CS552: Generative AI, Homework 3
Will Buchta
2/26/2025

Problem 1

a) See train_gan.py and GAN.py for my implementation. I chose to plot 25 faces rather than 20 for a nice 5x5 collage. The generator architecture consists of a linear layer that up-samples to 128x6x6 after reshaping, followed by two transposed convolution layers to further up-sample to 24x24. The discriminator is a simple convolutional network that finishes with a linear layer down to a single likelihood prediction. I trained the model for 100 epochs and achieved very reasonable results. However, there is a rather curious mode collapse; all of the faces are "smiley", and most have some form of severe defect. See Figure 1:

Figure 1: Generated GAN faces after 100 epochs, latent z=100

b) I swapped out the generator architecture for that of the VAE's decoder from Homework 2 and loaded the model weights trained from there to act as a starting point. Ideally, this would help the GAN create more realistic face images because the VAE decoder starts from a learned P(X|z), and the results would be better than just the VAE because the GAN would remove some of the blur. In my case, I had trouble getting it to train. The discriminator was too powerful right away and had a loss that was between 1000 and 10,000x lower than the generator. To try and fix this, I added label smoothing to the discriminator, but to little more success. Figure 2 and 3 show my results from the training run. As you can see, they are essentially the same faces.

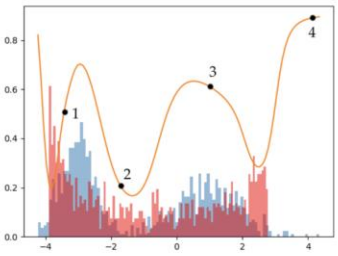
Figure 2: VAE Decoder after GAN training

Figure 3: VAE Decoder from Homework 2 with same latent Z as Figure 2

Problem 2

(a) **Explain D's Outputs [4 pts]**:

Suppose we train a GAN to generate 1-d data to match a given data distribution $P_{\text{data}}(x)$. The figure below shows two histograms (red and blue) as well as an orange curve. The blue histogram is the true data distribution $P_{\text{data}}(x)$. The red histogram is the generated data distribution $P(G(z))$ of a trained GAN generator G whose input $z \sim \mathcal{U}(0, 1)$. The orange curve shows the output $D(G(z))$ of the trained discriminator D.



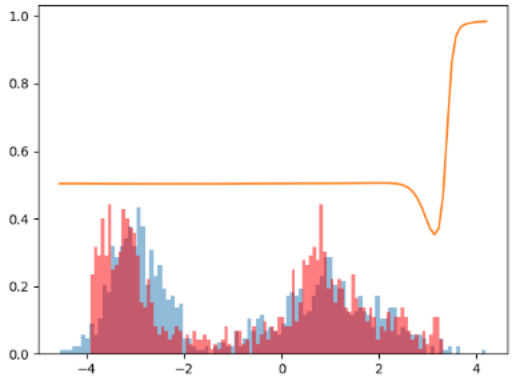
For each of the four labeled points $1, 2, 3, 4$ in the figure, explain in qualitative terms why the corresponding outputs of the discriminator $D(x_1), D(x_2), D(x_3), D(x_4)$ equal their respective values 0.5, 0.2, 0.6, and 0.9.

When the orange curve is near 1, that means the discriminator believes the data it has been fed is real. Based off that, the following can be deduced:

1. The true data distribution and the generated data distribution have the same probability at this point, meaning that the discriminator will not be able to tell them a part and is essentially flipping a coin to determine authenticity. Therefore, the output is around 0.5 or 50%.
2. At point 2, there is a much higher concentration of generated images than real images, so the discriminator predicts a low chance that the data is real because it is unlikely that real data exists at that point.
3. At point 3, there is a slightly higher frequency of real data than generated data, so the discriminator predicts a slightly more than 50% chance the data is real.
4. At point 4, there is no visible data from the generator, and only real data exists at this point, so the discriminator predicts a very high probability that the data is real. This is the converse of point 2.

(b) **What Will the GAN Learn [4 pts]:**

The figure below is similar to the previous exercise but represents the GAN at a later stage of its training. As shown in the histograms, G never produces an output less than about -3.9, nor does it produce an output bigger than about 3.4.



i. For which region ($x < -3.9$ or $x > 3.4$) is it *more likely* that G, through continued training, will learn to produce data similar to the true data distribution, and why?

ii. What is undesirable about the flatness of the output of D for $x < 2.5$?

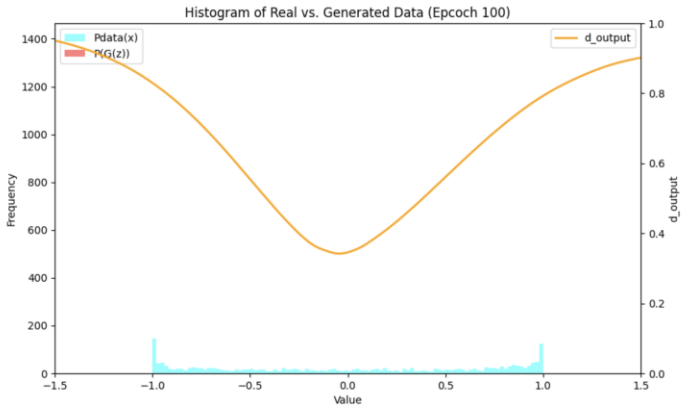
i) I want to say that the generator would learn to output data for x < -3.9 because there is a higher probability of real data there that it can learn from, but since the above example is given after training for some time, the discriminator already has some well-learned weights. Since D's output is essentially locked at 0.5, the generator won't learn much in that region. However, D is able to tell that data for x > 3.4 is fake, so G will be able to learn.

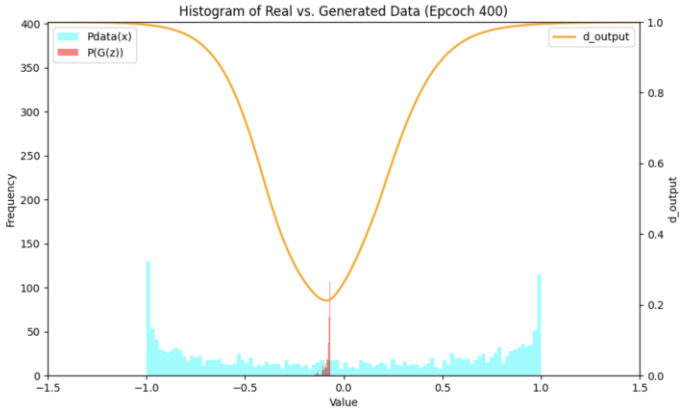
ii) The flat output of the discriminator provides little information to the loss function, so G will not learn very much because changes in its output will not affect the loss.

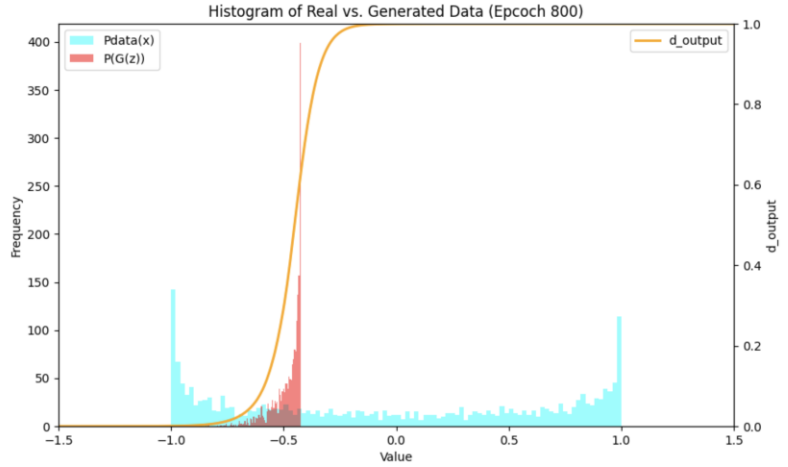
(c) **Show How a GAN Learns to Approximate $P_{\text{data}}(x)$ [20 pts]**

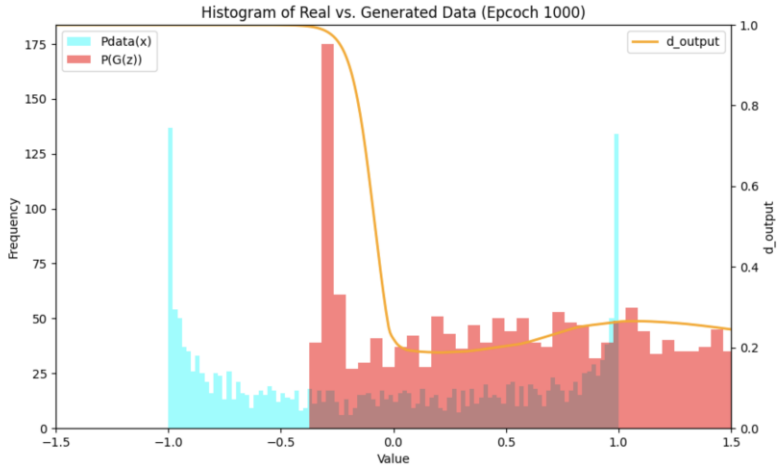
Similar in spirit to Homework 2, part 2, this is an open-ended exercise: Choose some true data distribution $P_{\text{data}}(x)$ (where, for simplicity and ease of visualization, $x \in \mathbb{R}$), choose a GAN architecture, and show how, over the course of GAN training, the generator can learn to approximate $P_{\text{data}}(x)$ more and more closely. In addition to your code, you should submit a sequence of 10 graphs (one for every few epochs of training) containing the same 2 histograms and orange curve as in the previous exercises, but tailored to your $P_{\text{data}}(x)$, $P(G(z))$, and D.

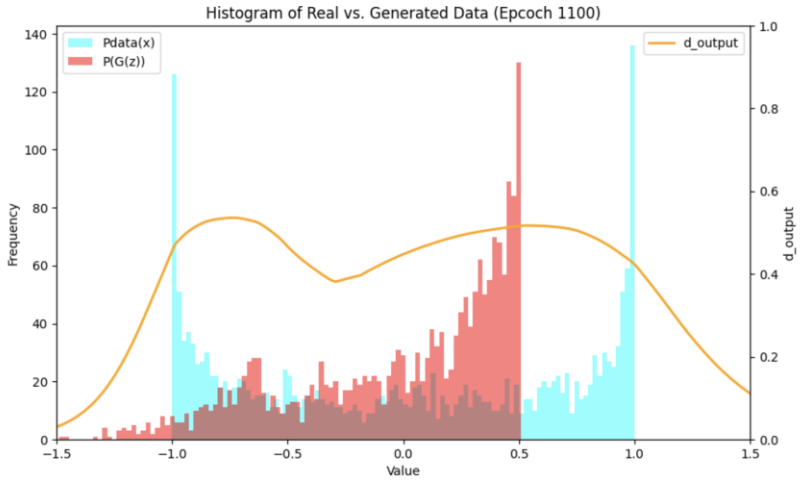
See visualize_gan.py. For my P(x), I chose to sample from sin(x) on the interval -pi to +pi. My generator is a simple linear NN with 1 input dimension, 2 hidden layers each having a dimension of 2048, and a single output dimension. The discriminator has the same architecture. The training sequence is quite interesting, and it's clear to see how G's distribution slowly learns P(x) and the discriminator's output gets closer and closer to a flat line at 0.5.

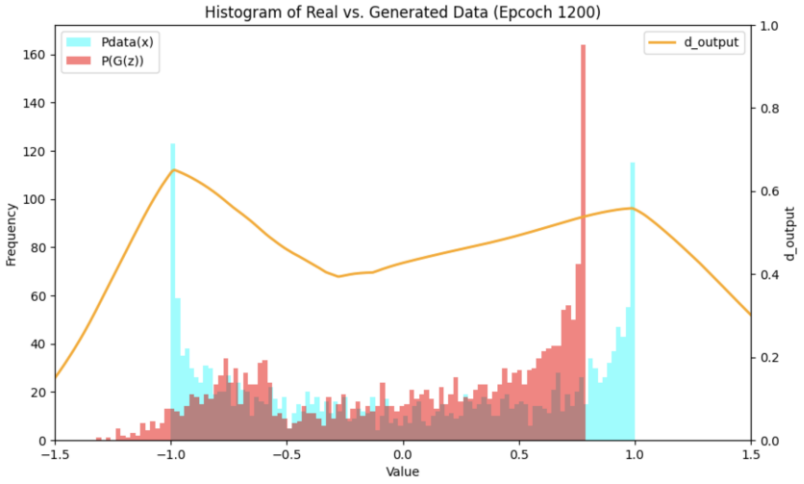


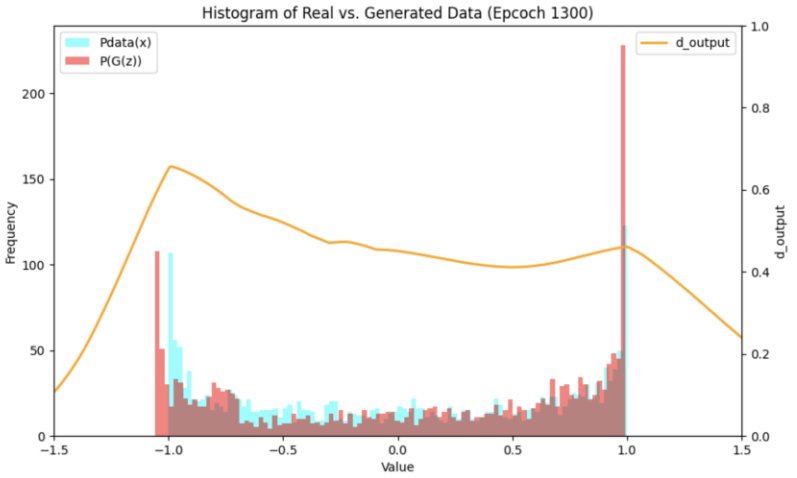


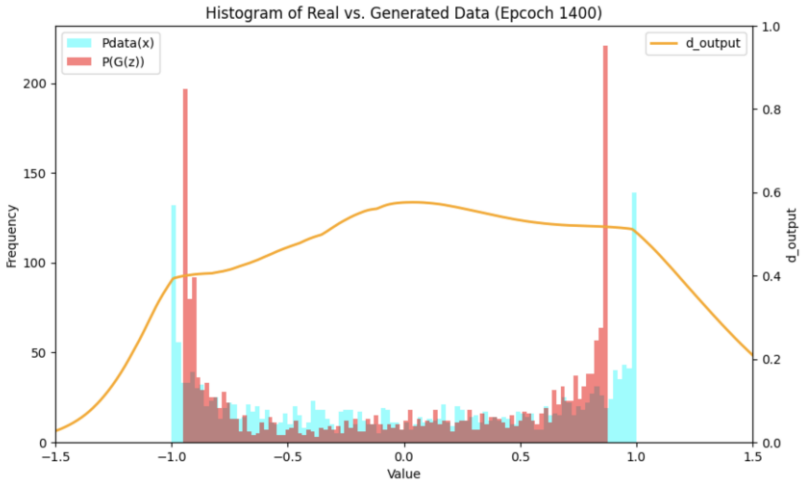

Histogram of Real vs. Generated Data (Epcoch 800)
Pdata(x)
P(G(z))
d_output
Frequency
d_output
Value

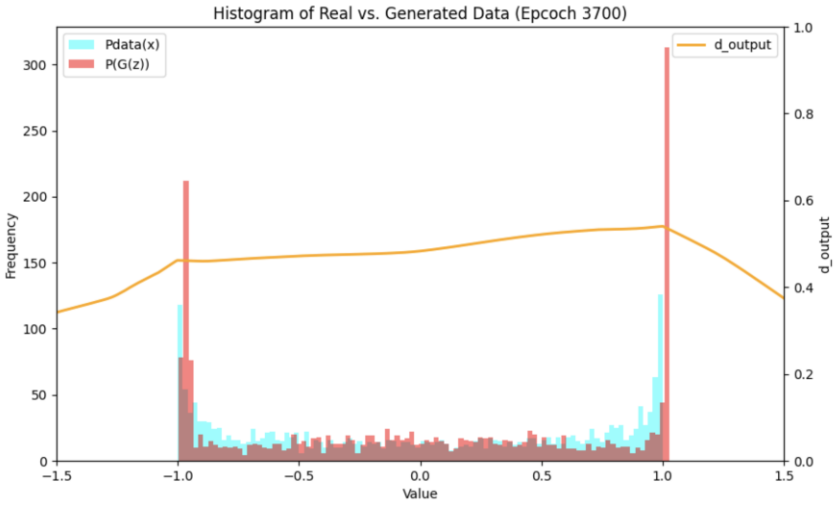

Histogram of Real vs. Generated Data (Epcoch 1000)
Pdata(x)
P(G(z))
d_output
Frequency
d_output
Value

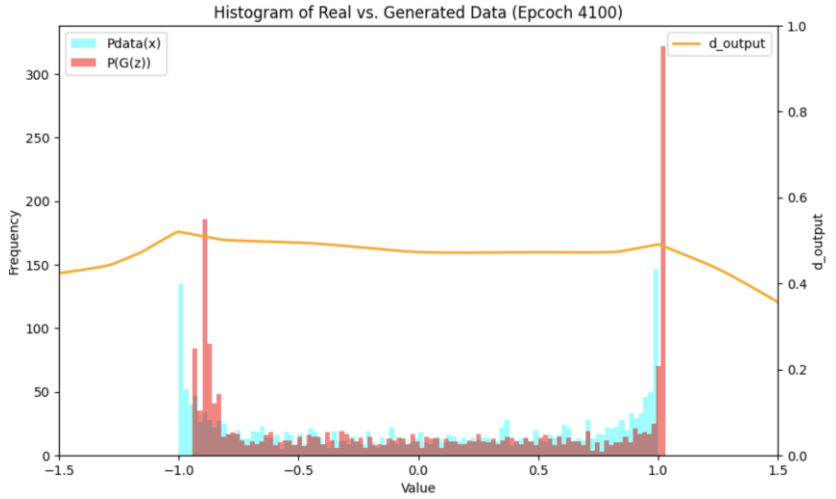

Histogram of Real vs. Generated Data (Epcoch 1100)
Pdata(x)
P(G(z))
d_output
Frequency
d_output
Value


Histogram of Real vs. Generated Data (Epcoch 1200)
Pdata(x)
P(G(z))
d_output
Frequency
d_output
Value


Histogram of Real vs. Generated Data (Epcoch 1300)
Pdata(x)
P(G(z))
d_output
Frequency
d_output
Value


Histogram of Real vs. Generated Data (Epcoch 1400)
Pdata(x)
P(G(z))
d_output
Frequency
d_output
Value


Histogram of Real vs. Generated Data (Epcoch 3700)
Pdata(x)
P(G(z))
d_output
Frequency
d_output
Value


Histogram of Real vs. Generated Data (Epcoch 4100)
Pdata(x)
P(G(z))
d_output
Frequency
d_output
Value