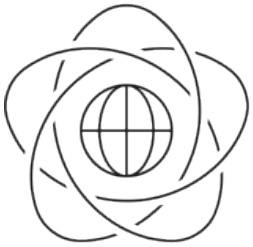

Wydział Matematyki
i Nauk Informacyjnych

Politechnika Warszawska

Deep Learning, Deep Learning Methods

Project 3: Generative models

Kinga Frańczak, Mateusz Wiktorzak

WARSAW June 11, 2025

Contents

1. Introduction

2. Theoretical Background and Literature Review

Generative models aim to learn the underlying distribution of a dataset in order to generate new, synthetic instances that resemble the training data. Unlike discriminative models that learn the conditional probability $P(y|x)$, generative models attempt to learn the joint probability distribution $P(x, y)$ or the marginal distribution $P(x)$. Such models are instrumental in various tasks including image synthesis, data imputation, and semi-supervised learning.

2.1. Generative Adversarial Networks (GANs)

Introduced by Goodfellow et al. in 2014, Generative Adversarial Networks (GANs) [1] represent a powerful class of generative models based on a game-theoretic framework. A GAN consists of two neural networks: a generator G and a discriminator D, which are trained simultaneously in a minimax game. The generator G maps samples $z \sim p_z(z)$ from a simple prior distribution (e.g., Gaussian or uniform) to the data space, aiming to produce data that is indistinguishable from real data. The discriminator D receives both real data $x \sim p_{\text{data}}(x)$ and fake data $G(z)$, and attempts to correctly classify them as real or generated.

The training objective of a GAN is expressed as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

At balance, the generator produces data that is indistinguishable from the real data distribution, and the discriminator is maximally uncertain, outputting $D(x) = 0.5$ for all inputs.

The original GAN paper by Goodfellow et al. (2014) [1] laid the foundation for adversarial training. Despite its innovative architecture, early GANs suffered from training instability and mode collapse, where the generator produces limited diversity in samples. To address these issues, subsequent works proposed architectural and algorithmic improvements.

Radford et al. (2015) [2] introduced Deep Convolutional GANs (DCGANs), which leveraged convolutional layers to generate higher quality images. Arjovsky et al. (2017) [3] proposed the Wasserstein GAN (WGAN), replacing the Jensen-Shannon divergence in the loss function with the Earth Mover's (Wasserstein) distance, improving training stability.

Despite their success, GANs face challenges such as mode collapse, convergence issues, and difficulty in evaluating generative performance. Current research explores so-

lutions such as spectral normalization, two-time-scale update rules (TTUR), and improved evaluation metrics like FID (Fréchet Inception Distance) [4].

2.2. Variational Autoencoders (VAE's)

Introduced in 2013 in the article [5], Variational Autoencoders are a class of deep learning models created to map the input variable to a smaller latent space with an encoder, and recreate it from reduced data by a decoder.

As opposed to typical autoencoders, which are trained to map latent space elements to discrete space, the latent space in variational autoencoders is probabilistic, described by distribution parameters.

The loss function used in backpropagation consists of two parts: reconstruction loss and Kullback-Leibler Divergence. Reconstruction loss describes how much input differs from output, and Kullback-Leibler Divergence describes how much the latent space distribution differs from the probabilistic distribution, in the case of the article, the Gaussian distribution.

The Variational Autoencoders, due to a continuous probabilistic latent space, cause the smooth transition between two points. The output of the variational autoencoders is often blurry, which is characteristic for the network type, and is a drowback in creating realistic and sharp images.

2.3. Evaluation techniques

Fréchet Inception Distance is a widely used metric for quantitatively evaluating the performance of generative models [4]. It measures the similarity between the distributions of real and generated images in the feature space of a pretrained Inception network. Specifically, FID computes the Fréchet distance between two multivariate Gaussians estimated from the activations of the real and generated images. Let μ_r, Σ_r and μ_g, Σ_g denote the means and covariances of the real and generated image features, respectively. The FID is defined as:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right)$$

Lower FID values indicate a closer match between the two distributions, reflecting higher quality and diversity of generated samples. Unlike earlier metrics such as Inception Score (IS), FID is sensitive to both the visual fidelity and mode diversity of generated data, making it more reliable for assessing distribution similarity in generative tasks.

Mode collapse is a common failure mode in the training of generative models, where the generator learns to produce a limited variety of outputs, ignoring the diversity present in the real data distribution. As a result, it generates similar or identical samples regardless

of the input noise vector. With GANs, this phenomenon occurs when the generator finds a subset of outputs that consistently fool the discriminator, leading to reduced variability and poor generalization. Addressing mode collapse remains a key challenge in stabilizing GAN training and improving sample diversity [6].

Latent space interpolation is a qualitative technique used to assess the smoothness and continuity of a model's learned latent space. It involves generating two samples by passing two different latent vectors through the generator, and then linearly interpolating between these vectors to produce intermediate latent codes. These intermediate vectors are also passed through the generator, producing a sequence of images that transition from one generated sample to another. If the generator has learned a meaningful latent space, the resulting images should change gradually and coherently, preserving semantic consistency. This method visually demonstrates the generator's ability to model a continuous data distribution and capture underlying structure in the input space.

3. Methodology and Experimentation

3.1. Dataset

Dataset consists of 29,843 RGB images (.png) of cats in 64x64 resolution. It was collected from Kaggle.

3.2. GAN implementation

In our experiments, we adopt a Deep Convolutional GAN (DCGAN) architecture consisting of a generator and a discriminator trained adversarially. The generator transforms a random noise vector $z \in \mathbb{R}^d$ into a synthetic image of size $64 \times 64 \times 3$ through a cascade of transposed convolutional layers. Each layer increases the spatial resolution by a factor of two, interleaved with batch normalization and ReLU activations, and culminates in a Tanh activation to map pixel values to $[-1, 1]$. Conversely, the discriminator receives an image and processes it through a mirror stack of strided convolutional layers, batch normalization, and LeakyReLU activations, producing a single scalar via a sigmoid function that represents the probability of the image being real. Pseudocode:

Require: latent dimension d, feature map sizes F_G, F_D, learning rate η, epochs E, batch size B

Initialize generator G and discriminator D; set optimizers Adam(G) and Adam(D) with $\beta = (0.5, 0.999)$; use binary cross-entropy loss $\mathscr{L}_{\text{BCE}}$.
for $e = 1$ to E **do**
  for each minibatch of real images x of size B **do** ▷ Discriminator update
    Sample noise $z \sim \mathcal{N}(0, I_d)$, compute $\tilde{x} = G(z)$
    $\ell_D \leftarrow \mathscr{L}_{\text{BCE}}(D(x), 1) + \mathscr{L}_{\text{BCE}}(D(\tilde{x}), 0)$

Update D by descending $\nabla_D \ell_D$ ▷ Generator update
Sample new noise $z \sim \mathcal{N}(0, I_d)$, compute $\tilde{x} = G(z)$
$\ell_G \leftarrow \mathscr{L}_{\text{BCE}}(D(\tilde{x}), 1)$
Update G by descending $\nabla_G \ell_G$
end for
end for
Save final weights of G and D

Training proceeds by alternating between discriminator and generator updates. At each iteration, the discriminator learns to assign high scores to real images and low scores to generator outputs, while the generator strives to produce samples that maximize the discriminator's prediction of "real". Model parameters are optimized using Adam with a learning rate $\eta = 2 \times 10^{-4}$, and minibatches of size $B = 128$. Periodic sampling of the generator's outputs permits both qualitative inspection and quantitative evaluation, ensuring that the adversarial training remains stable and produces progressively more realistic images.

3.3. Variational autoencoder implementation

The implementation used in our experiments is based on the notebook [7]. The model consists of encoder and decoder, which are mirror reflections of one another. The latent space between the encoder and decoder is described by two parameters of Gaussian distribution: μ and σ.

The **Encoder** consisted of following layers:

- Input: (3, 64, 64)
- Hidden Layer: 512
- Hidden Layer: 512
- Output: 16

and the **Decoder** decoder of:

- Input: 16
- Hidden Layer: 512
- Hidden Layer: 512
- Output: (3, 64, 64)

The model used Adam optimiser. After each layer, the LeakyReLU (0.2) function was used.

4. Results and Discussion

4.1. GAN evaluation

In the test, we've set the learning rate to *2e-4* and the batch size to 128. The model was trained for 50 epochs, and the losses are presented in the table.

Epoch	D Loss	G Loss
1	0.3554	1.7254
2	0.8433	1.4582
3	0.9313	2.5497
4	0.7322	6.8336
5	1.3214	9.0833
6	0.1538	5.3249
7	0.5576	6.0926
8	0.2614	3.5190
9	0.3989	2.9087
10	0.0293	4.8368
11	0.3420	3.3074
12	0.4660	8.2312
... (25 epochs omitted) ...		
38	0.0908	4.8939
39	0.1649	6.7847
40	0.2461	3.7128
41	0.1874	7.1254
42	0.5161	0.0384
43	0.5714	0.9851
44	0.1364	5.1552
45	0.0172	6.6626
46	0.0862	8.8995
47	0.1399	5.3797
48	0.1071	9.2634
49	0.9051	0.2932
50	0.1063	8.1885

Table 4.1. GAN training log with discriminator and generator losses over 50 epochs (middle 25 epochs omitted).

Initially, the generator and discriminator losses fluctuate significantly, reflecting instability common in early GAN training. From around epoch 10 onward, the generator loss generally increases while the discriminator loss decreases or remains low, suggesting that the generator improves in producing realistic outputs. However, occasional spikes and collapses (e.g., epoch 42, 49) indicate episodes of mode collapse or discriminator overpowering. Overall, the training shows signs of convergence with generator performance improving, but instability remains throughout.

We've also checked how the generated images look. Below are shown generated random 16 images every 10 epochs:

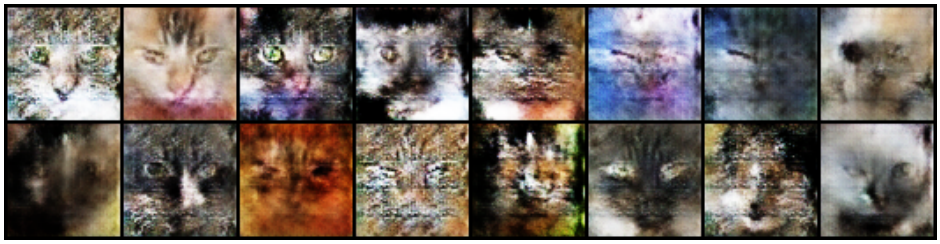

Figure 4.1. Images generated after 10 epochs

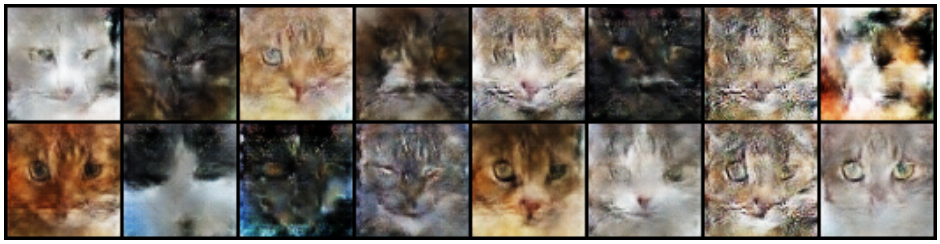

Figure 4.2. Images generated after 20 epochs

Figure 4.3. Images generated after 30 epochs

Figure 4.4. Images generated after 40 epochs

From qualitative evaluation perspective, we can see a rapid increase in the quality of the images, especially between the first 10, 20 and 30 epochs, then the differences aren't huge. Also, for every 16 randomly generated images, there are a few that don't resemble cats.

We also calculated a bigger set of final images which look very decent:

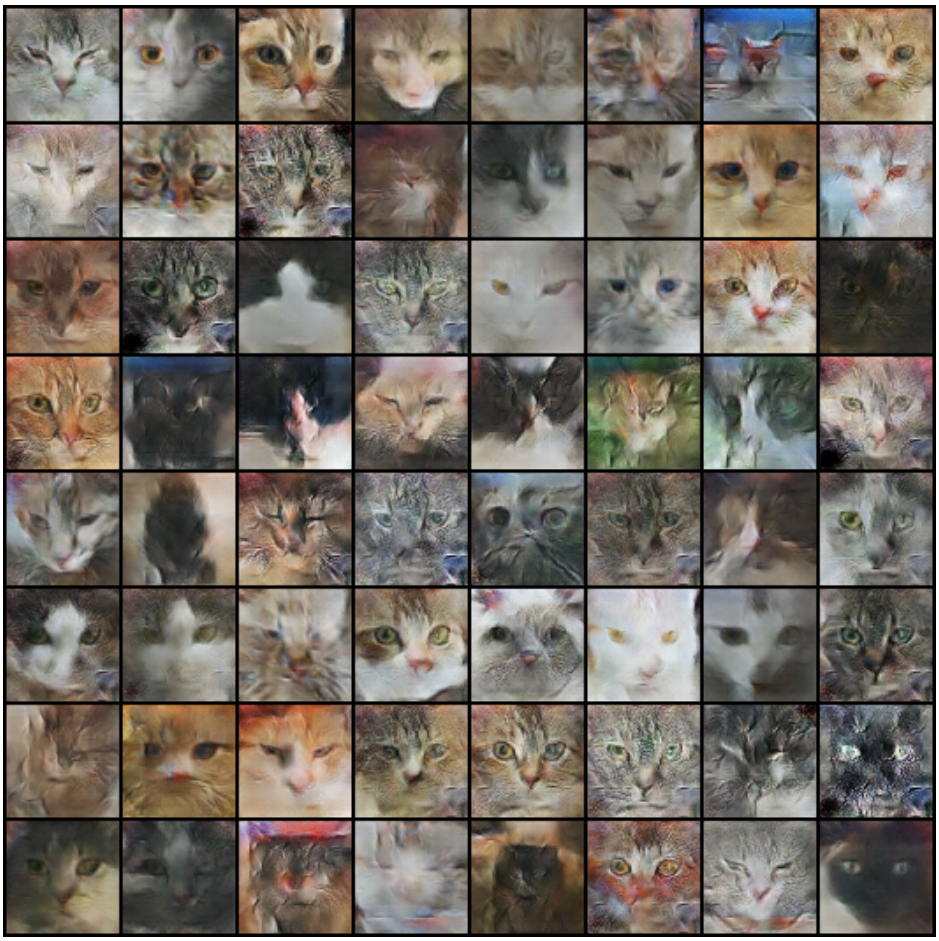

Figure 4.5. Images generated after 40 epochs

To quantitatively evaluate the quality of the generated images, the *Fréchet Inception Distance (FID)* was computed between 1000 generated samples and 1000 real images randomly selected from the dataset. The FID measures the distance between feature distributions of real and generated images extracted using a pretrained Inception network. Formally, it is the Fréchet distance between two multivariate Gaussians fitted to these features. A lower FID value indicates that the generated images are more similar to the real ones in terms of both quality and diversity. In this experiment, a FID score of **132.665** suggests that there is a noticeable gap between the generated and real image distributions, indicating room for improvement in the generator's performance.

To detect potential *mode collapse* in the generator, we analysed pairwise cosine similarity between 1000 generated samples. If many image pairs are very similar in pixel space, it may indicate that the generator is producing limited variations regardless of different input noise vectors. Each image was flattened, and cosine similarity was computed between all pairs. A similarity above 0.9 was considered highly similar. Out of 999,000 possible pairs, **2412** were found to be highly similar. This relatively low proportion (~0.24%) suggests that

the generator does not exhibit strong mode collapse and produces a reasonably diverse set of outputs.

To evaluate the smoothness and continuity of the generator's latent space, we performed a linear interpolation between two randomly sampled latent vectors z_1 and z_2. A total of 10 intermediate points were generated between these endpoints. As shown in Figure 4.6, the transition between images is gradual and coherent, indicating that the generator has learned a meaningful and continuous representation of the data manifold.

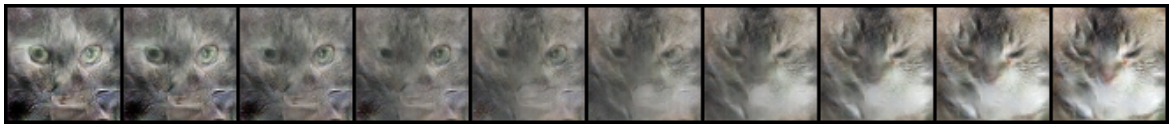

Figure 4.6. Latent space interpolation between two noise vectors.

4.2. Variational autoencoder evaluation

The variational autoencoder was trained for 50 epochs, with the whole dataset. The batch size was set to 128, and the learning rate was set to 0.0002. The change of loss function value for the first and last five epochs is presented in table 4.2.

Epoch	Average Loss
1	7686.8
2	7326.6
3	7226.8
4	7189.8
5	7157.8
... (40 epochs omitted) ...	
46	7039.9
47	7038.7
48	7037.9
49	7037.2
50	7036.5

Table 4.2. Loss changes during VAE training

The loss function was declining with each iteration, in bigger increments at the beginning of the learning process, and then stabilising at the end.

The model was assessed qualitatively by generating images from a random sample of the latent space. The images are presented in figure 4.7. On the randomly generated images, the main features of the cat head are visible, such as eyes, nose and mouth and the shape of the head. The generated images are blurry, most noticeably at the edges of the picture, which is common with the variational autoencoders.

The second qualitative assessment of the model was done with linear interpolation between two randomly selected images, presented in figure 4.8. Looking at the recreation

Figure 4.7. Images generated from a random sample.

of the original images (in top left and bottom right) shows that the images were recreated by a decoder, which tends to be blurry. The transition between the original images is really smooth, and each intermediate image resembles an anatomically correct cat.

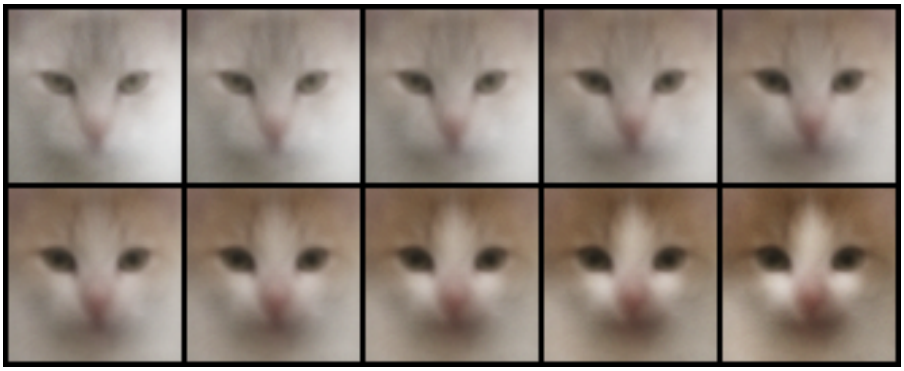

Figure 4.8. Linear Interpolation between two randomly sampled images

The variational autoencoder generates images without distinctive details, but compared to the GAN model, it doesn't produced images with weird combinations of features, that are not looking anatomically correct. The sharpness of images may be improved by increasing the size of the latent space and training the model for more epochs.

5. Conclusions and Future Work

In terms of DCGAN, we could observe that the model is performing decently. It isn't as complicated as some available models; thus, it trains in a relatively short amount of time, but also the results aren't on the level of top models. Despite that, it is a solid solution. It achieves an alright FID score, mode collapse didn't appear, and linear interpolation between two randomly sampled latent vectors was smooth.

6. Application Instruction

All files are available at github (excluding large collections of images which are produced by code). They are formed in the following way.

- cats_data/ - directory with cat images
- output/ - directory with GANs and VAE results
  - generated_for_fid/ - 1000 generated images for FID calculation
    - discriminator.pth - discriminator file
    - generator.pth - generator file
    - fresh_samples - generated images from final generator
    - interpolation.png - image showing interpolation for GAN
    - sample_5.png - generated images after 5 epochs of training (same for 10, 15, ..., 50)
    - interpolation_vae.png - image showing interpolation for VAE
    - generated_sample_vae.png - images generated from random sample by VAE
- real_subset_for_fid/ - 1000 real images for FID calculation
- GAN_50_epochs.ipynb - file with the code calculating DCGAN
- VAE.ipynb - file with code training and evaluation VAE model

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1406.2661.

[2] A. Radford, L. Metz, and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, 2016. arXiv: 1511.06434 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1511.06434.

[3] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein gan*, 2017. arXiv: 1701.07875 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1701.07875.

[4] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, *Gans trained by a two time-scale update rule converge to a local nash equilibrium*, 2018. arXiv: 1706.08500 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1706.08500.

[5] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: 1312.6114 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1312.6114.

[6] R. Durall, A. Chatzimichailidis, P. Labus, and J. Keuper, *Combating mode collapse in gan training: An empirical analysis using hessian eigenvalues*, 2020. arXiv: 2012.09673 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2012.09673.

[7] Jackson Kang, *Vae-tutorial*, Accessed: June 11, 2025. [Online]. Available: https://github.com/Jackson-Kang/Pytorch-VAE-tutorial?tab=readme-ov-file.