Project 3: Generative Models

Kinga Frańczak, Mateusz Wiktorzak

2025

Outline

Introduction

Generative models learn the data distribution $P(x)$ or joint distribution $P(x, y)$.

Generative Adversarial Networks (GANs)

- Introduced by Goodfellow et al. (2014)
- Two networks: Generator G, Discriminator D
- Minimax objective:

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

- Variants:
  - DCGANs – Deep convolutional layers
  - WGANs – Wasserstein loss for stability
  - Common issues: mode collapse, training instability

Variational Autoencoder

- Introduced by Diederik P. Kingma and Max Welling (2022)
- Main objective: encoder maps points to the latent space with a probabilistic distribution (Gaussian)
- Loss function:
  - Reconstruction loss
  - Kullback-Leibler Divergence - measures difference between probability distributions
- Characteristics: smooth transition between to samples, blurry output images

Dataset

- Cat images dataset from Kaggle
- 29,843 RGB images, resolution 64×64

DCGAN Architecture

Generator, input: latent vector $z \in \mathbb{R}^{\text{latent\_dim} \times 1 \times 1}$

- `ConvTranspose2d(latent_dim, fm*8, 4, 1, 0)`
- `ConvTranspose2d(fm*8, fm*4, 4, 2, 1)`
- `ConvTranspose2d(fm*4, fm*2, 4, 2, 1)`
- `ConvTranspose2d(fm*2, fm, 4, 2, 1)`
- `ConvTranspose2d(fm, img_channels, 4, 2, 1)`

BatchNorm + ReLU after each (except output layer)

Output: $64 \times 64 \times 3$, scaled to [-1, 1] using `Tanh`

Discriminator, input: image $64 \times 64 \times 3$

- `Conv2d(img_channels, fm, 4, 2, 1)`
- `Conv2d(fm, fm*2, 4, 2, 1)`
- `Conv2d(fm*2, fm*4, 4, 2, 1)`
- `Conv2d(fm*4, fm*8, 4, 2, 1)`
- `Conv2d(fm*8, 1, 4, 1, 0)`

LeakyReLU (0.2) after each layer

BatchNorm used except for first and last layer

Output: scalar probability via `Sigmoid`

Training: Binary cross-entropy, Adam optimizer

Variational Autoencoders - Architecture

Encoder

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

Decoder

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

LeakyReLU (0.2) after each layer
Training: Adam optimizer

Training Dynamics DCGAN

- Batch size: 128, learning rate: 2×10^{-4}, 50 epochs

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
. . . (40 epochs omitted) . . .		
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: DCGAN training log with discriminator and generator losses

Generated Samples

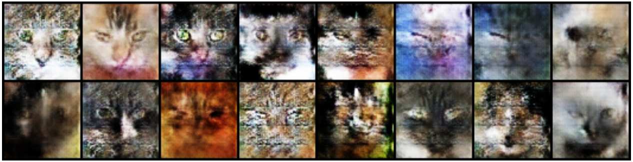
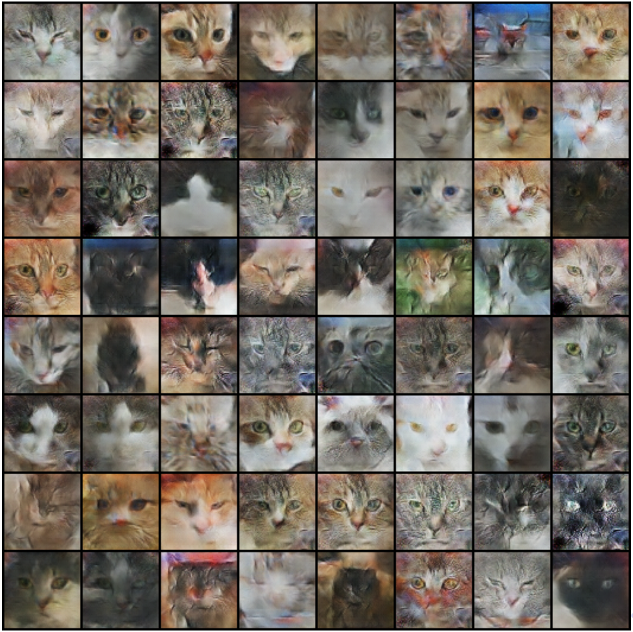


Figure: Images after 10 epochs

Figure: Images after 30 epochs

Final Generated Images



Further Evaluation

- FID:
  - Calculated FID between 1000 real and 1000 generated images
  - Lower FID = better image quality and diversity
  - Our FID: **132.7** (164.5 after first 10 epochs)
- Mode colapse:
  - Pairwise cosine similarity between 1000 generated samples
  - A similarity above 0.9 was considered highly similar.
  - Out of 999 000 possible pairs, 2412 were found to be highly similar.
- Linear interpolation between two randomly sampled latent vectors:

Variational Autoencoders - Training

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: Loss changes during VAE training

Variational Autoencoders - Random Sample

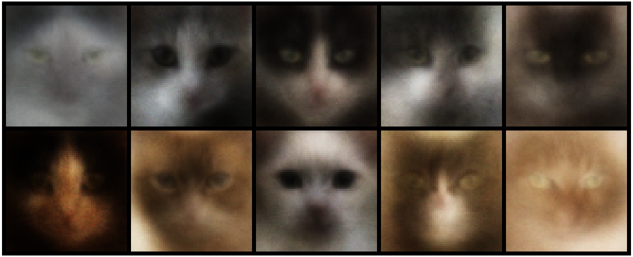


Figure: Images generated from random sample.

Variational Autoencoders - Interpolation

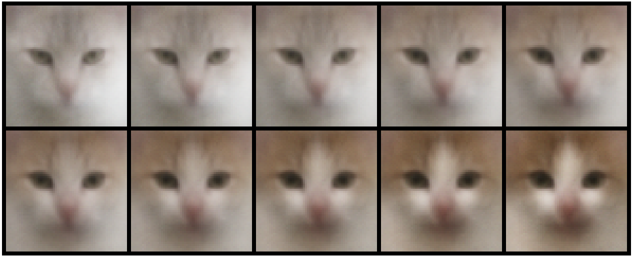


Figure: Linear Interpolation between two randomly sampled images

Conclusion

- DCGANs effectively generated realistic cat images
- Visual quality improved over epochs
- Training remained unstable at times
- VAE generates smooth-looking, blurry images

References

- Goodfellow et al., 2014
- Radford et al., 2015
- Arjovsky et al., 2017
- Heusel et al., 2018
- Diederik P. Kingma and Max Welling, 2022