Training a Generative Adversarial Network for Artistic Image Generation on the WikiArt Dataset

Alissa Jouljian

December 10, 2024

Abstract

GANs have been performing exceptionally well in generating synthetic images of high quality. As part of this work, we implement and train a GAN to generate artistic images using the WikiArt dataset composed of more than 80,000 images from diverse styles and genres. We describe with great detail the preprocessing of the dataset, model architecture, and training procedure. Finally, we evaluate performance and discuss possible improvements to the model.`https://github.com/alissajouljian/GAN`

1 Introduction

Recent advances in generative modeling have been nothing short of phenomenal with the discovery of Generative Adversarial Networks. In general, GANs involve two neural networks: a generator and a discriminator, which are trained simultaneously in a zero-sum game framework. A generator learns to generate realistic images, while a discriminator is used to distinguish between real and synthetic images. Over time, a generator will improve until it synthesizes images that are undistinguishable from the actual dataset.

The project pertains to training a GAN on WikiArt, a dataset consisting of more than 80,000 images of artwork, organized by genre and style; it is highly diverse and complex, thus very interesting for training generative models. Our goal will be to create visually plausible artworks while managing to preserve diversity in artistic styles.

2 Dataset and Preprocessing

The WikiArt dataset is openly available and contains images from 27 different artistic styles, including but not limited to *Impressionism*, *Abstract Expressionism*, and *Renaissance*. The dataset size is about 25GB, with images in varying resolutions.

To prepare the dataset for training, we performed the following preprocessing steps:

- Resizing all images to 64×64 pixels for consistency with the model architecture.
- Normalizing pixel values to the range $[-1, 1]$ to match the output of the generator's activation function.
- Organizing the images into train and test subsets.

The preprocessing pipeline ensures efficient loading and uniform input dimensions. We used PyTorch's `ImageFolder` and `DataLoader` utilities for dataset management.

3 Model Architecture

The GAN comprises two main components:

1. **Generator:** The generator starts with a random noise vector sampled from a standard normal distribution. This vector is progressively upsampled using transpose convolution layers, batch normalization, and ReLU activations to generate an image of size $3 \times 64 \times 64$.

2. **Discriminator:** The discriminator is a convolutional network that downsamples input images to a scalar output, representing the probability of the image being real. It uses convolutional layers, LeakyReLU activations, and batch normalization.

The generator employs a `Tanh` activation in the final layer to scale outputs to $[-1, 1]$, matching the normalized image range. The discriminator uses a `Sigmoid` activation to produce probabilities.

4 Loss Functions and Optimization

The GAN's training process is guided by two loss functions:

1. **Discriminator Loss:** The discriminator maximizes the probability of correctly classifying real and fake images. Its loss is computed as:

$$L_D = -\mathbb{E}[\log D(x)] - \mathbb{E}[\log(1 - D(G(z)))]$$

   where $D(x)$ is the discriminator's output for real images, and $D(G(z))$ is its output for generated images.

2. **Generator Loss:** The generator's objective is to minimize the discriminator's ability to distinguish fake images, effectively maximizing $\log(D(G(z)))$. Its loss is given by:

$$L_G = -\mathbb{E}[\log(D(G(z)))]$$

The Adam optimizer was used for both networks with a learning rate of 0.0002 and $\beta_1 = 0.5$, $\beta_2 = 0.999$. These hyperparameters are widely adopted in GAN training for stability.

5 Training Process

The GAN was trained on batches of 128 images for 400 epochs. During each iteration:

- The discriminator was trained to maximize its ability to classify real and fake images.

- The generator was updated to produce more realistic images by backpropagating through the discriminator's gradients.

Sample images were generated and saved at the end of each epoch to monitor progress.

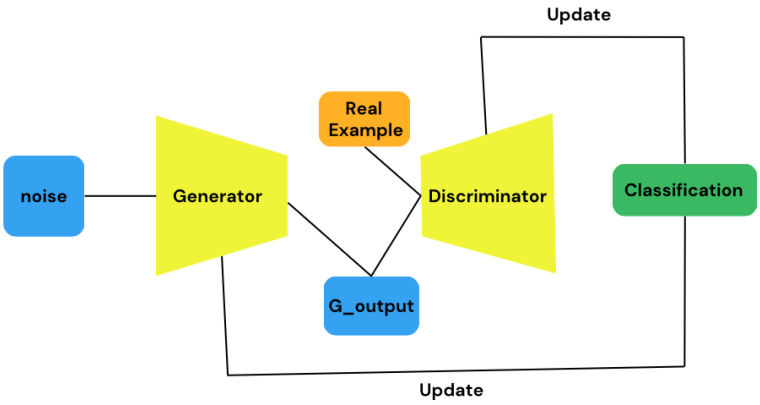

Figure 1: Images generated after 400 epochs.

6 Results and Discussion

This generator has seen significantly better times during training, generating visually quite plausible images. Noisy domination in early outputs gave way to distinct artistic patterns in later epochs. The discriminator loss converges with time.

Table 1 summarizes the quantitative results based on discriminator accuracy and generator loss.

Epoch	Discriminator Loss	Generator Loss
1	0.8068	1.9141
25	0.3722	3.3951
50	0.9423	0.4805

Table 1: Training results at various epochs.

In Figure 2, we observe the progressive improvement of the image quality as the training advances from 0 to 50 epochs. Initially, at epoch 0, the image starts with a rudimentary representation, lacking clarity and detail. However, as the training continues, the image undergoes noticeable enhancements, with each epoch contributing to increased refinement and better-defined features.

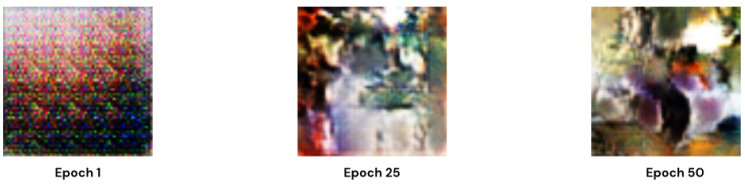

Figure 2: Generated images progress.

7 Conclusion and Future Work

This paper presented a GAN trained on the WikiArt dataset to generate artistic images. The results demonstrate the model's ability to learn complex artistic distributions. However, there is room for improvement:

- Increasing the image resolution to 128×128 or higher.
- Incorporating conditional GANs to control style or artist generation.
- Using more advanced architectures such as StyleGAN for higher-quality results.

References

1. I. Goodfellow, et al. "Generative Adversarial Networks." *Advances in Neural Information Processing Systems*, 2014.

2. A. Radford, et al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *arXiv preprint arXiv:1511.06434*, 2015.

3. WikiArt Dataset: `https://www.wikiart.org/`