1			
			
Test 2			
			
Test 4			
			
Test 7			
