Richard Paredes
UHID: 1492535
COSC 4368

Problem Set 2

Task 5

1. **Tanh Activation and Scaling:**

   After updating the activation function of the generator from sigmoid to tanh along with scaling the dataset's pixel values from (0, 255) to (-1, 1), the generator seems to fail to produce any useful output. Based on the produced images, there were no images containing digits generated by the generator similar to the real handwritten digits from the dataset. The only marks on the output were miniscule markings that were hardly visible. Consequently, the discriminator correctly classified every generator's output as fake for every epoch. Additionally, the discriminator was also able to classify the real samples with 100% accuracy in every epoch. Interestingly, incorporating the tanh activation function also reduced the time to classify by roughly 2 minutes. Unfortunately, considering the average time to train the models was about 1hr 19min, this reduction in time is hardly significant. Perhaps the training process started poorly and propagated, allowing the generator to continue producing rubbish output. If this were identified earlier, the training would have been preempted and restarted to avoid wasting time producing a terrible model.

Figure 1: Generator using Tanh activation; output after 100 epochs

2. **Change Latent Space**

   Reducing the latent space by half, from 100 dimensions to 50, seemed to have very little impact on the quality of results and speed of training. Both the base GAN model from the tutorial and the GAN model with reduced latent space took 1hr 19m to train 100 epochs. The generator with reduced latent space was only faster by about 40 seconds. This is interesting, since the latent space represents the compression of the output space from the generator. With a reduced latent space, it was expected that the model may produce lower-quality output images or would take significantly less time (or both). Nonetheless, if examined very closely, the results of the updated generator model do seem to be slightly smoother with less noise in the produced

digits. Although extremely minor in difference, this result was not expected since the output space is being compressed even more. Moreover, the discriminator had an increased accuracy after the third epoch for the fake samples. On the other side, the discriminator started off with high accuracy for the real samples, but gradually became less accurate the more it was trained.

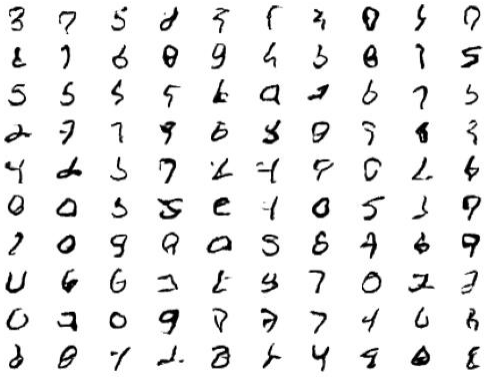

Figure 2: Generator using reduced latent space; output after 100 epochs

3. **Label Smoothing**

Somewhat similar to the results of the **Tanh Activation and Pixel Scaling**, the discriminator was able to correctly identify the fake samples with 100% accuracy throughout every epoch. However, the discriminator also had 0% accuracy for the real samples, which was interesting. Perhaps updating the label caused the discriminator to classify everything as fake. Nonetheless, the resulting images appear pretty well generated throughout the epochs, similar to the base GAN model. The more epochs the model was trained, the better the output produces in terms of similarity to real handwritten digits from the dataset. Therefore, perhaps updating the label and another confounding variable resulted in the discriminator to have poor output, potentially affecting some results of the experiment (even though the generator was still producing plausible output). This situation is like a reverse of what happened in the **Tanh Activation and Pixel Scaling** experiment, where the generator was the failing model. Re-running this experiment may be useful to see if the output is the same.

Richard Paredes
UHID: 1492535
COSC 4368

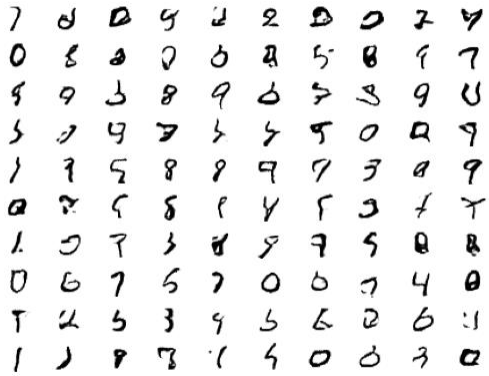

Figure 3: Generator using label smoothing; output after 100 epochs

4. **How to make generator broaden its scope?**

   The discriminator may get stuck in a local minimum or have vanishing gradients, allowing the generator to optimize for that particular discriminator in that generation. This optimization would result in the generator producing the same plausible output without variation. One way to solve this is to update the discriminator and/or generator's loss functions. By having a better discriminator loss function, the discriminator can prevent vanishing gradients and continue training and learning to reject generator output that was optimized towards the discriminator. Similarly, updating generator loss function can be used to prevent the generator from optimizing towards a single generation of the discriminator, allowing it to produce variable outputs rather than focusing on a single, plausible one.

5. **How to improve GAN convergence?**

   As the generator trains, the discriminator begins to have trouble determining between real and fake data. By this point, the discriminator basically randomly chooses whether a sample is real or fake, and doesn't provide meaningful feedback to the generator, preventing the GAN from converging. In fact, since the generator is learning based on random feedback, the quality of the model may be reduced. Therefore, finding ways to improve convergence is important. One way to do this is by introducing noise to the discriminator's input samples so that the training happens more gradually and is meaningful to the generator.

6. **Adding Batch Normalization**

   Based on the results, introducing batch normalization to the generator model seemed to have improvements on the overall training of the GAN. Specifically, as the discriminator seemed to improve in some early epochs, the generator was able to train with this feedback, thereby decreasing the discriminator's overall accuracy in the next generation. This may serve as an indication that the generator was producing plausible output that the discriminator struggled to classify. Unfortunately, the batch normalization process did also result in longer training time.

Richard Paredes
UHID: 1492535
COSC 4368

Every epoch took roughly one minute longer. Thus, the total time to train for 10 epochs was 1hr 29min, which was about 10 minutes longer than the GAN without batch normalization. Nonetheless, considering the accuracy results seemed better overall, the increased time was a worthwhile investment.

Figure 4: Generator using batch normalization; output after 100 epochs