DMSc in Data Science

Abstract Art Generation with GANs

By Alexios Gkolovko
MSc Student

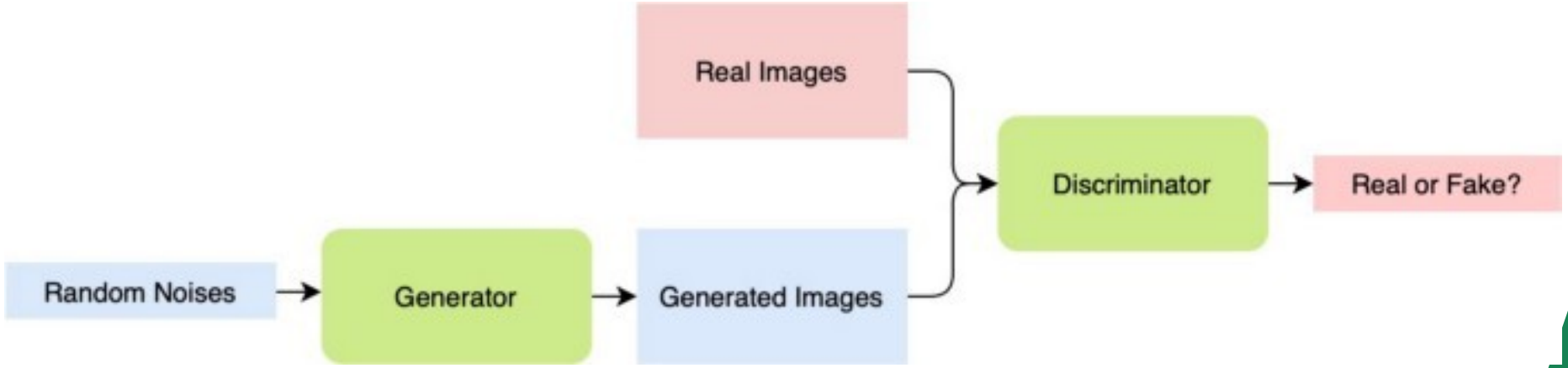
Project Objectives

- The implementation and training of a Generative Adversarial Network (GAN) that will be able to create abstract paintings by itself.
- The evaluation of the implemented GAN.

GANs

- Generative Adversarial Network (GAN)
- Class of Machine Learning frameworks
- Introduced by Ian Goodfellow and his colleagues in June 2014
- Generator: produces artificial data (usually images) based on a given dataset
- Discriminator: tries to distinguish genuine from generated data

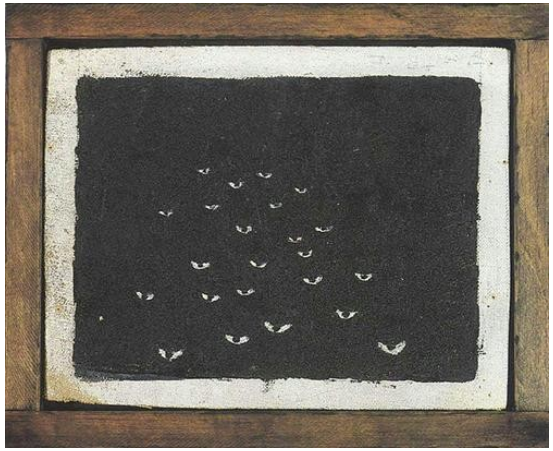
GAN example

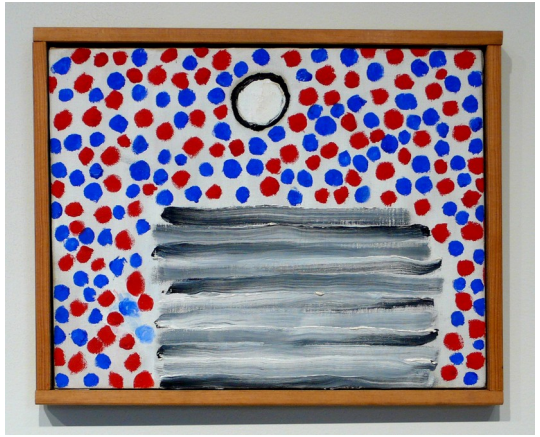


*source:*https://towardsdatascience.com/building-a-gan-with-pytorch-237b4b07ca9a

Abstract Art Gallery Dataset

- Size: 727MB
- 2.782 images, depicting paintings by abstractionists
- Large Dataset, slow training.
- Solution: use subset
  - Size: ~ 80MB
  - 571 images





Tools

- Matplotlib for the diagrams
- Tqdm
- IPython and, the most fundamental,
- ***PyTorch*** (including torchvision)

Preprocessing

- Data transformation
  - Resizing: to have pictures of the same size
  - Cropping: to remove elements that could serve as noise (frames)
  - Normalization (and denormalization): to improve the performance

Implementation of Discriminator and Generator

Discriminator

- Uses Convolutional Neural Networks (CNNs) to distinguish genuine from fake paintings.
- Leaky ReLU instead of ReLU: allows the pass of a small gradient signal for negative values, so the gradients from the discriminator flow stronger into the generator.

Generator

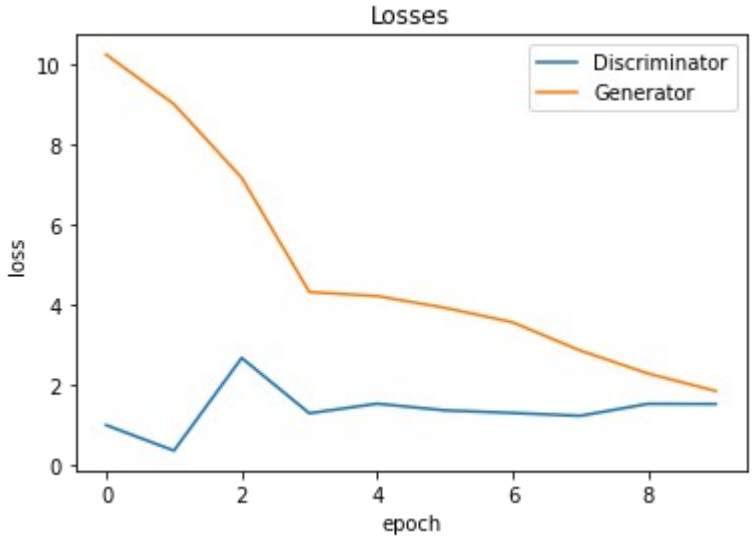
- ConvTranspose2d layer from PyTorch
- Latent tensor (vector of random numbers as seed to generate images)

Training

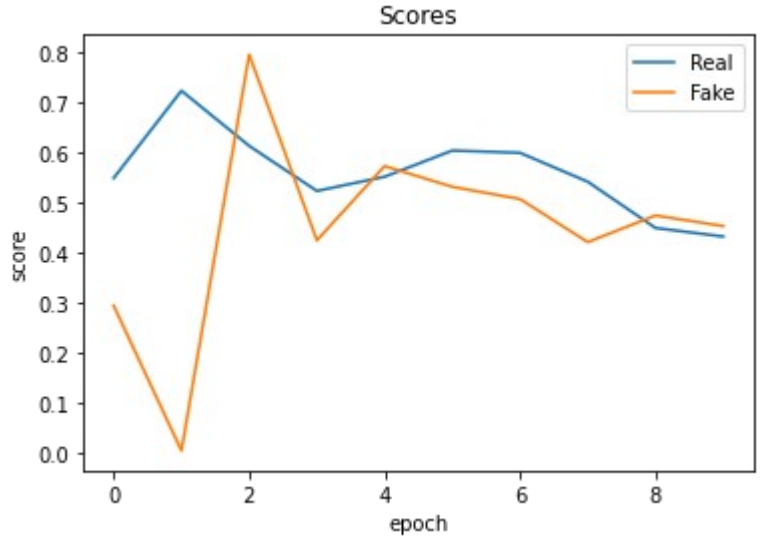
- No of epochs: 10
- Evaluation Measures:
  - loss of the discriminator
  - loss of the generator
  - correctly classified genuine images
  - correctly classified fake images
- Completion time: ~1h

Results: Diagrams

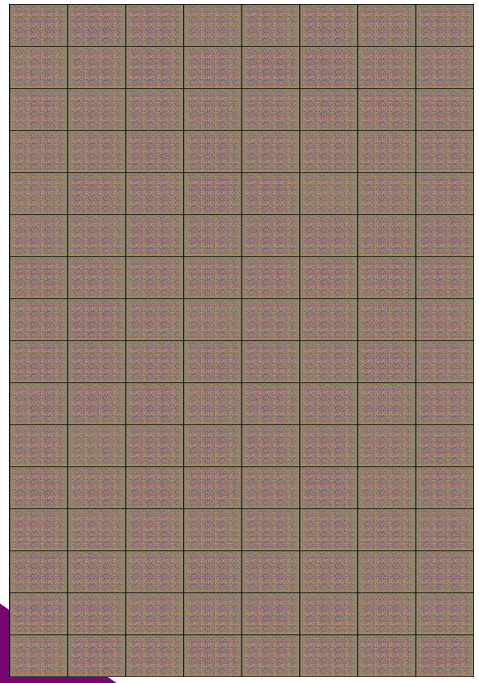
Loss-to-Epoch diagram

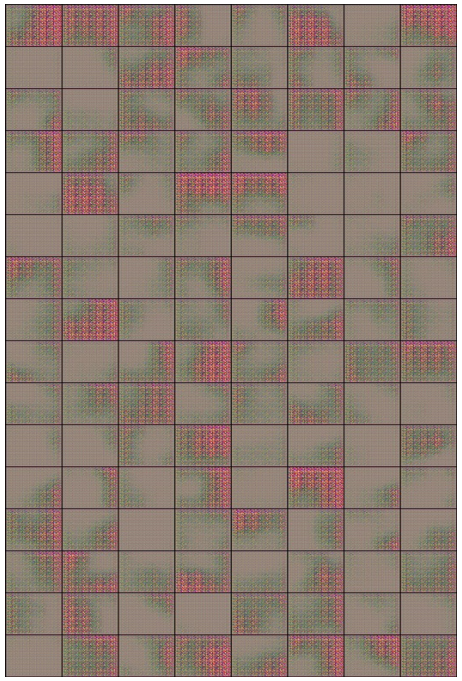


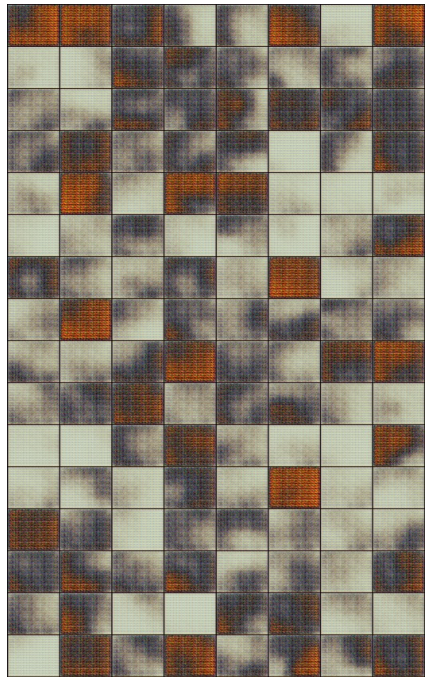
Score-to-Epoch Diagram



Results: Generated Paintings







Conclusion

- It seems that the generated pictures improve in quality.
- The generator’s performance improves by epoch (the loss decreases)
- The discriminator’s performance is stable (decreases slightly)
- It seems that training in more epochs would lead to an improvement in the quality of the generated pictures and an overall improvement in the performance of the generator.

Any Questions ?

Thank You!