P2_GAN

Saturday, April 3, 2021 11:40 PM

Model:

For the GAN architecture, I used these hyperparameters to yield optimal results: (Hidden layers: 2, Batch size: 16, Epochs: 30, Learning Rate = 0.0001)

Discriminator:

For the discriminator network, I had two intermediate Linear layers of dimensions 128 and 32, with a LeakyReLU (which accepts small negative inputs) and Dropout following each layer. Then, since the discriminator's output must be a 1 (Real) or 0 (Fake), I used a sigmoid for the output layer.

Linear (200->128)
LeakyReLU(0.2)
Dropout(0.1)
Linear(128->32)
LeakyReLU(0.2)
Dropout(0.1)
Linear(32->1)
Sigmoid()

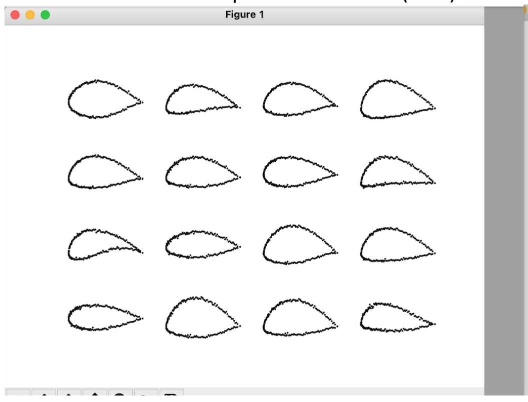
Generator:

For the generator, my architecture is as follows:
Batch norm is used to normalize the input to each layer.

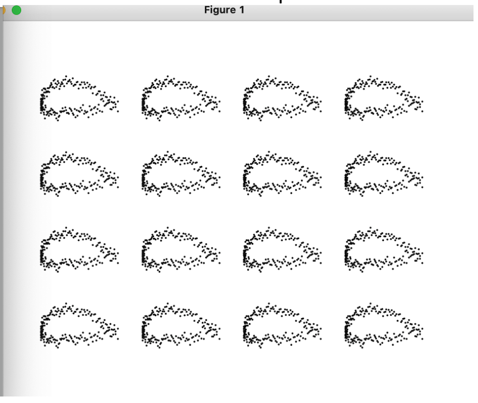
Linear (32 -> 48)
LeakyReLU(0.2)
Linear(48->64)
BatchNorm1D(64)
LeakyReLU(0.2)
Linear(64->128)
BatchNorm1D(128)
LeakyReLU(0.2)
Linear(128->200)
Tanh()

Training:
After experimentation I found that multiple runs of the discriminator (5-6 runs) for every one run of the generator significantly strengthens the output airfoil images, although training takes much longer.

5-6 Discriminator Runs per 1 Generator (best)

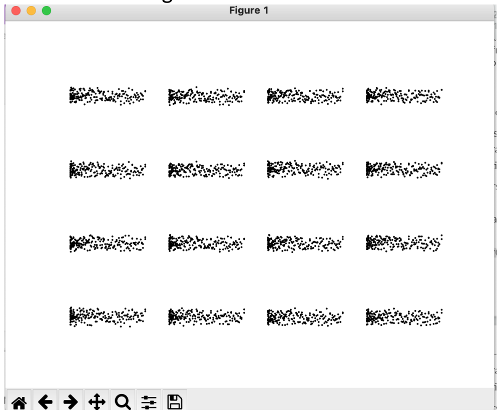


1 Discriminator Run per 1 Generator

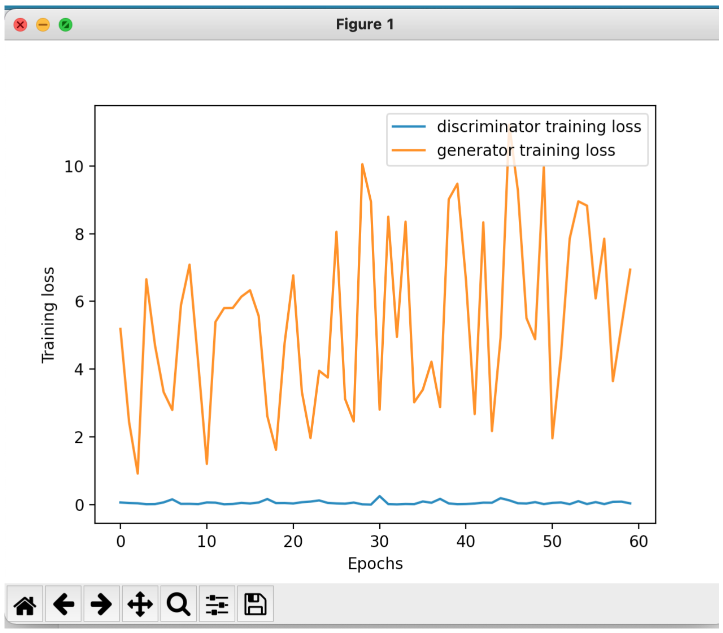


Optimizer: I am using Adam optimizer with learning rates of 0.001. I experimented with SGD with momentum but the output airfoils became indistinguishable in shape (shown below). Using unbalanced learning rates between the discriminator/generator did not impact the loss as much as I had anticipated.

Using SGD with Momentum

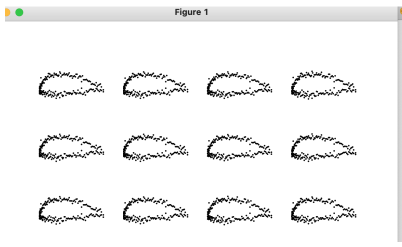


Utils:
In my utils.py, I created helper functions to 1) create real labels, 2) create fake labels, and 3) generate noise of dimension latent_dim.

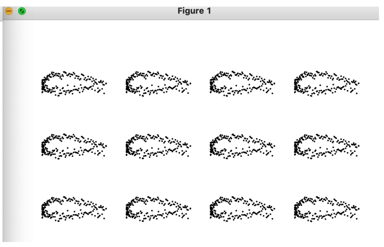


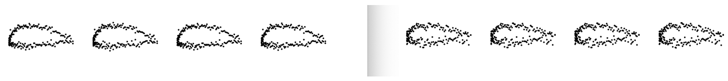
Epochs:
I found that using 60 epochs is optimal; using fewer epochs doesn't let the generator learn enough (hence the discriminator will do well and the generator performs badly). Using too many epochs may make the discriminator too good, which doesn't provide the generator enough information to make any progress.

75 epochs

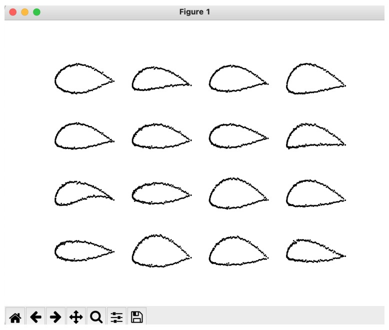


10 epochs

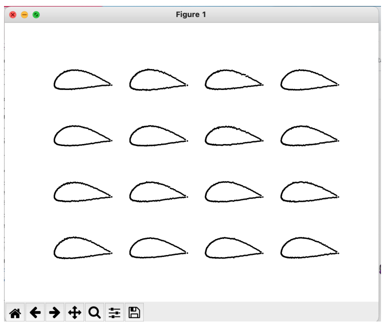




GAN (Final)



VAE (Final)



Observations:

- VAE uses an encoder to encode an input into latent_dimension, then uses a decoder to "reconstruct" an output that tries to match the input. On the other hand, GAN can be thought of as a minimax/adverserial system where the discriminator/generator are competing
- VAE-generated airfoils are much cleaner and less noisy than the GAN, and only took 30 epochs (whereas GAN took 60). I hypothesize that this is because GANs are much more difficult/complex to train, since there are many factors that have to be considered in order to maintain balance between the generator/discriminator.
- VAE takes much less time (~1 minute), where training/testing the GAN took around 4.5 minutes to run 60 epochs.