CS 7180 Special Topics in AI: Deep Learning
Northeastern University, Spring 2019

Assignment 3

Name: Sabbir Ahmad, NUID: 001860911
Email: ahmad.sab@husky.neu.edu

1 Sketch Generation

1.1 Rendering

Rendering is done using the provided script `draw_strokes.py`. Each sketch sample can be rendered using the function `draw_strokes`. For a grid of sketch samples `make_grid_svg` and `draw_strokes` both functions were used. For the grid, each sample is first normalized so that the values are between [-1,1]. An example grid of sketches is shown in Figure 1.

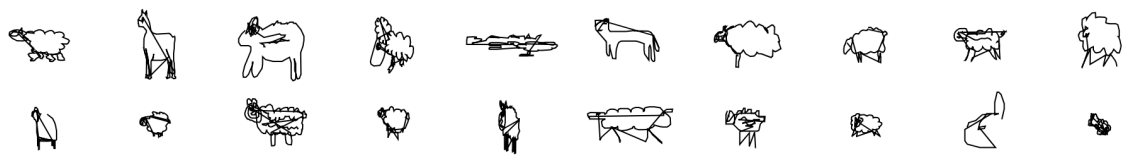


Figure 1: Rendering example of 20 sketch samples as a grid.

1.2 Pre-processing

For pre-processing, each sample of sketches is padded with zero such that each sample is of shape [250, 3], as 250 is the most data points in any sample. Each sample is also normalized so that each value in the sample data is in range [-1,1]. Each sketch is considered as a sequence of 250 length timestamp with 3 features at each timestamp. The input to the model is of shape [7400, 250, 3] for the training set.

1.3 GAN Model

1.3.1 Model Training

The baseline model uses LSTM generator and discriminator. For both the generator and the discriminator, 1 layer for LSTM is used. The generator has fully-connected layer with *tanh* activation at the output. The discriminator has two fully-connected layer and the final output is just one neuron defining real or fake sketches.
Training the GAN did not converged as can be seen from the training loss curve in Figure 2. The discriminator loss became very low and the generator loss became very high.
Some other settings of the network was changed (e.g. chaning number of LSTM layers, learning rates, etc) to see how the output changes, but the network seem not to converge. The sketches generated from this generator using the same noise input at different epochs

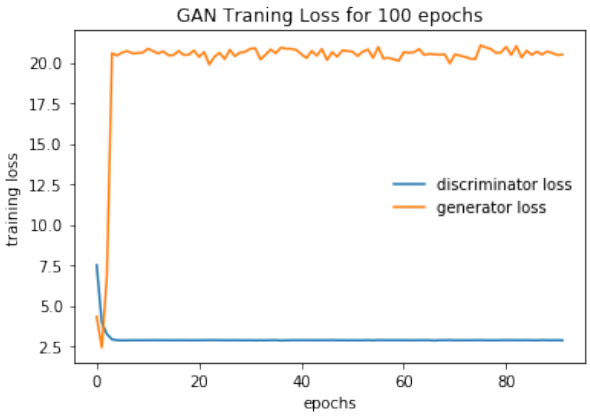
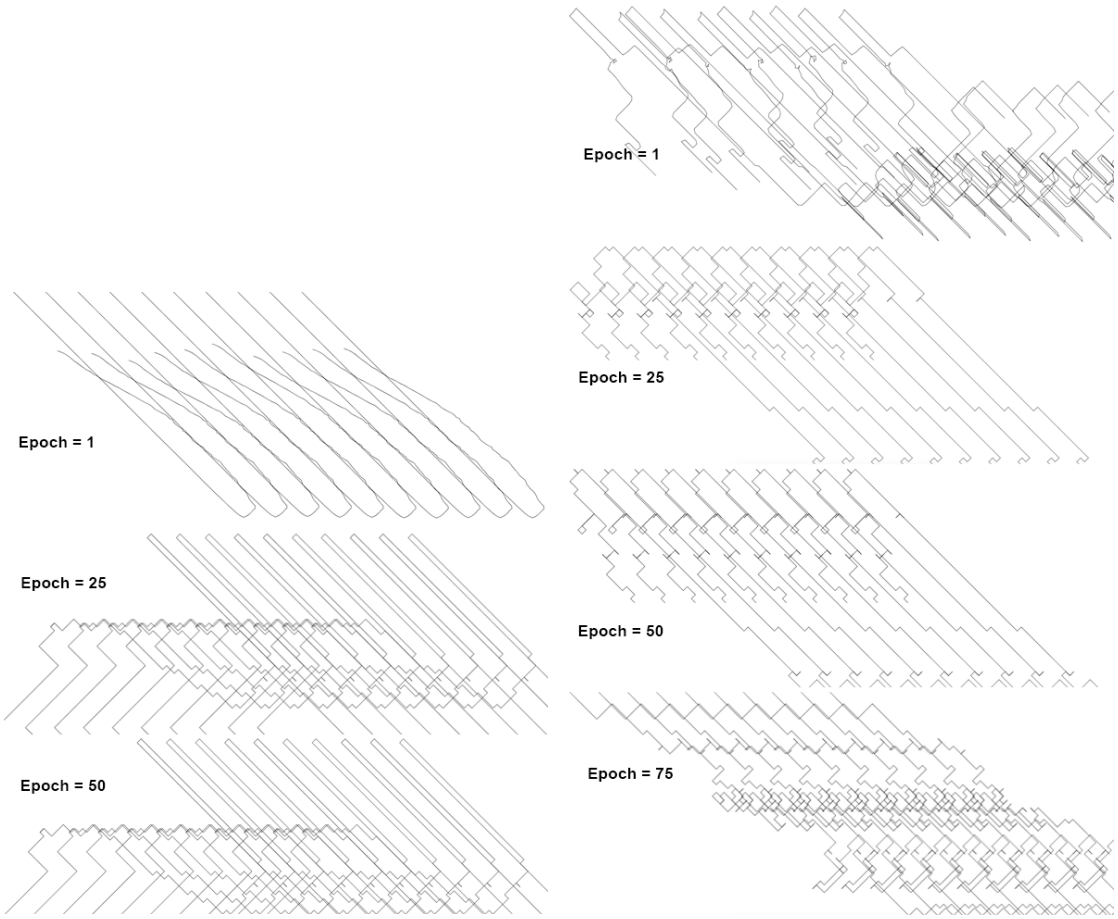


Figure 2: Baseline GAN training loss for generator and discriminator.

look like mostly straight lines as shown in Figure 3. Note that, each epoch sketch contains 10 sketches in the figure shown. All of them were very similar with some changes. Another thing to mention is that I have run the codes in Google CoLab, and while saving the notebooks, the svg images are not shown if opened in the local computer. I have attached the sketches in the report as shown in Figure 3.



(a) Model with 1 LSTM layer (b) Model with 2 LSTM layers

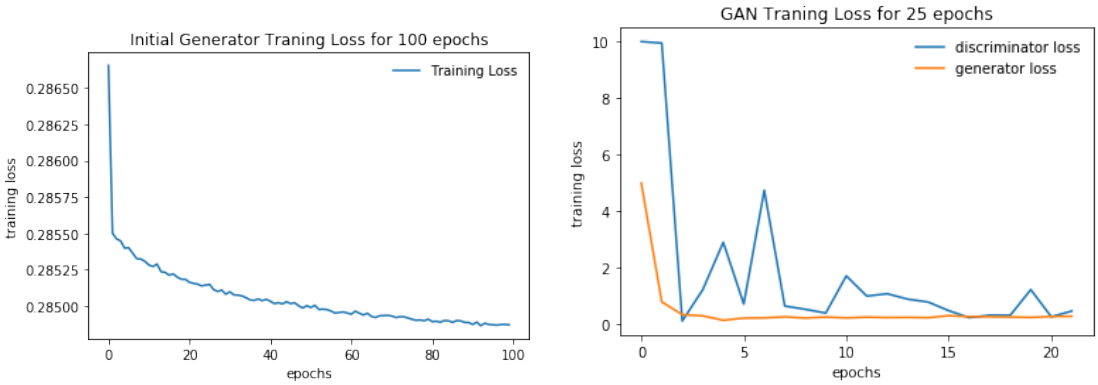
Figure 3: Baseline Model sketch generation at different epochs for same noise input.

1.3.2 Model Improvement

For improvement, many combinations has been tried as stated below.

1. Using pre-trained generator with L^2 reconstruction loss.
2. Using bi-directional LSTM for generator.
3. Changing the distribution of the generated noise.
4. Using different target values for the fake and real sketches for discriminator.
5. Using different optimizers.
6. Traning discriminator more than generator.

Using different combinations of the previous mentioned modifications some results are shown below. Figure 4(a) contains training loss for uni-directional and bi-directional generator using L^2 reconstruction loss.
Using the pre-trained generator, the GAN was trained. The training loss for both discriminator and generator is shown in Figure 4(b).



(a) Bi-directional LSTM generator pre-training loss

(b) Bi-directional LSTM model GAN training loss for discriminator and generator

Figure 4: GAN loss over epochs.

.

Example sketches at different iteration for same noise input are shown in Figure 5. This model seems to converge from the training loss curve as discriminator and generator loss are at similar level with a few spikes. But the generated sketches was not very good, although better than the baseline model.

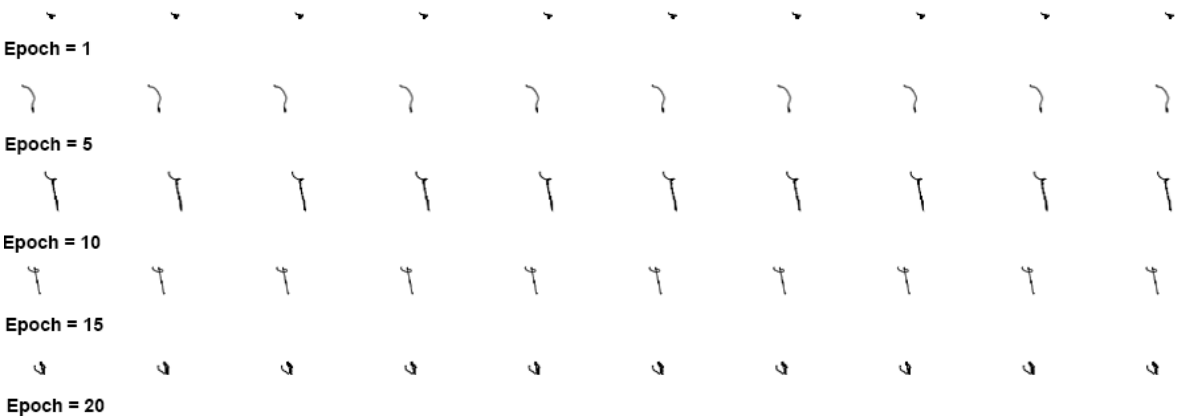
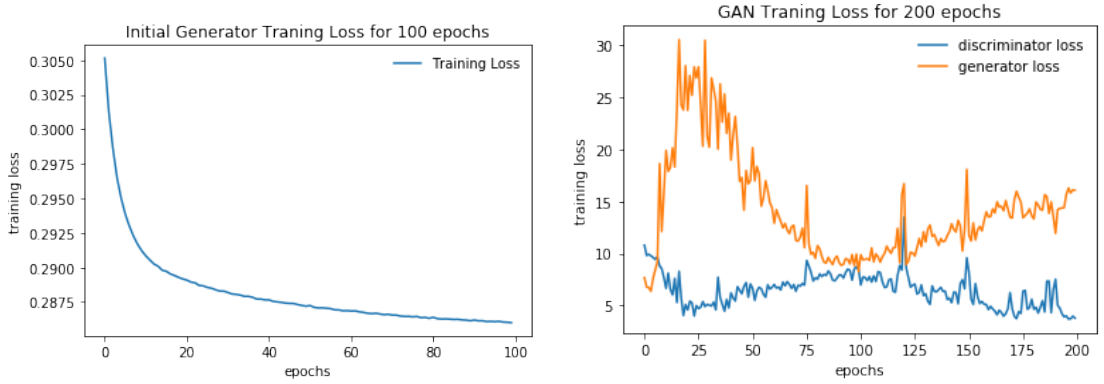


Figure 5: Improved GAN example sketchs at different epochs for same noise input.

1.3.3 Improve using a fully-connected architecture

Rather than using LSTM architecture, a simple architecture with fully-connected neural network is used for both discriminator and generator to see how this affects the sketches. Both the discriminator and generator uses three layer fully-connected neural networks with leaky ReLU activation function. For the final layer, the generator uses *tanh* and the discriminator uses *sigmoid* activations.
The generator was pre-trained and the training loss for the pre-training is shown in Figure 6(a).
Similarly as before, using the pre-trained generator the GAN was trained. Figure 6(b) shows that the GAN seems to converge around epoch 100.



(a) Traning loss for pre-training generator with fully-connected network

(b) Traning loss for fully-connected model discriminator and generator in GAN

Figure 6: GAN loss over epochs.

.

Sample sketches generated from this GAN is shown in Figure 7.
Reference: Some help has been taken from this website: link

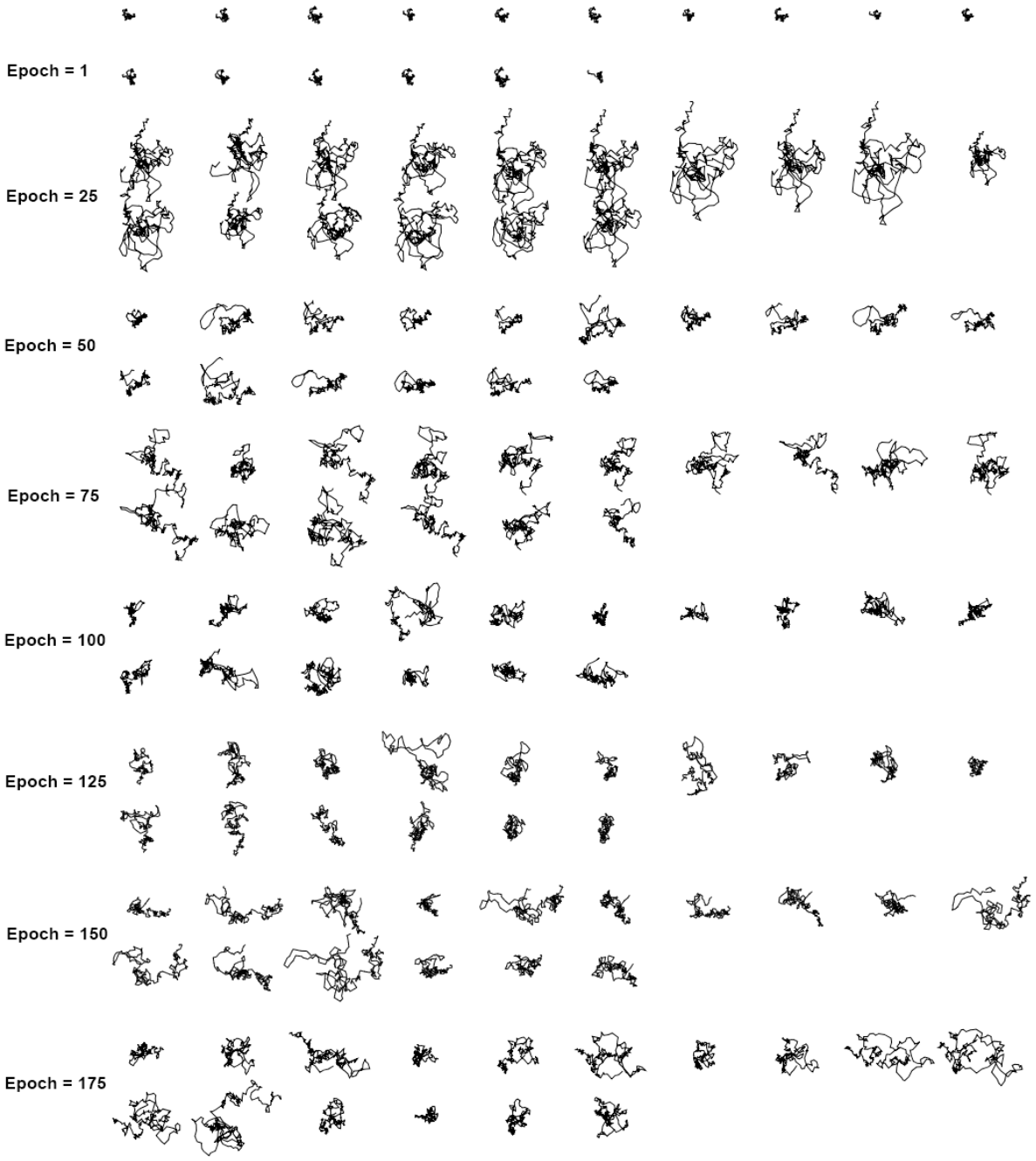


Figure 7: GAN example sketches for fully-connected generator model at different epochs for same noise input.