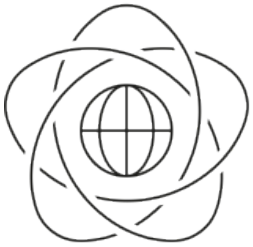

Wydział Matematyki
i Nauk Informacyjnych

Politechnika Warszawska

Deep Learning, Deep Learning Methods

Project 3: Genrative models

Mikołaj Rzepiński, Krzysztof Wolny
student record book number 276890, 305765

supervisor
Zinelabidine Leghelimi

WARSAW June 11, 2025

Contents

1. Introduction

In this project, we create images using a Generative Adversarial Network (GAN) model. First, we generate cat images using a dataset from kaggle.Then, we analyze the latent space of our GAN model. LadaGAN[1] works well while using much less computing power, which is important for our limited resources. It performs similarly to top models like diffusion models but is much more efficient. This makes it a good choice for our project.

- LadaGAN:
  - Research paper: `https://arxiv.org/abs/2401.09596`
  - Github repositories: `https://github.com/milmor/LadaGAN-pytorch`, `https://github.com/milmor/LadaGAN`
- Datasets
  - Cat Dataset: `https://www.kaggle.com/datasets/borhanitrash/cat-dataset`
- Exploration the GAN Latent
  - How to Explore the GAN Latent Space When Generating Faces by Jason Brownleehttps://machinelearningmastery.com/how-to-interpolate-and-perform-vector-

2. Methodology and Experimentation

In this project, we use a Generative Adversarial Network (GAN) model — LadaGAN. The model is trained on a cat image dataset from Kaggle.

We use 1,000 images from the dataset for training and another 1,000 for evaluation. The evaluation is based on the Fréchet Inception Distance (FID) score. All images are resized to 64×64 pixels.

In total, we trained the model for 12,000 iterations. Every 1,000 iterations, we calculated the FID score and generated 64 cat images. We also saved the model with the best FID score for further use in latent space exploration.

After training, we used the model with the best FID score to analyze the latent space. We generated two noise vectors and performed interpolation by creating 8 intermediate images between these two latent points. This allowed us to visualize the transition between the two generated images.

3. Results and Discussion

3.1. Training LadaGAN and its results

Figure 4.1 illustrates the progression of generated images throughout the training. At iteration 1,000 (Figure 4.1a), the outputs are just a noise with no recognizable structure.

By iteration 2,000 (Figure 4.1b), the model starts forming basic cat-like features, although the images remain blurry and inconsistent.

As training progresses, visual quality significantly improves. By iteration 6,000 (Figure 4.1d), most generated samples resemble real cats, with clearly defined eyes, fur patterns, and facial structures. This improvement continues through iterations 10,000 and 12,000 (Figures 4.1e–f), where the majority of generated images show clear, high-quality cat faces with diverse appearances and minimal visual artifacts.

Table 4.1 presents the FID scores calculated across iterations. The initial FID score at iteration 1,000 was very high (348.75), reflecting poor generation quality. However, the score quickly dropped, reaching 79.63 at iteration 3,000 and continuing to improve, stabilizing around 40 after iteration 8,000. The lowest FID score (40.30) was recorded at iteration 10,000.

Overall, both the visual results and FID scores demonstrate that LadaGAN effectively learns to generate high-quality cat images. The visual progression and metrics show consistent improvement.

(a) Iteration 1000 **(b)** Iteration 2000 **(c)** Iteration 3000

(d) Iteration 6000 **(e)** Iteration 10000 **(f)** Iteration 12000

Figure 3.1. Progression of GAN-generated images across training iterations

Iteration	FID Score
1000	348.75
2000	168.91
3000	79.63
4000	57.72
5000	48.41
6000	44.01
7000	43.19
8000	41.35
9000	40.98
10000	40.30
11000	47.30
12000	43.63

Table 3.1. FID scores over training iterations for LadaGAN

3.2. Latent noise matrixes

To see how the model understands shapes and features, we created two random noise vectors and made a smooth transition between them. The result is shown in Figure 4.2.

The first and last images are from the original noise vectors. The images in the middle were made by mixing these two vectors step by step. As we can see, the change happens slowly and naturally — for example, the color of the fur, shape of the face, and position of the eyes change gradually.

This shows that the model learned a good structure of the image space. Small changes in the input noise create small and realistic changes in the generated images.

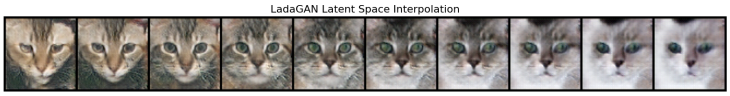

Figure 3.2. Generated images together with their latent noise matrix

4. Conclusions and Future Work

We successfully managed to generate cat images using the LadaGAN model. Most of the generated images clearly resemble cats. However, some outputs still contain deformations or only loosely resemble cats, especially in terms of contour and shape.

Possible further work:

- Train on a larger dataset - in this project, we used only 1,000 images for training.

- Explore model hyperparameters - for example, test different learning rates, as well as different batch sizes or optimizer settings.
- Train for more iterations - while the model improved up to 12,000 iterations, further training could yield better FID scores and image quality.
- Evaluate on different datasets - testing the model on other image datasets (e.g., dogs, cats and dogs, objects) could help test it better.
- Use data augmentation - applying transformations (like rotation, flipping, or cropping) during training may improve performance, especially with small datasets.
- Improve model architecture - exploring newer GAN variants or adding normalization layers might help generate higher quality images.

5. Application Instruction

5.1. Run `DeepLearning_Project_3_Rzepinski__Wolny.ipynb`

1. Open the `.ipynb` file from the ZIP archive in Google Colab First, unzip the project folder on your computer. Then open Google Colab, click `File` → `Upload notebook`, and select the `DeepLearning_Project_3_Rzepinski__Wolny.ipynb` file from the extracted folder.
2. Use Google Colab
   Make sure you select a GPU runtime.
   (Click `Runtime` → `Change runtime type` → choose `GPU`).
3. Download JSON with Kaggle API token
   Go to kaggle.com/account, scroll to `API`, and click `Create New API Token`.
   This will download a file called `kaggle.json` — you will need it to access datasets from Kaggle.
4. Run the notebook
   Run each cell step by step (from top to bottom).
   When the first cell asks you to upload a file, upload the `kaggle.json` file with your Kaggle API token.
5. Before running `!python LadaGAN-pytorch/train.py -data`$_{dir}$ = ./$data_{path}$/−−$fid_{real}$$_{dir}$ = ./$eval_{path}$/$changein$`LadaGAN-pytorch/config.pyn_fid_real: 1000,n_fid_gen: 1000,n_iter:`

5.2. Run `latent_matrixec.ipynb`

1. `Clone 'LadaGAN-pytorch' with !git clone https://github.com/milmor/LadaGAN-pytorch.`
2. Install necessary libraries.
3. Put `latent_matrixec.ipynb` inside `LadaGANpytorch` folder.
4. Unzip `test_model.zip` and change model_dir to be matching `test_model` folder.
5. Run notebook.

References

[1] E. Morales-Juarez and G. Fuentes-Pineda, *Efficient generative adversarial networks using linear additive-attention transformers*, 2024. arXiv: 2401.09596 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2401.09596.