Faulty of Mathematics and Information Science
Warsaw University of Technology

Project documentation

Generative Adversarial Networks

Subject: Deep Learning

Authors:
Marcin Łukaszyk, Patryk Wrona

Warsaw
2022

Contents

1. Introduction

1.1. Description of the research problem

In this laboratory project, generative adversarial network architectures serving the purpose of generating new images are investigated. All of the experiments must have been conducted on kaggle's lsun bedroom dataset. [1] It is divided in numerous directories containing on the whole about 300 thousands of bedroom colour images of different size. Each image was reshaped to 64 x 64 x 3, where 3 is the number of color channels. After this step, the images were scaled to (-1, 1).

In this work, the impact of different architectures and hyperparameters of Generative Adversarial Networks (*GAN*) was verified. First, each *GAN* model was trained using Google Colab. The assessment of the models was performed qualitatively (investigating generated images visually) as well as with the use of Frechet Inception Distance (*FID*) metric that compares distributions of real and newly generated images. In the literature [4], FID of 20-30 are considered as very good scores, and after testing once a random model, it achieved scores about FID = 2000, that is why any FID scores between 20-200 could be considered as 'satisfactory enough' for this dataset. As an additional experiment, having 2 latent vectors being input vectors of a well performing GAN model, generated images using the linear interpolation were created.

The authors used 2 different architectures of *GAN* models, along with tuning some of their hyperparameters, changing their number of layers or adding some improvements in order to seal with mode collapse:

— DCGAN – based on [2]
— WGAN-GP – based on [3]

1.2. Instruction of the application – results' reproduction

In order to reproduce the results covered in this report, one must use python notebooks attached to this report:

— **DL3-Final.ipynb** - file containing the final implementation details of DC-GAN architecture [6]
— **DL3_Gan_L01ep20.ipynb** - file containing the experimenting details of DC-GAN architecture with lower learning rate and more epochs [5]
— **WGAN_part6.ipynb** - file containing the implementation and testing details of the first of 2 WGAN-GP architectures
— **WGAN2_part3.ipynb** - file containing the implementation and testing details of the second of 2 WGAN-GP architectures

2. Network Architectures

2.1. Theory - Generative Adversarial Networks

Generative Adversarial Networks (*GANs*) are an example of an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate new observations. Its structure is divided in 2 sub-models: a generator and a discriminator. The generator's task is to learn how to generate new images basing on given input vector that could be drawn at random. The discriminator is a critic that learns to discern fake (generated by the generator) and real images and gives the feedback to the generator. In this deep learning scenario the one sub-model is trying to outsmart the other sub-model.

A *DCGAN* is a direct extension of the *GAN* described above, except that it explicitly uses convolutional layers in the discriminator and generator, respectively.

The *WGAN* leverages the Wasserstein distance to produce a value function that has better theoretical properties than the value function used in the original GAN paper. WGAN requires that the discriminator lie within the space of 1-Lipschitz functions.

The *WGAN-GP* method proposes a "gradient penalty" by adding a loss term that keeps the L2 norm of the discriminator gradients close to 1.

2.2. DCGAN

This network architecture is a "normal" GAN that strictly uses convolution layers. We use The discriminator:

— **Input**: (64, 64, 3)
— **Conv2d**: (128, 128, 64)
— **Conv2d**: (256, 256, 128)
— **Conv2d**: (512, 512, 256)
— **Flatten**: 5184
— **Dense**: 1 (no activation function)

The generator:

— **Input**: 100
— **Conv2d**: 100, 100, 512 with batch normalization
— **Conv2d**: 512, 512, 256 with batch normalization
— **Conv2d**: 256, 256, 128 with batch normalization
— **Conv2d**: 128, 128, 64 with batch normalization
— **Conv2d**: 64, 64, 3 with batch normalization
— Activation function: **relu**

2.3. WGAN-GP (1)

This network architecture is based on [3], small differences were made in order to adjust the model to our image size. Batch size = 128, latent vectors dimension = 128, gradient penalty weight = 10. The architecture of this model is as follows.

The discriminator:

— **Input**: (64, 64, 3)
— **Conv2d**: (34, 34, 128)
— **Conv2d**: (17, 17, 256)
— **Conv2d**: (9, 9, 64)
— **Flatten**: 5184
— **Dense**: 1 (no activation function)

The generator:

— **Input**: 128
— **Dense**: 8192
— **Conv2d**: 16, 16, 128 with batch normalization
— **Conv2d**: 32, 32, 256 with batch normalization
— **Conv2d**: 64, 64, 3 with batch normalization
— Activation function: **tanh**

2.4. WGAN-GP (2)

This network architecture is based on [3]. Comparing to the previous model described in section 2.3, 0.2 dropout layers were added after each convolutional layer of generator, latent vector size was increased 8 times, and discriminator received noised real and fake images in order to alleviate the influence of mode collapse. This noise came from normal distribution with mean = 0, and standard deviation = 0.001 as not to make huge impact on images having values between -1 and 1. Batch size = 128, latent vectors dimension = 1024, gradient penalty weight = 10. The architecture of this model is as follows.

The discriminator:

— **Input**: (64, 64, 3)
— **Conv2d**: (34, 34, 128)
— **Conv2d**: (17, 17, 256)
— **Flatten**: 18496
— **Dense**: 1 (no activation function)

The generator:

— **Input**: 1024
— **Dense**: 128
— **Conv2d**: 16, 16, 256 with batch normalization
— **Conv2d**: 32, 32, 64 with batch normalization
— **Conv2d**: 64, 64, 3 with batch normalization
— Activation function: **tanh**

3. Results

3.1. Results - DC-GAN

DC-GAN model converged quite quickly – 5 epochs of training lasted for about 1 hour and the resulting generated imaged are very satisfactory. After further training, the DC-GAN did not improve over additional epochs. Nevertheless, increasing learning rate in this case led to better results in terms of generated images and FID. The best images generated by the DC-GAN model are shown in figure 3.1 and the loss function of generator and discriminator of this model is presented in figure 3.2.

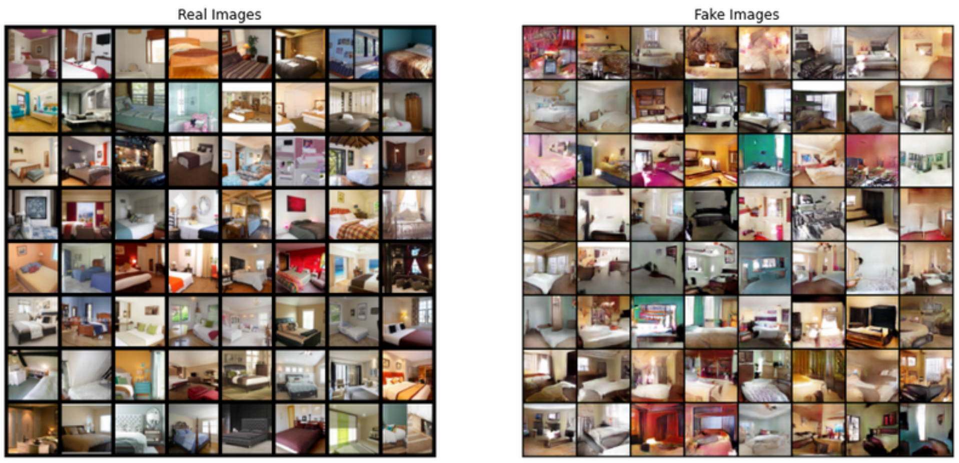

Figure 3.1: Real images compared to fake images generated by DC-GAN model's generator

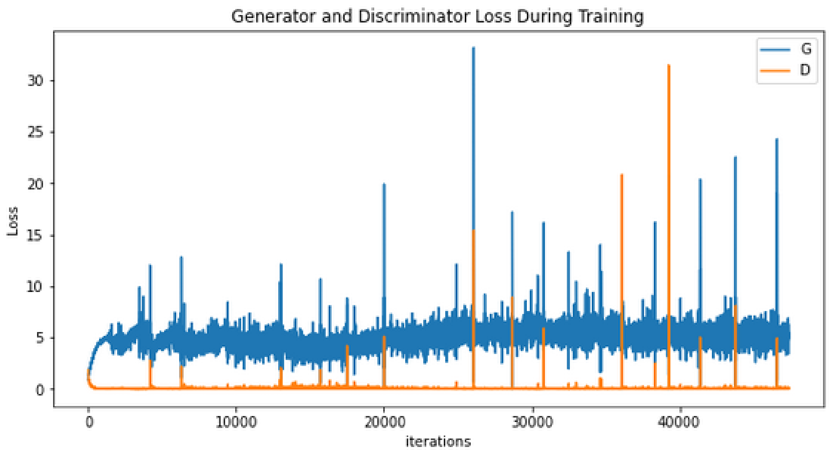

Figure 3.2: Loss function of generator and discriminator of DC-GAN model, G - generator, D - discriminator

3.1.1. Images generated from latent vector's interpolation

Having 2 latent vectors, 2 satisfactorily looking images were generated using DC-GAN's generator model. Next step included linear interpolation between these 2 vectors – 8 latent vectors right between the previous 2 vectors were generated. Using these additional 8 vectors, 8 images were generated. These images are presented in figures 3.3, 3.4 .

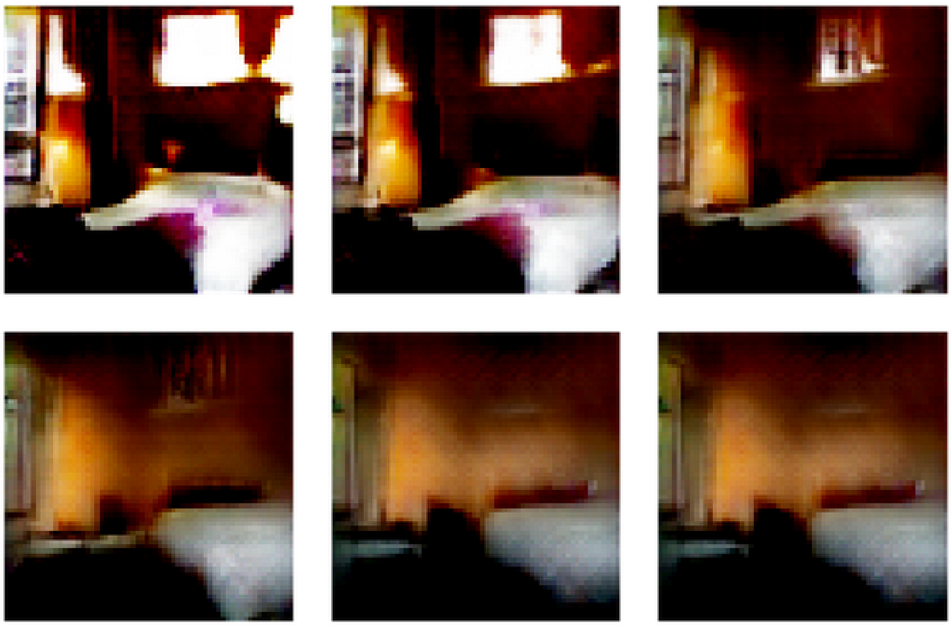
Figure 3.3: Images generated from latent vectors' interpolation - 1

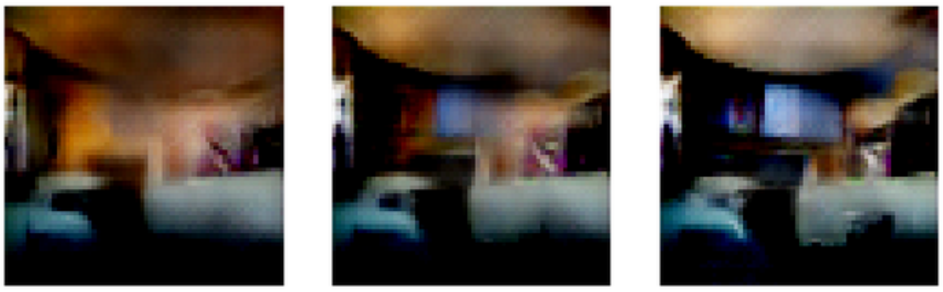
Figure 3.4: Images generated from latent vectors' interpolation - 2

The video showing changing interpolated images could be found at this page: VIDEO

3.2. Results - WGAN-GP (1)

Training of model having this architecture was very long – 1 epoch on the whole dataset lasted for about 1 hour. The whole process of training involved about 15 epochs (when cases of increasing learning rate are added, it would sum up to 16 epochs, but the result from this 1 additional epoch was abandoned because of much worse performance in FID score). The x axis on figure 3.5 is arbitrary – the models were saved and assessed after almost same time intervals. The last 3 values refer to training WGAN with lower learning rate (lr = 0.00002 instead of 0.0002) and were made on 1/2 of epoch for testing if FID values would further decrease.

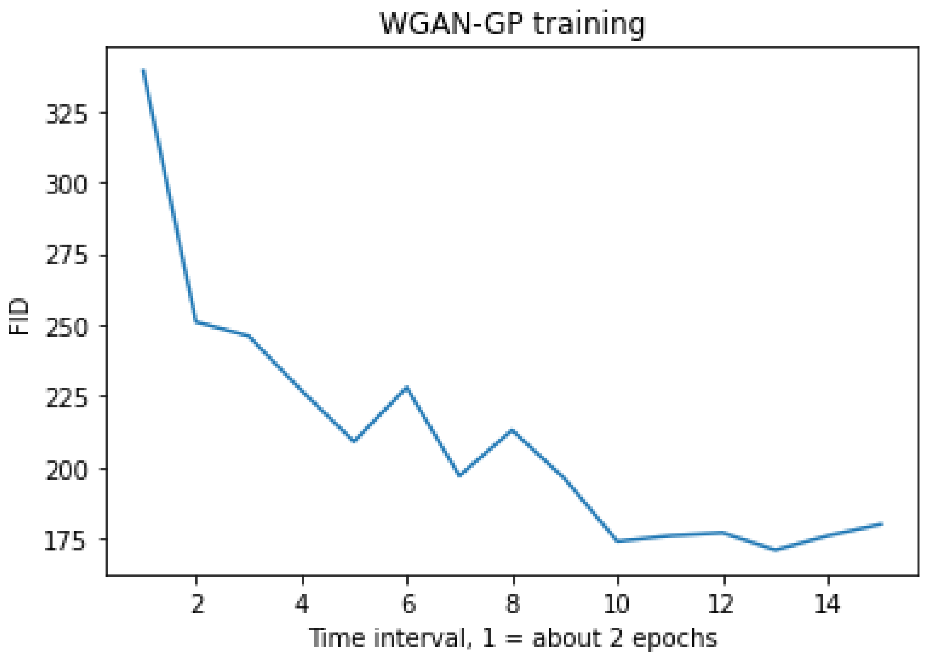

Figure 3.5: Change of FID score in function of number of epochs of training - WGAN-GP

Unfortunately, despite having very low (meaning good) FID score of 170, the last model did not generate satisfactory images. Examplary generated image is shown in figure 3.6.

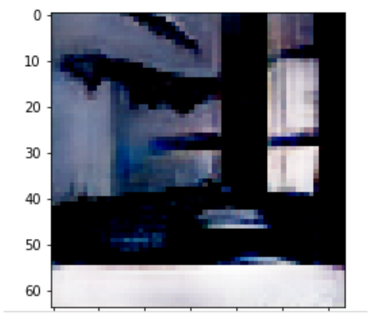

Figure 3.6: Exemplary generated image by WGAN-GP (1) having FID about 170

3.3. Results - WGAN-GP (2)

Training of this additional model having different (as compared to the first version of WGAN-GP) architecture was quite quick – 1 epoch on the whole dataset lasted for about 15 minutes, meaning it was about 4 times faster than the previous architecture. The training as shown in figure 3.7 lasted for 10 epochs.

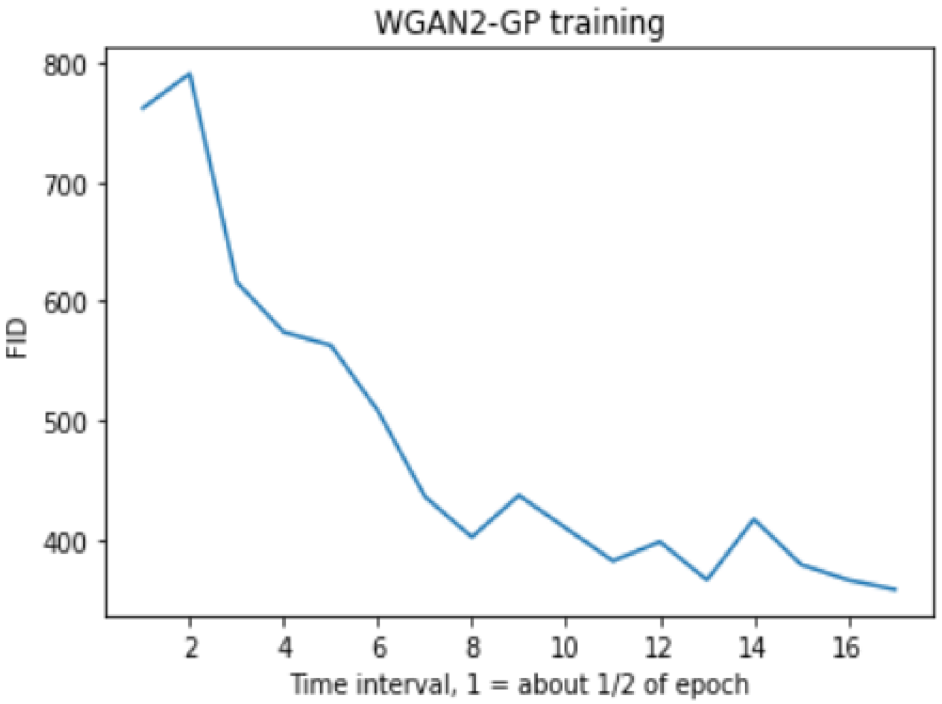

Figure 3.7: Change of FID score in function of number of epochs of training - WGAN2-GP

Unfortunately, despite having comparable to DCGAN FID score of about 350, WGAN-GP (2) model did not generate satisfactory images. Examplary generated image is shown in figure 3.8.

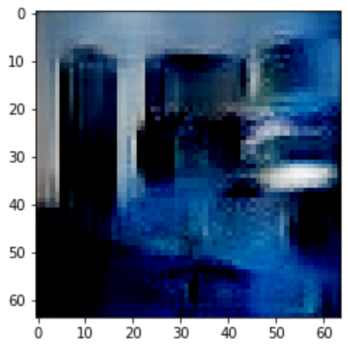

Figure 3.8: Exemplary generated image by WGAN-GP (2) having FID about 350

4. Conclusions

The results obtained by GAN models are generated images. Despite the fact of having quite high FID score (about 300), the DCGAN model obtained satisfactory images for this dataset as could be seen by examples in chapter 3.1.

As far as interpolation is concerned, it is very clear that after interpolating latent vectors, generated images seem to transform one to another in a smooth way. It gives some notion about the space of images being also linked with the space of latent vectors used as input vectors for generator.

Bibliography

[1] LSUN bedroom scene 20% sample
https://www.kaggle.com/datasets/jhoward/lsun_bedroom

[2] DCGAN for Beginners
https://www.analyticsvidhya.com/blog/2021/07/deep-convolutional-generative-adversarial-network-dcgan-for-beginners/

[3] Wasserstein GAN with Gradient Penalty
https://keras.io/examples/generative/wgan_gp/

[4] Best values of FID on lsun bedroom dataset
https://paperswithcode.com/sota/image-generation-on-lsun-bedroom-256-x-256

[5] DCGAN-20epochs
https://colab.research.google.com/drive/1XwKybwMDqCawqvMPphrrejI4Ejeena3h?usp=sharing

[6] DCGAN Final best model
https://colab.research.google.com/drive/12whoSA5ZLqzpzQXEgPefmMjRouhsyt5P?usp=sharing