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

Assignment 3

Name: Tyler Brown UID: 001684955

The goal of Assignment 3 is to train a generative adversarial network that can draw sketches. My implementation is based on "A Neural Representation of Sketch Drawings" [1] by Ha and Eck (2017).

1 Rendering

Figure 1 contains an image of a sheep that I rendered using the *draw_strokes.py* script provided to us in this assignment.

Figure 1: Rendering a Sheep Image

2 Pre-processing

The pre-processing steps I used were very similar to those by Ha and Eck [1]. Alexis David Jacq, a PhD student and intern at Google Brain, implemented the Ha and Eck model using PyTorch [2]. It make sense to include many of his pre-processing steps in my model because is processing identical data. The Ha and Eck paper [1] is so similar that their README file on GitHub includes the graphic used in Section 2.1 of our Assignment 3 handout.

Pre-processing steps are broken out into my *SketchDataPipeline* class in *sheepgan.py*. These steps include

1. Finding the maximum sequence size for additional preprocessing
2. Purifying strokes to remove sequences that are too small or too long. It also removes large gaps.
3. Normalize the dataset by a normalizing scale factor.

In terms of modifying Ha and Eck [1] to be compatible with a GAN, I added a batch generation method, and a special type casting method for output from the generator. The batch generation method for my GAN model randomly selects a batch of normal data, keeps the sequence lengths and batch dimensions, then replaces all normal data with randomly selected values within the range of values seen in normal data. You'll also see a "gencast_tensor" method which is unfortunately where I got stuck at the deadline. This casting method reformats output from the generator so it can be fed into the discriminator.

3 GAN

The GAN I chose to construct is really exciting to me because it attempted to modify Sketch-RNN, a Sequence-to-Sequence Variational Autoencoder (VAE) [1]. These models are typically seen as an opposite approach to a GAN model. Without getting into too many low level details related to Sketch-RNN, this model allows for image generation conditional on a batch item. If we create random data of the same dimensions as a normal batch item, then it is feasible that sketch-rnn could be used as a generator. Following the same idea, we can take the predicted strokes, reprocess the output, then feed that output into a discriminator. The discriminator can use the generator's output to conditionally generate its own image. We could also randomly use true images.We could then use a loss function to compute the difference between the discriminator's image and the valid image from true data. This loss function would update the GAN optimizer. When sketches created by the discriminator become closer to the valid image, then we should be able to approximate the underlying distribution.

Theoretical advantages of using a GAN on top of two Sequence-to-Sequence VAE's may help better approximate the loss function. My understanding is that the current loss function being used by these models is a proxy for an intractable loss function dictated by theory. Using a GAN to bring these approximate bounds in closer could yield better accuracy results.

I was able to create several outputs with the generator. I compared those with outputs from Jacq [2] using a cat instead of a sheep. His cats look better than my sheep so I'll most likely need to revisit my hyperparameters. Figure 2 provides an example of a sheep conditionally generated after 3400 epochs.

The intuition is to improve conditionally generated images from the Sequence-to-Sequence VAE using a GAN which can help better approximate its loss function. This intuition can be readily tested using an experiment. We can see that I was able to get the generator working without an issue. I was also able to get the discriminator to train independently. I had issues getting the handoff from generator to discriminator working correctly in the time we have available for this assignment.

4 Improving GAN

This GAN could be improved by debugging the dimensionality associated with a handoff from generator to discriminator. Once this is working, additional thought could be taken to choose a GAN loss function to best optimize this model. We can anticipate sequences of different lengths so choosing a loss function which is robust to these sequence length

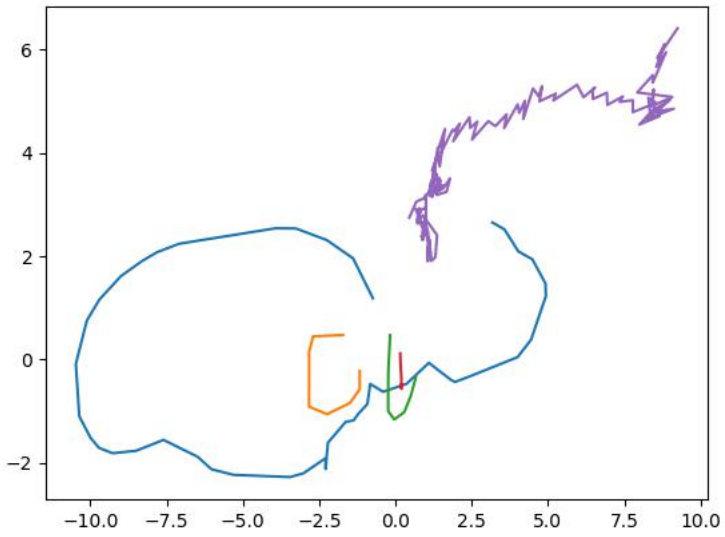

Figure 2: Generator after 3500 Epochs using True Data

variations will require careful thought.

I was able to improve over a basic GAN implementation by modifying a Sequence-to-Sequence VAE. I was able to get several components of this model working but not the whole thing. In terms of modifications, I did go through the Sequence-to-Sequence VAE and put substantial effort into figuring out a way that it could be modified for use as a GAN.

References

[1] David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017.

[2] Alexis David Jacq. alexis-jacq/pytorch-sketch-rnn: a pytorch implementation of https://arxiv.org/abs/1704.03477. `https://github.com/alexis-jacq/Pytorch-Sketch-RNN`. (Accessed on 04/14/2019).