Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

LSUN Bedroom Image Generation Using Diffusion Models

Adam Majczyk 313420, Szymon Matuszewski 313435

Version 1.0

08.06.2024

Contents

1 Introduction

1.1 Background

Generative models In the recent years generative models have become increasingly popular. They are frequently used on the Internet for creating memes, AI-art, AI-covers and many more. However they also pose the threat of creating disinformation. It is easy for a model to generate an image of, for example, a president being caught in an uncomfortable setting.

Models can generate based on a prompt (textual or different) or be unconditional models, created for the purpose of generated one type of image (or sound).

On Figure 1 a sample image generated with a state-of-the art model is presented.

Figure 1: "Astronaut riding a horse" - generated in Stable Diffusion XL, Public Domain

GAN

GANs [1] (generative adversarial networks) are a neural network architecture created (in 2014) for the purpose of image generation.

On Figure 2 the architecture of a GAN is presented.

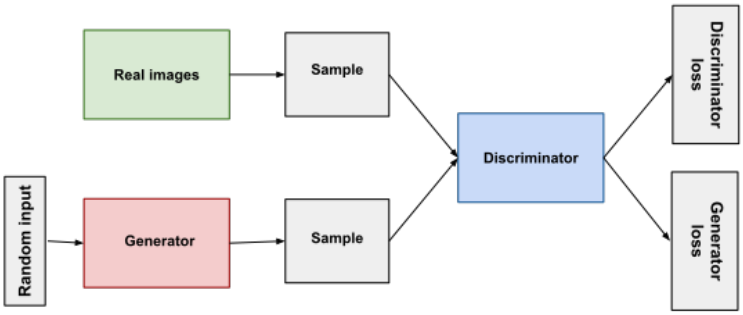


Figure 2: GAN architecture [2]

A GAN works by creating two competing networks. A generator and a discriminator. The generators task is to generate an image and the discriminators job is to say if the image is good or not.

Diffusion

As of 2021, GAN, a formerly state-of-the art approach for generating images, has been dethroned [3] by Diffusion [4] based models, which have been introduced in 2015 (a year after GANs). Diffusion however has started to gain popularity in 2020, following substantial advances made to the algorithm, such as the introduction of latent Diffusion, especially Stable Diffusion [5] introduced in 2022.

The approach is inspired by diffusion - a thermodynamics process which describes the movement of atoms.

The general idea is that the model can perform two main procedures:

- **Forward step** - the model add noise to the image (in some set number of steps) until the image becomes noise. This is the simulation of diffusion of information.
- **Backward/Reverse step** - Beginning from noise, the model reverses the forward step and reconstructs the image.

The architecture of a Diffusion model is presented in Figure 3.

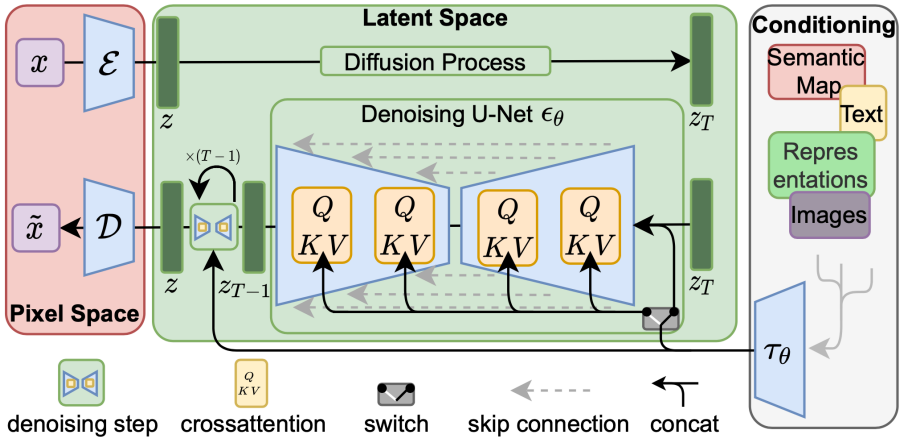


Figure 3: Diffusion model architecture [6]

1.2 Problem Definition

The problem discussed in this report is one in the class of unconditional generation. The aim is to generate images of bedrooms and test out different diffusion models to achieve this task.

1.3 Dataset

The dataset used for this project is a 20% sample of the LSUN bedroom scene dataset. [7] [8]. The 20% sample consists of over 300.000 images of varying sizes and dimensions. The images are saved as RGB files in JPG format. They are pictures of bedrooms.

1.4 Preprocessing

Because of the varying sizes of pictures (models need a square) each picture which was not a square was preprocessed. The preprocessing consisted of adding white striped to the top and bottom (or left and right) of a picture, to make it square with a side length equal to the longer side of the original picture. See example in Figure 4.

(a) Original 341x256 image

(b) Square 341x341 image

Figure 4: Preprocessing - square pictures (black border only to visualize the change)

Later datasets with varying subsets (10%, 1%, 0.25%) of the square images were generated. Then those subsets were resized to 96x96 and 128x128 sizes as the models need a uniform size of picture for training. Random horizontal flip and normalization were applied (with a random seed to enable reproducibility).

1.5 Approaches

It was decided to attempt to train two types of unconditional diffusion models. A DDPM and an LDM (often also referred to as a DDPM with an added VQ-VAE).

DDPM

DDPM are a class of diffusion models which use the classic approach described earlier. The diffusion is performed in the pixel space. To perform the training the `diffusers` library was used (namely `DDPMScheduler`, `DDPMPipeline`, `UNet2DModel` were the core elements of the architecture).

LDM

LDM are a class of diffusion models which use a modifier approach. They first encode the image into a latent representation using a VQ-VAE (into the latent space). Then they perform diffusion on the latent space. They are the base for Stable Diffusion.

1.6 Used hardware and packages

The project is largely implemented using `torch` and using `diffusers`. CUDA (NVidia RTX 4090 and 4080) was used to speed up the computation.

2 DDMP

2.1 Architectures

We have tested 3 U-Net architectures, namely ones with architectures:

- **small**
  - **layers per block**: 1
  - **block out channels**: 64, 64, 128, 128, 256
  - **block down types**: DownBlock2D, DownBlock2D, DownBlock2D, DownBlock2D, AttnDownBlock2D
  - **block up types**: AttnUpBlock2D, UpBlock2D, UpBlock2D, UpBlock2D, UpBlock2D
- **mid**
  - **layers per block**: 2
  - **block out channels**: 128, 128, 256, 256, 512, 512
  - **block down types**: DownBlock2D, DownBlock2D, DownBlock2D, DownBlock2D, AttnDownBlock2D, DownBlock2D
  - **block up types**: UpBlock2D, AttnUpBlock2D, UpBlock2D, UpBlock2D, UpBlock2D, UpBlock2D
- **big**
  - **layers per block**: 3
  - **block out channels**: 128, 256, 256, 512, 512, 768
  - **block down types**: DownBlock2D, DownBlock2D, DownBlock2D, DownBlock2D, AttnDownBlock2D, DownBlock2D
  - **block up types**: UpBlock2D, UpBlock2D, UpBlock2D, UpBlock2D, AttnUpBlock2D, UpBlock2D

NOTE: The architectures were hand picked and may not be the best.

NOTE: All models were trained using 500 denoising timesteps.

2.2 Hyperparameters

The following hyperparameters have been taken into consideration:

- **Learning rate**: 10^{-3}, 10^{-4}, 10^{-5}
- **Optimizer**: AdamW, SGD

2.3 Experiments

The experiments consisted of training each hyperparameter/architecture combination 5 times and saving the results. They were trained for 35 epochs each. This experiment was conducted on 96x96 images. The batch size was set to 16.

NOTE: Due to large computation expenses, the hyperparameter experiment was trained using 0.25% of the original dataset. Also FID scores were calculated each 5 epochs for small U-Net, 7 for mid U-Net, 10 for big U-Net, due to long computation time of FID. FID was also always calculated after the last epoch.

2.4 Results

In the following sections, the discussion of the results is presented.

2.5 Best hyperparameter set selection

In Table 1 the results of the hyperparameter experiments are presented. The table is sorted by the mean value of the FID in the last epoch in a descending fashion.

NOTE: Since there were hyperparameter combinations for which an iteration has resulted in the explosion of the gradient (namely Adam/big/0.001 and Adam/mid/0.001) they were omitted in the table.

				FID in last epoch				**Loss in last epoch**			
n	**learn. rate**	**optimizer**	**UNET**	**mean**	**std**	**min**	**max**	**mean**	**std**	**min**	**max**
1	1.0E-04	Adam	mid	208.61	3.2	203.55	212.27	0.035	0	0.035	0.036
2	1.0E-03	Adam	small	209.41	7.22	203.21	219.75	0.035	0	0.034	0.035
3	1.0E-04	Adam	big	210.24	3.9	205.92	213.94	0.036	0.001	0.036	0.037
4	1.0E-04	Adam	small	216.66	3.84	211.71	219.94	0.048	0.001	0.047	0.049
5	1.0E-05	Adam	big	239.54	3.84	234.44	244.81	0.059	0.001	0.058	0.06
6	1.0E-05	Adam	mid	263.44	3.61	260.32	268.49	0.061	0.001	0.06	0.062
7	1.0E-05	Adam	small	294.64	2.52	291.25	297.86	0.103	0.001	0.102	0.104
8	1.0E-03	SGD	mid	304.21	3.37	300.88	307.94	0.157	0.001	0.156	0.158
9	1.0E-03	SGD	big	305.39	2.38	302	307.84	0.159	0.001	0.158	0.159
10	1.0E-03	SGD	small	311.61	2.27	308.49	314.12	0.221	0.001	0.22	0.222
11	1.0E-05	SGD	big	323.16	2.21	320.14	325.43	1.168	0	1.168	1.169
12	1.0E-04	SGD	big	333.28	2.33	330.1	335.59	0.905	0	0.904	0.905
13	1.0E-04	SGD	small	335.53	2.27	332.59	338.06	0.89	0	0.89	0.89
14	1.0E-05	SGD	small	336.71	2.24	333.66	338.9	1.061	0	1.061	1.061
15	1.0E-05	SGD	mid	353.03	2.91	350.28	356.74	1.059	0	1.059	1.06
16	1.0E-04	SGD	mid	354.61	2.83	351.91	358.08	0.823	0	0.822	0.823

Table 1: The results of the hyperparameter experiments

One may arrive at such conclusions regarding the choice of hyperparameters:

- It is clear that out of the optimizers **Adam** is the clear winner. It has consistently outperformed **SGD**.
- Larger sizes of UNet do not guarantee better performance.
- The learning rate of 0.0001 (10^{-4}) appears to be the best - it appeared 3 times in the top 3 configurations.

Hence the subsequent ***Big Model*** was trained on this combination of hyperparameters and architecture: **Adam/0.0001 (10^{-4})/mid**.

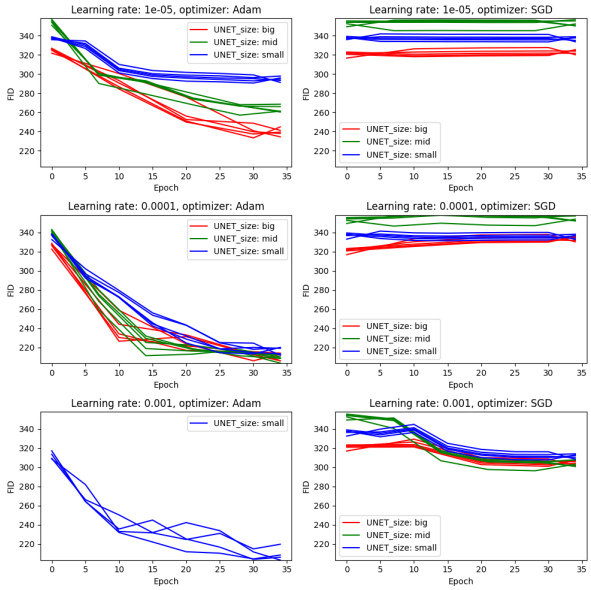
NOTE: Please note that the values of FID score may not be comparable with values lower than 10 available in literature [9], due to the low percentage of images used in this project. This may also be caused by the differences in preprocessing of images and image sizes (in this project 96x96, commonly 256x256 in literature).

NOTE: The lowest FID that was achieved during the development was 130 on a 128x128 model with a small UNet and 0.001 (10^{-3}) learning rate using Adam optimizer. This result was achieved on 1% of the data.

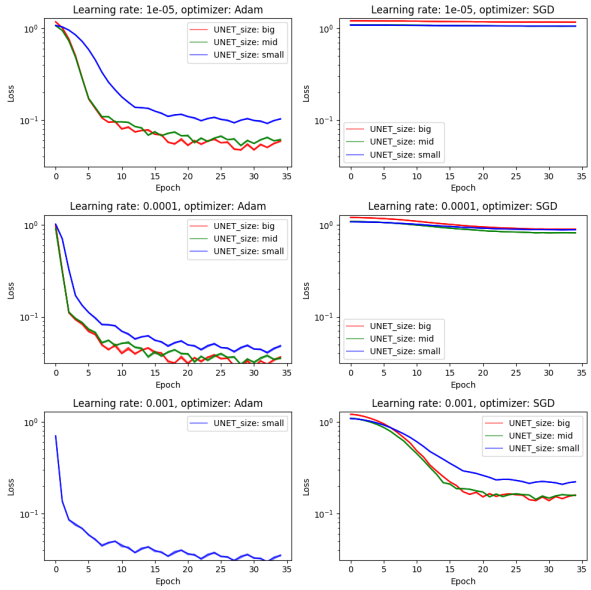
NOTE: FID was not calculated for larger samples of data, as the computation time would have exceeded a week. The value of FID would likely be lower, had it been calculated on 10-50% of the data. This would however require more computation power than an RTX 4090 can provide or a significantly longer computation timeframe.

2.6 Hyperparameters that overcome training/mode collapse

In Figures 5a and 5b the values of FID scores and loss are presented over the course of the training process.



(a) FID Scores in training, split over Learning Rate and Optimizer



(b) Loss in training, split over Learning Rate and Optimizer

Figure 5: Training performance metrics split over Learning Rate and Optimizer

By analysing the graphs one may come to the conclusions that:

- It is clear that **SGD** can get stuck and not generate better looking images. **Adam** performs far better.
- Both **large** and **mid** sized UNets performed well and did not refuse to get better in the span of 30 iterations.
- The **big** and **mid** sized UNets may experience training collapse (explosion of gradient).

NOTE: some combinations are missing from the graphs (mid/0.001 (10^{-3})/Adam and big/0.001 (10^{-3})/Adam) as at some point the loss of the objective function has reached NaN value - the training has collapsed.

NOTE: mode collapse is not described, as it has not been experienced.

2.7 Big Model

The model has been trained for 30 epochs with configuration **Adam/0.0001** (10^{-4})**/mid** on 128x128 images (10% of the dataset). In Figure 6 a sample of 9 images generated using that model are presented.

Figure 6: 9 images generated using ***Big Model***, FID $\approx$ 130

In the opinion of the writers they are similar to the bedroom images found in the dataset. The generation result is deemed visually satisfactory.

2.8 Qualitative checks

By looking at the example images in Figure 6 one can determine that:

- The images are blurrier than the original images.
- Some room configurations miss a bed (bottom middle).
- Some walls are curvy (middle) or start connect with the ceiling in the middle of the wall (right middle).

In summary the images looked at from afar do resemble a bedroom. However looking at them closer, one can determine that they are not actual bedrooms.

3 LDM

In this section the training of LDM and the generation results are presented.

3.1 Failure

An attempt was made to train an LDM model (a DDPM with an added VQ-VAE). This has however resulted in a failure to generate visually good looking results.

3.2 Generated Images

In Figure 6 a sample of 9 images generated using the LDM is presented.

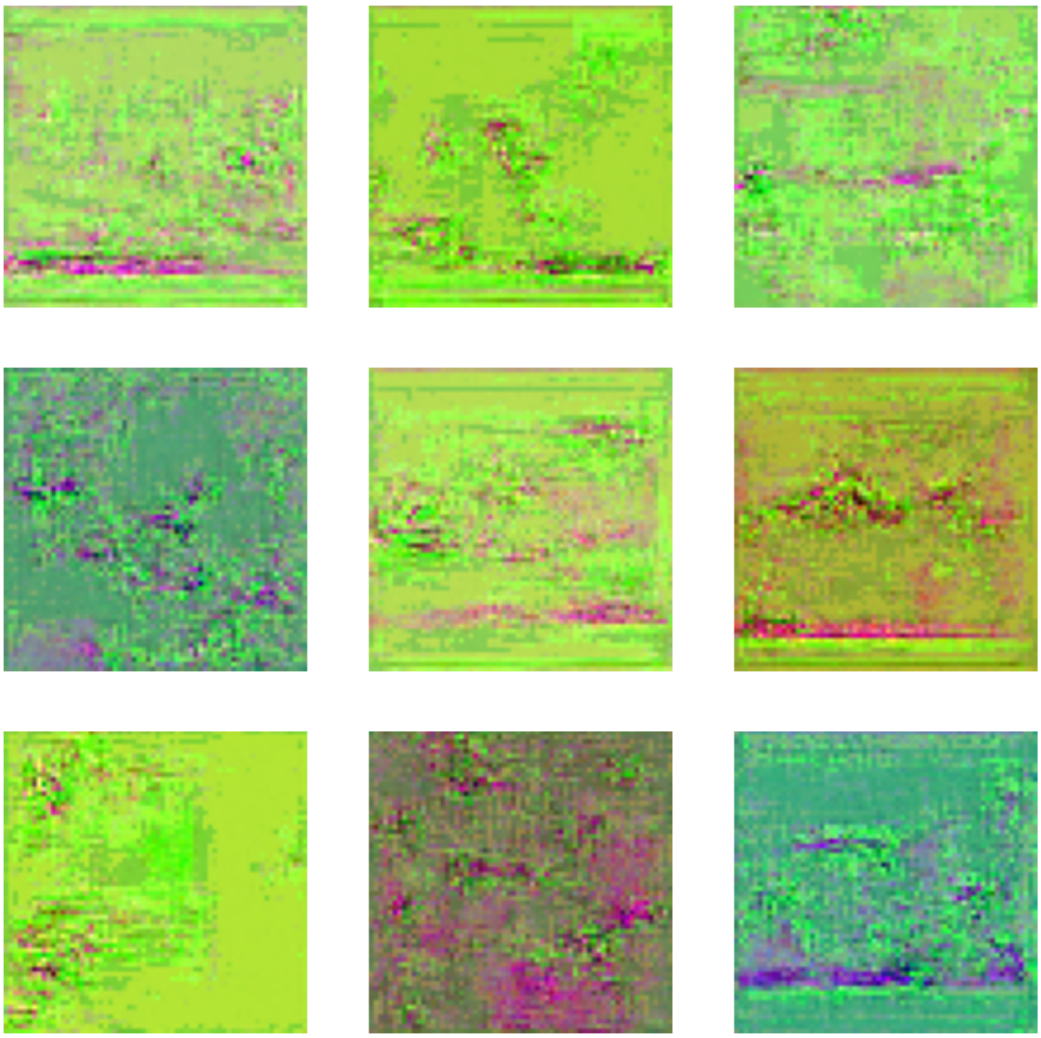


Figure 7: 9 images generated using ***Failed LDM*** after 50 epochs

Because of the failure LDMs have not been considered throughout the rest of the project.

4 Interpolation of Latents for selected model

4.1 Approach

Proper implementation of extraction of latent vectors from a DDPM, `UNet2DModel`-based model, where by a latent the intermediate encoding in a mid block is understood was attempted. However due to inconsistent sizes of tensors, the latent representation being fed into the Up-block resulted in dimensionality errors.

This issue stems from the fact, that the `diffusers` library does not support extraction of latents (in the middle of the net) and feeding them beginning from the middle out of the box. An attempt at using hooks and custom UNets was made. It has also resulted in dimensionality errors.

Hence an approximate experiment has been conducted, where by the latent, the image passed through the entire model was understood.

NOTE: This issue stems primarily from the theoretical fact that DDPM models don't perform diffusion in the latent space, rather the pixel space so there is no easily available latent of a selected picture.

4.2 Result

The results of the interpolation are presented in Figure 8

Figure 8: Results of interpolation

It can be observed that the interpolated images start with a scene with a bed in the middle then gradually shift to a scene with a bar going through the middle (which resembles picture 2, that had white bars at top and bottom) and a window. The appearing white top and bottom bars can also be seen. This result makes sense, as the interpolated images should resemble the in between of the two original images.

NOTE: The model used for the interpolation was trained on 128x128 images using 10% of the data and batch size of 16. The UNet used was of size **mid**, optimizer **Adam** and learning rate of **0.0001** (10^{-4}).

5 Conclusions

By analysing the report one may come to the following conclusions:

- Training of Diffusion models requires immense computing power.
- Moderate (**mid**) sizes of UNet are a good trade-off between quality and training time.
- Adam is advised as an optimizer.
- LDMs are more complicated than DDPMs.
- You can obtain good results with small (10% $\approx$ 30.000 images) number of images.

6 Instruction

All source code is here: `https://github.com/amajczyk/2024L_DeepLearning_P3/tree/main`

6.1 Prerequisites

Install needed dependencies:

```
pip install -r requirements.txt
```

Now, you are able to regenerate the training pipeline.

6.2 Documentation

Files and folders in main directory:

- `README.md` - this file with a description of the repository
- `models/` - folder with saved models, their metadata and model definitions; subfolders contain `metadata.json` files with the metadata of the models and their training logs
- `plots/` - folder with plots generated during the project
- `PDFs/` - folder with PDFs with the project report
- `ddpm64.ipynb` - a Jupyter notebook with the implementation of the DDPM model for 64x64 images
- `ddpm128.ipynb` - a Jupyter notebook with the implementation of the DDPM model for 128x128 images
- `ddpm_hyperparameters_96.ipynb` - a Jupyter notebook with the implementation of the DDPM model for 96x96 images
- `ddpm_hypers_analysis.ipynb` - a Jupyter notebook with the analysis of the hyperparameters of the DDPM model's results
- `ddpm_vqvae_hyperparameters.ipynb` - a Jupyter notebook with the implementation of the LDM model with
- `interpolation.ipynb` - a Jupyter notebook with the implementation of the interpolation between images
- `ldm_config.yaml` - a configuration file for the LDM model
- `preprocess_images.ipynb` - a Jupyter notebook with the preprocessing of the images
- `redo_folder_structure.ipynb` - a Jupyter notebook with the code for the reorganization of the folder structure
- `requirements.txt` - a file with the required Python packages

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, June 2014.

[2] Google. Overview of GAN Structure | Machine Learning | Google for Developers.

[3] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis, June 2021. arXiv:2105.05233 [cs, stat].

[4] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, November 2015. arXiv:1503.03585 [cond-mat, q-bio, stat].

[5] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, April 2022. arXiv:2112.10752 [cs].

[6] Lilian Weng. What are Diffusion Models?, July 2021. Section: posts.

[7] LSUN bedroom scene 20% sample. LINK.

[8] LSUN Bedroom dataset. LINK.

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, December 2020. arXiv:2006.11239 [cs, stat].

List of Tables

List of Figures