ECE-GY 9123 / Spring 2021

Homework 6

Please upload your assignments on or before May 7, 2020.

- You are encouraged to discuss ideas with each other; but
- you **must acknowledge** your collaborator, and
- you **must compose your own** writeup and/or code independently.
- We **require** answers to theory questions to be written in LaTeX, and answers to coding questions in Python (Jupyter notebooks)
- Upload your answers in the form of a single PDF on Gradescope.

---

1. **(4 points)** In this problem we will see how minimax optimization (such as the one used for training GANs) is somewhat different from. For illustration, consider a simple problem, consider a simple function of 2 scalars:

$$\min_x \max_y f(x, y) = 4x^2 - 4y^2.$$

   You can try graphing this function in Python to visualize this (there is no need to include the graph in your answer).

   a. Determine the saddle point of this objective function. A saddle point is a point (x, y) for which f attains a local minimum along one direction and a local maximum along an orthogonal direction.

   b. Write down the gradient descent/ascent equations for this objective function starting from any arbitrary initial point.

   c. Determine the range of allowable step sizes for the ascent and descent required to ensure that the gradient descent/ascent converges to the saddle point.

   d. What if, instead of solving a minimax problem, you just did regular gradient *descent* for both x and y? Comment on the dynamics of the updates, and whether or not one would converge to the saddle point.

2. **(6 points)** In this problem, the goal is to train and visualize the outputs of a simple Deep Convolutional GAN (DCGAN) to generate realistic-looking (but fake) images of clothing.

   a. Use the FashionMNIST training dataset to train the DCGAN. APIs for downloading it are available in both PyTorch and TensorFlow. Images are grayscale and size 28×28.

   b. Use the following discriminator architecture (kernel size = 5×5 with stride = 2 in both directions):

      - 2D convolutions ($1 \times 28 \times 28 \to 64 \times 14 \times 14 \to 128 \times 7 \times 7$)
      - each convolutional layer is equipped with a Leaky ReLU with slope 0.3, followed by Dropout with parameter 0.3.
      - a dense layer that takes the flattened output of the last convolution and maps it to a scalar.

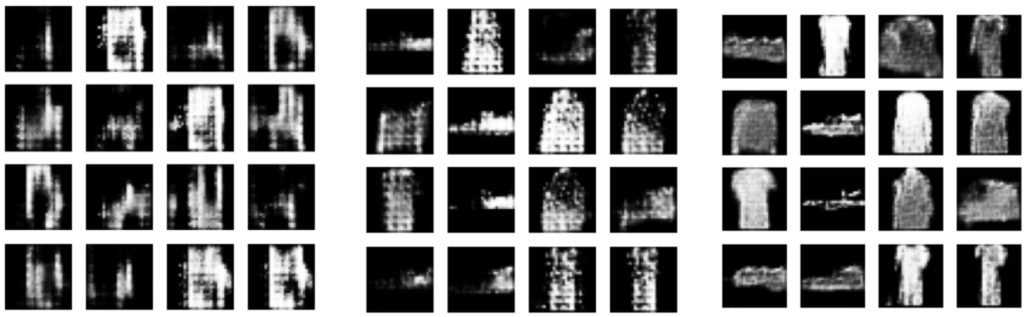
Here is a link that discusses how to appropriately choose padding and stride values in order to desired sizes.

c. Use the following generator architecture (which is essentially the reverse of a standard discriminative architecture). You can use the same kernel size. Construct:

- a dense layer that takes a unit Gaussian noise vector of length 100 and maps it to a vector of size $7 * 7 * 256$. No bias terms.
- several transpose 2D convolutions ($256 \times 7 \times 7 \rightarrow 128 \times 7 \times 7 \rightarrow 64 \times 14 \times 14 \rightarrow 1 \times 28 \times 28$). No bias terms.
- each convolutional layer (except the last one) is equipped with Batch Normalization (batch norm), followed by Leaky ReLU with slope 0.3. The last (output) layer is equipped with tanh activation (no batch norm).

d. Use the cross-entropy loss for training both the generator and the discriminator. Use the Adam optimizer with learning rate 10^{-4}.

e. Train it for 50 epochs. You can use minibatch sizes of 16, 32, or 64. Training may take several minutes (or even up to an hour), so be patient! Display intermediate images generated after $T = 10$, $T = 30$, and $T = 50$ epochs. If the random seeds are fixed throughout then you should get results of the following quality:



f. Report loss curves for both the discriminator and the generator loss over all epochs, and qualitatively comment on their behavior.