WikiArtGAN

By: *Jacob Frausto*

Project Idea

Generative Adversarial Networks (GANs) are a framework for teaching a DL model to capture the training data's distribution in order to generate new data from that same distribution. They are made of two distinct models, a generator and a discriminator.

My goal in this project was to form a direct extension of a GAN that explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively. It was first described by Radford et. al. in the paper Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks [1]. The discriminator is made up of strided convolution layers, batch norm layers, and LeakyReLU activations. The input is a 3x64x64 input image and the output is a scalar probability that the input is from the real data distribution. The generator is comprised of convolutional-transpose layers, batch norm layers, and ReLU activations. The input is a latent vector, zz, that is drawn from a standard normal distribution and the output is a 3x64x64 RGB image. The strided conv-transpose layers allow the latent vector to be transformed into a volume with the same shape as an image. In the paper, the authors also give some tips about how to setup the optimizers, how to calculate the loss functions, and how to initialize the model weights, all of which will be explained in the coming sections.

By training this model on the WikiArt dataset, I was able to produce resulting generated images that genuinely embody elements of real artwork.

Data

The data for this project contains 81446 pieces of visual art from various artists, taken from WikiArt.org, along with class labels for each image. These class labels include "style", with 27 classes included.

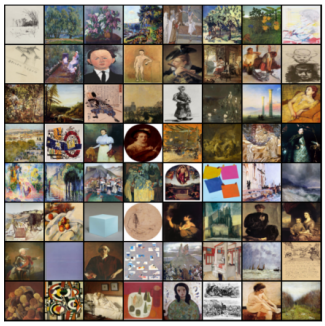

Figure 1: 64 example images pulled from the WikiArt dataset.

Hardware Used

Training took approximately 50 hours total to complete running on a Google Colab notebook with a singular NVIDIA Tesla K80 GPU.

Method

In short, the method for this project included performing the following in order:

1. Image Preprocessing
2. Train the Generator
3. Train the Discriminator
4. Evaluate Results

Model

The model is a GAN, so it has a generator and a discriminator.

The generator is designed to map a latent space vector to data-space. Since our data are images, converting this vector to data-space means ultimately creating a RGB image with the same size as the training images (i.e. 3x64x64). In practice, this is accomplished through a series of strided two dimensional convolutional transpose layers, each paired with a 2d batch norm layer and a ReLU activation. The output of the generator is fed through a Tanh function to return it to the input data range of [-1,1][1,1]. It is worth noting the existence of the batch norm functions after the conv-transpose layers, as these layers help with the flow of gradients during training.

The discriminator is a binary classification network that takes an image as input and outputs a scalar probability that the input image is real (as opposed to fake). Here, it takes in a 3x64x64 input image, processes it through a series of Conv2d, BatchNorm2d, and LeakyReLU layers, and outputs the final probability through a Sigmoid activation function. It is a good practice to use strided convolution rather than pooling to down-sample because it lets the network learn its own pooling function. The BatchNorm2d and LeakyReLU layers promote healthy gradient flow which is critical for the learning process of both the generator and discriminator.

Image Preprocessing

Three transformations are applied to each image in the dataset. First, each image is resized to 64x64. Then, the image is tensorized and normalized with mean and standard deviation.

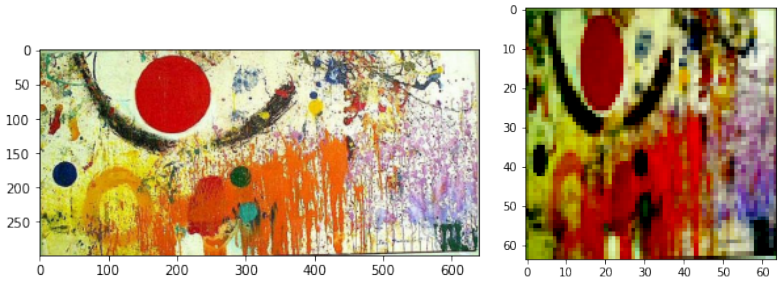

Figure 2: Left: An example image before preprocessing. Right: The same image after preprocessing.

Training

We train the model for 20 epochs using Keras *fit* function. We use a batch size of 128, a latent size of 128, and a learning rate of 0.001.

Results

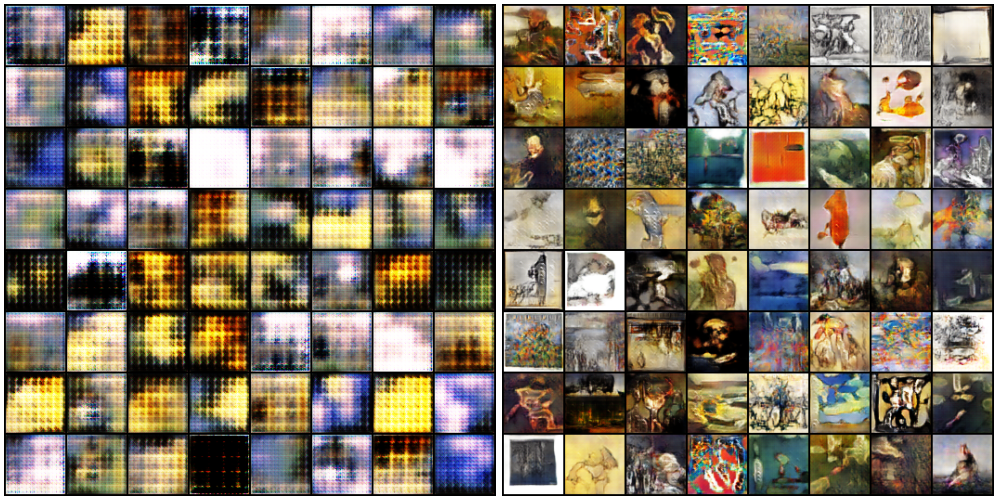

Figure 3: Left: The image generated after the first epoch of training. Right: The image generated after training has concluded.

Overall, the results of each experiment were satisfying and made sense. The model clearly learned the features of the art in the WikiArt dataset. As we can see in Figure 3, the final generated images after training embodies many aesthetic elements of real artwork. These include elements such as color, line, shape, form, texture, and composition. However, it's easy to see that there typically isn't a subject or any real objects that one can make out in these images. Some even have noise from the latent space.

A video timelapse of the images generated during training can be found at this link: https://drive.google.com/file/d/1-Ukgtm7vulL_YdVLldquyihAgHevasV2/view?usp=share_link.

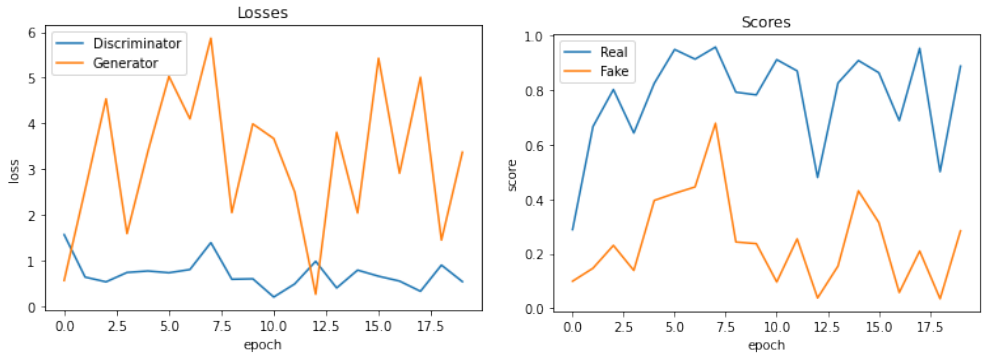

Figure 4: Left: Losses over 20 epochs. Right: Scores over 20 epochs.

Looking at Figure 4, I have to say that the losses and scores determined from the *train* function do correspond to each other, but are not so dependable as they don't seem to

converge. Looking at the intermediate and later images generated, it is evident that the model is indeed learning and trying to generate artwork from scratch on its own.

Conclusion

All in all, I am quite satisfied with the outcome of this project. The results show the efficacy of the model and the robust nature of art generation. I realize that the lack of tangible objects within the generated images likely comes from the fact that the model isn't able to discern between art style classes. The result is a strange blend of art styles that incorporate aspects of all kinds without ever converging to something tangible.

Further work will be done limiting the model's training to just a single style class.

References

[1] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2015. DOI: 10.48550/ARXIV.1511.06434. URL: https://arxiv.org/abs/1511.06434.