DEEP LEARNING 2022

IMAGE GENERATION WITH GENERATIVE ADVERSARIAL NETWORKS PROJECT NO. 3 REPORT

June 14, 2022

Elżbieta Jowik (298821), Agata Makarewicz (298827)

1 Introduction

The objective of this document is to describe the methodology and summarise the results of the applied approach to the problem of image generation with Generative Adversarial Networks (GANs). The report consists of two main parts. The first one is focused on giving a theoretical background of the project and adopted assumptions most of which concern the experiments' setup. The other one concerns the practical application of the networks under consideration and presents the performance of custom models. While the full code is included in the attached files, this paper points out only the most important facts like an overview of the design, performance of different models and interpretations of the results.

2 Data

The dataset used in this project is a 10% sample of the bedroom category from the *Large-scale Image Dataset (LSUN)* dataset [1]. The original dataset contains around one million labelled images for each of 10 scene categories and 20 object categories. The sample used for this project consists of 303125 images presenting bedroom scenes and is sourced from *Kaggle* [2]. The images are RGB images (with 3 channels corresponding to Red, Green and Blue colour) of (256 $\times$ 256) resolution.
The data was split into training (80%) and validation(20%) data. Then, several transformations and augmentation techniques were applied, such as resizing to (64 x 64) resolution, cropping, zoom and normalization. Additionally, noise vectors of appropriate size were generated as an input for the image generator (see Architectures). In case of one of the models (GAN; see Architectures), we reproduced crappification logic presented in one of the `fastai` course solutions [3]

3 Architectures

GANs standing for Generative Adversarial Nets is a framework for teaching a DL model to capture the training data's distribution so we can generate new data from that same distribution. The concept is that there are two models trained at the same time: a generator and a critic. The generator will try to make new images similar to the ones from the given dataset, and the critic will try to classify authentic pictures from the ones the generator does.
Models are trained against each other in the sense that at each step (more or less), the generator is frozen and the critic is trained by:

- getting one batch of real images,
- generating one batch of fake images,
- having the critic evaluate each batch and compute a loss function from that; the important part is that it rewards positively the detection of real images and penalizes the fake ones,
- update the weights of the critic with the gradients of this loss

The generator is designed to return images, while the critic is a binary classification network that takes an image as input and outputs a scalar probability that the input image is real (as opposed to fake).

[1] LSUN Dataset `http://www.yf.io/p/lsun`
[2] LSUN bedroom scene sample `https://www.kaggle.com/datasets/jhoward/lsun_bedroom`.
[3] FastAI course notes `https://github.com/aarcosg/fastai-course-v3-notes/blob/master/refactored_by_topics/CNN_L7_gan_feature-loss.md` .

In this project we tested and compared 3 different network architectures:

- `Wassertein GAN` (`fastai` package)
- `GAN` (`fastai` package)
- `DC-GAN` (`pytorch` package)

We implemented the first of the proposed approaches by defining a `basic_generator` and a `basic_critic` passed to `gan_learner`. To define a `GAN Learner` (the 2nd approach) we combined (previously pre-trained) generator and critic learner objects. For this purpose, we also specified the switcher, which is a callback that decides when to switch between discriminator and generator. The last approach is sourced directly from `PyTorch` documentation.

4 Experiments and evaluation

The stage of the experimental part of the project aimed at investigating the influence of parameter change on the obtained results and in the end finding the optimal configuration for each model. Depending on the type of the network, and also due to hardware limitations, the length of the learning process differed. For the WGAN approach 30 epochs were run, for GAN - 10, and DC-GAN - 15. The `batch_size` parameter was set permanently to 128 based on the parametrization of the solutions we used (except for GAN, where it needed to be changed to 12 due to memory issues). We investigated the influence of two important elements of each of the adopted approaches: learning rate and the type of optimizer. The following values of those parameters were considered:

- `learning rate`: 2e-3, 2e-4, 2e-5
- `optimizer`: Adam, RMSProp

When it comes to evaluation of the prepared models, there is no objective loss function used to train the GAN generator models and no way to objectively assess the progress of the training and the relative or absolute quality of the model from loss alone. Instead, a suite of qualitative and quantitative techniques has been developed to assess the performance of a GAN model based on the quality and diversity of the generated synthetic images. In order to provide a robust assessment of the GAN models we used, we combined two approaches:

- **Quantitive evaluation**. We calculate Frechet Inception Distance (FID) between the original images from the validation data and the generated ones. It is the main tool of comparison between the adopted approaches as it is
- **Qualitative evaluation**. We calculate the distance matrix between the original images from the validation data and the generated ones. Then, for every original image, the most similar images among the generated ones are selected using the Nearest Neighbours method and reviewed.

4.1 WGAN

Table 1: FID scores for the WGAN model depending on the optimizer and the learning rate (LR).

Opt.\LR	**2e-3**	**2e-4**	**2e-5**
Adam	287.12	110.84	376.23
RMSProp	317.34	111.94	392.58

Based on the FID scores presented in the Table 1 we can conclude that the `learning_rate` parameter had a much higher impact on the results than the choice of the optimizer. Best results were obtained for the `learning_rate` equal to 2e-4, no matter the optimizer. The scores around 300 can indicate that for these parameters the model needs more time to learn or simply does not converges at all. In the Figure 1 the batch of the results of the best model (according to the FID score) is presented. The model after 30 epochs of the learning process converges to quite good results.

Figure 1: Results of WGAN with Adam optimizer and 2e-4 learning rate.

4.2 GAN

Table 2: Training, validation, generator and discriminator losses for the GAN learning process.

Epoch	Training loss	Validation loss	Generator loss	Discriminator loss
1	5996.464	0.0449	0.0449	5952.878
2	6074.047	0.0449	0.0449	5440.343
3	6042.964	0.0449	0.0449	6571.562
4	5914.358	0.0449	0.0449	6959.235
5	6021.091	0.0449	0.0449	6034.638
6	5831.079	0.0449	0.0449	6816.565
7	5815.948	0.0449	0.0449	4874.709
8	5871.582	0.0449	0.0449	5601.537
9	6134.320	0.0449	0.0449	4892.370
10	5852.951	0.0449	0.0449	5848.696

Unfortunately, we did not get satisfactory results for the GAN model. The learning process seemed not to converge at all (see Table 2) and the FID value reached around 800. We experienced similar problems while training WGAN which forced us to switch to the older version of the `fastai` package. However, in case of GAN, we were not able to create the desired model using this version. Moreover, we experienced problems with GPU memory (we reached its limit during training) which prevented us from exploring the problem further, for instance investigating and improving the pretrained generator and discriminator models.

4.3 DC-GAN

Table 3: FID scores for the DC-GAN model depending on the optimizer and the learning rate (LR).

Opt.\LR	**2e-3**	**2e-4**	**2e-5**
Adam	372.47	76.55	100.35
RMSProp	398.9	434.76	138.5

Similarly to WGAN, we can observe that the best result is obtained for the following parameters: `learning_rate` equal to 2e-4 and Adam optimizer. Figure 2 shows a batch of images generated by the model with the mentioned parametrization. Additionally, we can conclude that `learning_rate` equal to 2e-5 also seems to lead to satisfactory results but probably more epochs are needed to improve the obtained results.

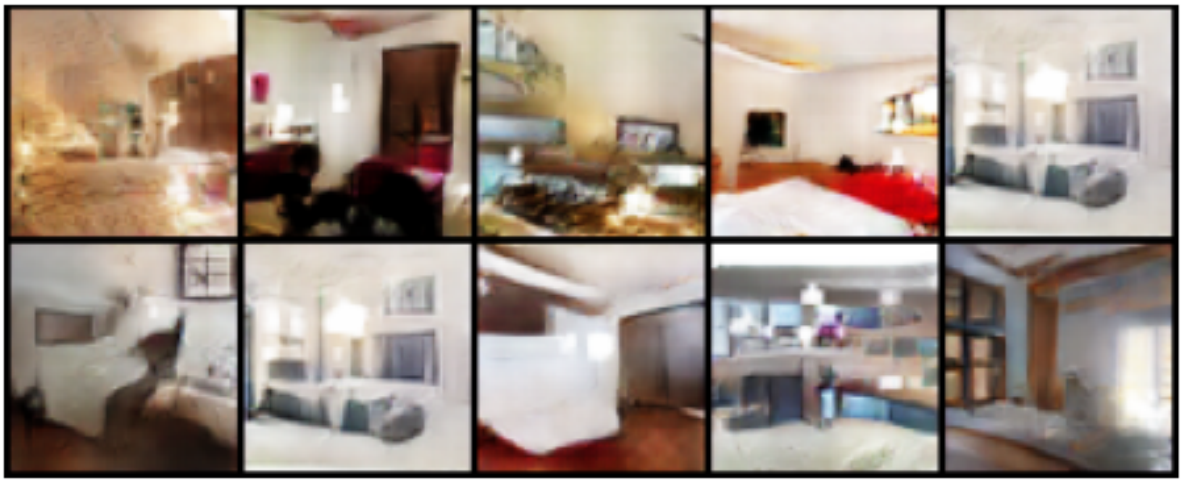

Figure 2: Results of DC-GAN with Adam optimizer and 2e-4 learning rate.

4.4 Best results

Based on the obtained results, we decided to test the most promising solutions - WGAN and DC-GAN with `learning_rate` equal to 2e-4 - one more time, this time giving them more time to learn and setting the number of epochs to 50.

Table 4: FID scores for WGAN and DC-GAN model after 50 epochs of the learning process.

WGAN	DC-GAN
90.52	38.68

In case of WGAN we can see there is an improvement when it comes to the FID score, which decreased from 110 to 90, but it is not a significant difference so it is hard to spot a visible improvement when it comes to quality of the generated pictures (see Figures 1 and 3). However, we can assume that the continuation of the learning process would lead to even smaller FID and, therefore, visible quality improvement. In case of DC-GAN, we can observe that generated images are almost identical to the original (resized) ones, which is confirmed by a significantly lower FID score. A longer learning process would probably lead to even better results, however, these are more than satisfactory and better than we expected.

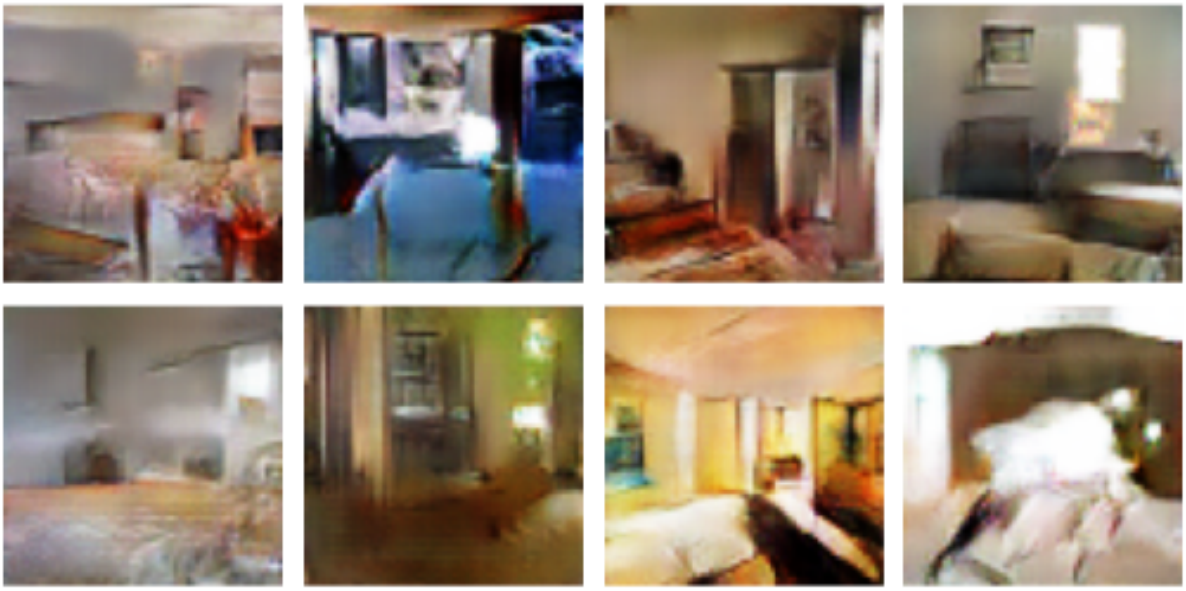

Figure 3: Results of WGAN with Adam optimizer and 2e-4 learning rate after 50 epochs.

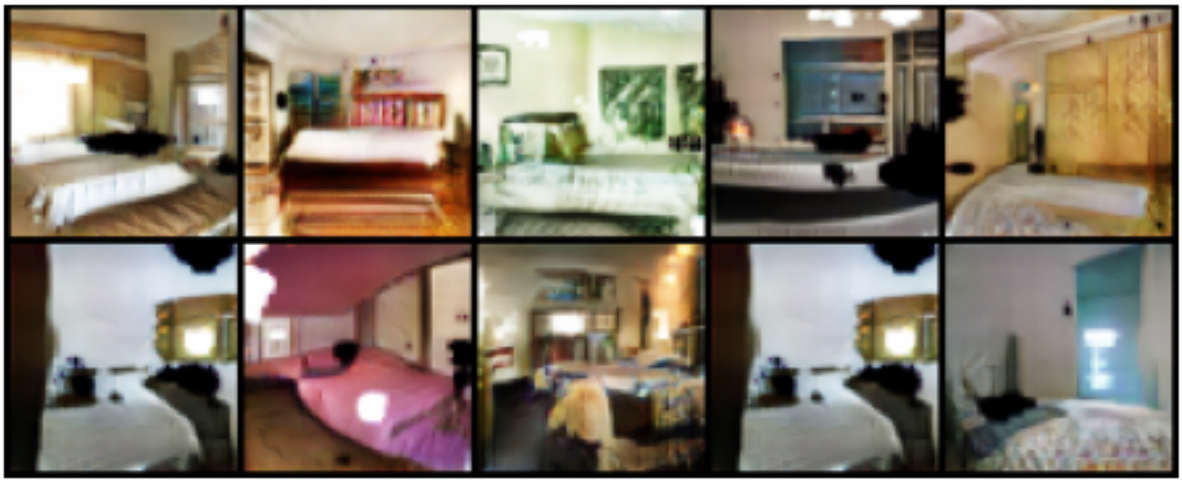

Figure 4: Results of DC-GAN with Adam optimizer and 2e-4 learning rate after 50 epochs.

4.5 Latent vectors

Additionally to the experiments described above, we selected two of the generated images and the latent vectors corresponding to them. Then, we performed linear interpolation between the two vectors, generating eight additional latent vectors, and used our best model (DC-GAN) to generate images from them. The result was a series of images that reflected the transition between the two original images. Figure 5 presents the ten generated images (eight newly generated and two generated previously). We can clearly see the step-wise progression from the first pictures on the left to the final ones on the right.

Figure 5: Ten images generated by interpolation of the latent vectors of the two selected, originally generated images.

5 Summary

To sum up, out of all the tested approaches, DC-GAN performed best and needed the smallest amount of epochs of the learning process to converge to a satisfactory result. Additionally, given the set of the values we tested, `learning_rate` equal to 2e-4 turned out to guarantee the best results no matter the model. When it comes to the optimizer, in case of the WGAN approach, the results returned by Adam and RMSProp did not differ significantly. However, in case of the DC-GAN approach, only the Adam optimizer led to sensible results.

In general, comparing obtained FID scores to the best ones found in the literature for the LSUN dataset (3.48) we can observe that our best results are approximately 10 times worse. However, there are two important factors worth mentioning. Firstly, FID implementations differ; in this project, *PyTorch* implementation was used, and not all solutions mention which implementation was used. Secondly, the best results are obtained using different architectures than the ones we used (for instance Projected GAN), additionally, joined with other solutions such as TTUR [4]. If we compare our results to the simple DC-GAN presented in this paper, we can observe the values for the number of epochs we run (15-30) are similar (from the 150-300 range). Therefore, we may conclude that our results would be surely better if we performed a longer learning process, but still, we were able to obtain a satisfactory result after prolonging the process for DC-GAN.

Among the failure cases that are common to see when training GAN models - mode collapse and convergence failure - we experienced mainly the latter. While investigating a sizeable batch of generated images for the reasonable solutions (the ones that started to converge, see Figure 6), we can observe a high diversity of the images which indicated we did not experience the mode collapse during the learning process. Mode collapse simply means many identical generated examples, regardless of the input point in the latent space. In order to simulate such behaviour, we could impair our GAN by restricting the size of the latent dimension directly, forcing the model to only generate a small subset of plausible outputs. When it comes to convergence failure, we experienced it during hyperparameters tuning, mainly for the too large learning rate value. Figure 7 present a sample of images generated by WGAN for `learning_rate` parameter equal to 2e-3. We can observe that what is generated is simply a batch of colourful pixels, which does

[4] Two time-scale update rule (TTUR) for GANs `https://www.researchgate.net/figure/Mean-maximum-and-minimum-FID-over-eight-runs-for-WGAN-GP-on-LSUN-Bedrooms-TTUR-learning_fig5_317930102`

not reflect even a part of the target bedroom image at all. Fortunately, finding an optimal value of the hyperparameter helped to overcome this issue.

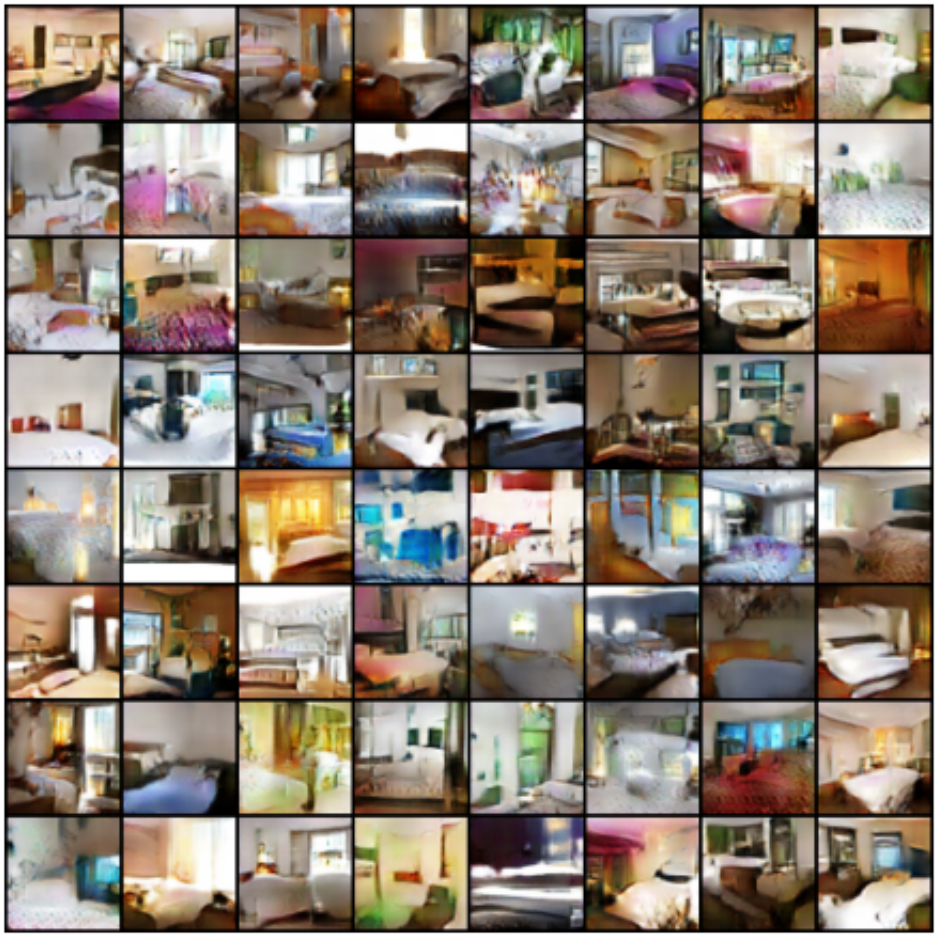

Figure 6: A batch of size 64 of images generated by best DC-GAN approach.

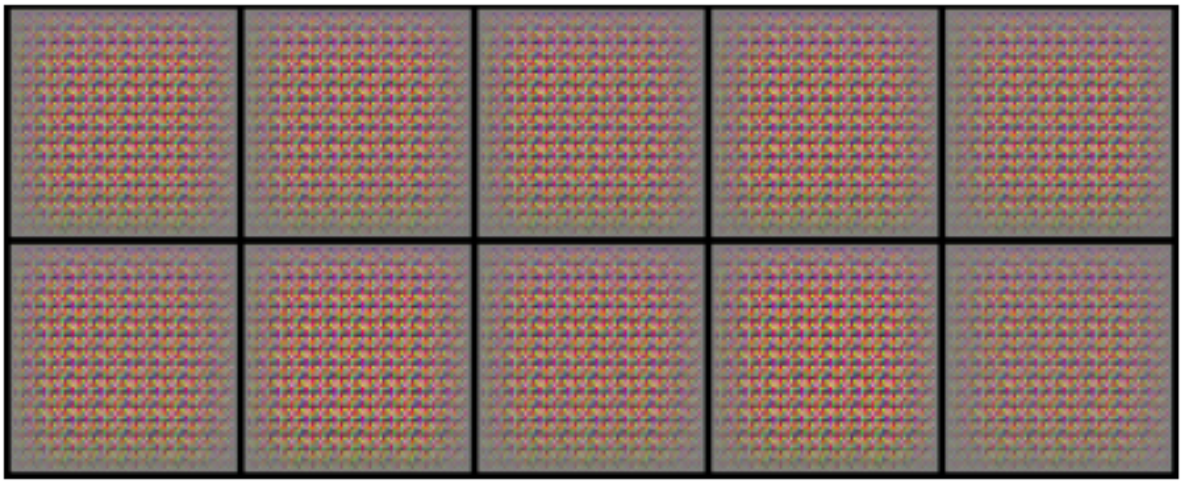

Figure 7: Results of DC-GAN experiencing convergence failure for learning rate equal to 2e-3 and 30 epochs.