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

Assignment 3 (10 Points)

Due 11:59 p.m. Sunday, April 14, 2019

Submission Instructions

Please submit your assignment as a .zip archive with filename `LastnameFirstname.zip` (replacing `Lastname` with your last name and `Firstname` with your first name), containing a PDF of your assignment writeup **in the main directory** with filename `LastnameFirstname_3.pdf` using the provided LaTeX template. (`http://roseyu.com/CS7180/latex_template.tex`). Submit your code as Jupyter notebook .ipynb files or .py files, and **include any images generated by your code along with your answers in the solution .pdf file.** We ask that you use Python 3.6 for your code, and that you comment your code such that the TAs can follow along and run it without any issues. Failure to do so will result in a **1 point deduction**.

1 Sketch Drawing

The Sheep Market is a collection of 10,000 sheep created by workers on Amazon's Mechanical Turk. Each worker was paid $.02 (US) to "draw a sheep facing left." These sketches are a unique data set that can help developers train new neural networks, help researchers see patterns in how people around the world draw, and help artists create things we haven't begun to think of. Figure 1 shows a few sketches from this dataset.

Figure 1: Example sketches of the Sheep Market dataset, created by Mechanical Turk workers who are asked to draw a sheep facing left.

The goal for this assignment is to train a generative adversarial network that can draw sketches. The sheep market dataset is available in the data file `sheep_market.npz`, provided in the data folder. Read the generative modeling chapter (Chapter 20) of the Deep Learning book for more details of GANs.

2 Dataset

2.1 Rendering (1 points)

Each sketch sample is stored as list of coordinate offsets, representing the strokes: horizonal offset $\triangle x$, vertical offset $\triangle y$, and a binary value representing whether the pen is lifted away from the paper. Figure 2 shows an example of a turtle drawing. The turtle is represented as a sequence of ($\triangle x$, $\triangle y$, binary pen state) points. In the rendered form, the line color corresponds to the sequential stroke ordering to illustrate the ordering. We also provide a visualization script file `draw_strokes.py` in the data folder to help you render the strokes.

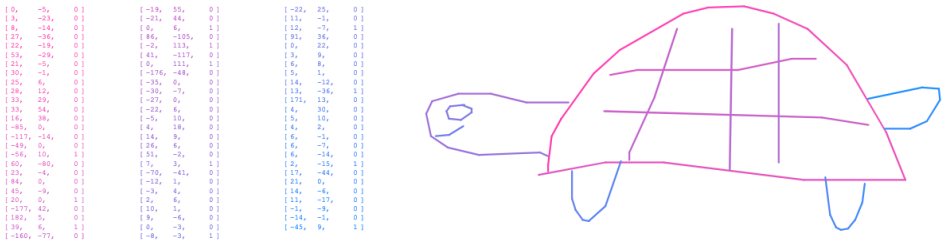

Figure 2: A turtle drawing in vector format (left) and the visualized sketch (right).

Note that the provided dataset is a simplied version of the original Sheep Market data, where we kept 8000 out of the 10,027 original sheep, and applied the RDP line simplification algorithm to get the samples below 250 data points. The train/validation/test split sizes are 7400/300/300 respectively.

2.2 Pre-processing (2 points)

Before training any neural network, the first step is to pre-process the dataset and prepare the inputs for the model. How you pre-process is completely up to you. Here are a couple of questions to help you decide how to do pre-processing: Each stroke-based sketch contains different number of points, thus is of variable length. How will you deal with sequences with variable length? It may be helpful to understand the concept of padding to make the sequences in a mini-batch of equal length. Do you normalize the data? How will you handle long sequences?

3 Generative Adversarial Networks

3.1 Model Training (4 points)

Try doing sketch generation using a generative adversarial network (GAN). Please follow these guidelines in this section:

- Train a **LSTM** generator to generate the sketch. You should also have a standard fully-connected output layer with an activation function.
- Train a **LSTM** discriminator to discrimniate between the generated fake sketch and the ground truth sketch.

- Your model should minimize the GAN objective function. Make sure that you train both the generator and the discriminator alternatively to avoid mode collapse. Keep close track of their training loss curves for correct behavior.
- To draw a sketch, draw samples from your trained GAN model conditioned on a prior noise distribution (usually a standard normal).

3.2 Improving generative models (3 points)

There are many ways you can improve the basic GAN model. For example, you can try to pre-train the generator using $L2$ reconstruction loss and use the pre-trained generator to warm-start the GAN model. You can also try to use bi-directional LSTMs, Wasserstein GAN or other advanced neural network architectures to enhance your generator. Evaluate your improvements over the baseline model both qualitatively and quantitatively (e.g. using the perplexity metric). You may use any publicly-available code and papers as long as you reference them appropriately.

4 Report

Rending Your report should contain a visualization of the dataset using the provided rendering script.

Pre-Processing Your report should contain a section dedicated to pre-processsing. Explain your choices, as well as why you chose these choices initially. What was your final pre-processing? How did you pad your sequences, and split up the data into separate sequences? What changed as you continued on your project? What did you try that didn't work? Also write about any analysis you did on the dataset to help you make these decisions.

GAN Explain in detail what model you implemented? What parameters did you tune? Visualize the training loss curve for the generator and the discriminator. Comment on the sketches that your model produced. Does the LSTM successfully learn sketch structure and/or drawing patterns? How about the runtime/amount of training data needed? .

Improving GANs Explain in detail your improvement over the basic GAN model. What module(s) did you change? What are the improvements both qualitatively and quantitatively? Feel free to be include visualizations and interpretations, but remember that accurately representing data is still your primary objective. Your figures, tables, and diagrams should contribute to a discussion about your model.

5 Additional Resources

- Google Quick Draw
- Draw together with a neural networks