Deep Learning Fall 2022 Midterm Project
Lucia Rhode and Nicole Joseph

Our goal for the midterm project was to reproduce figures 2A and 2E from the paper "Adversarial Autoencoders" by Makhzani et al.

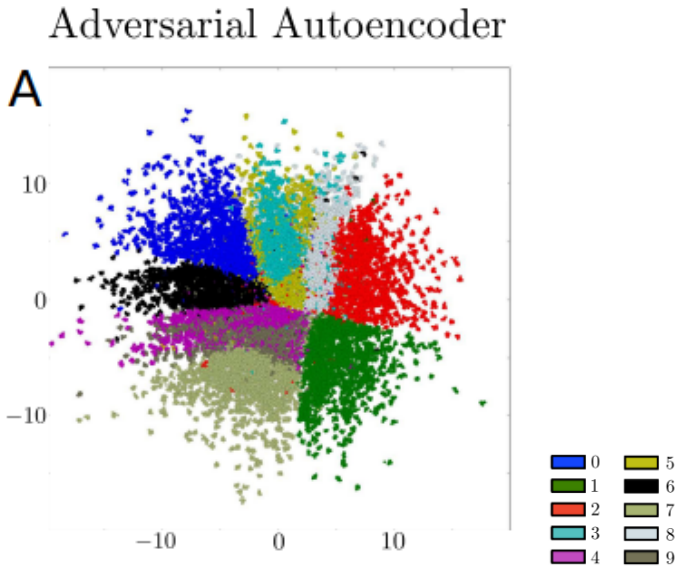

Figure 1. Figure 2A from "Adversarial Autoencoders" by Makhzani et al.

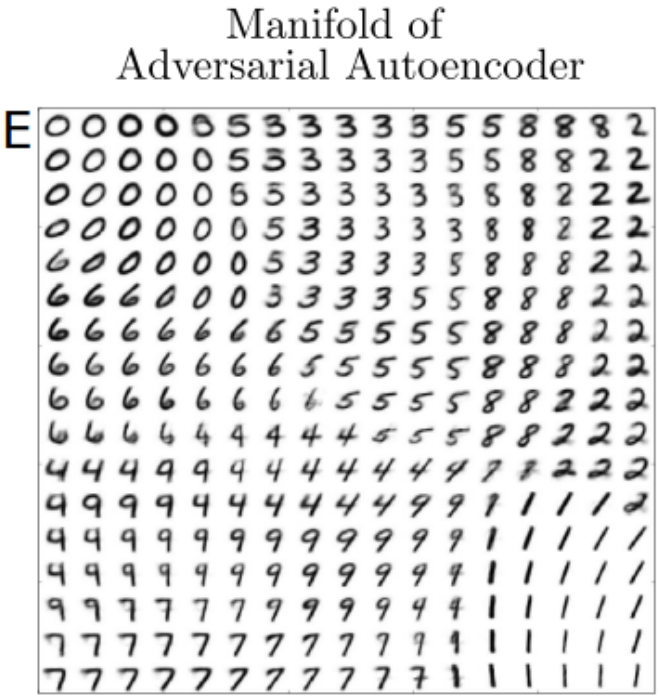

Figure 2. Figure 2E from "Adversarial Autoencoders" by Makhzani et al.

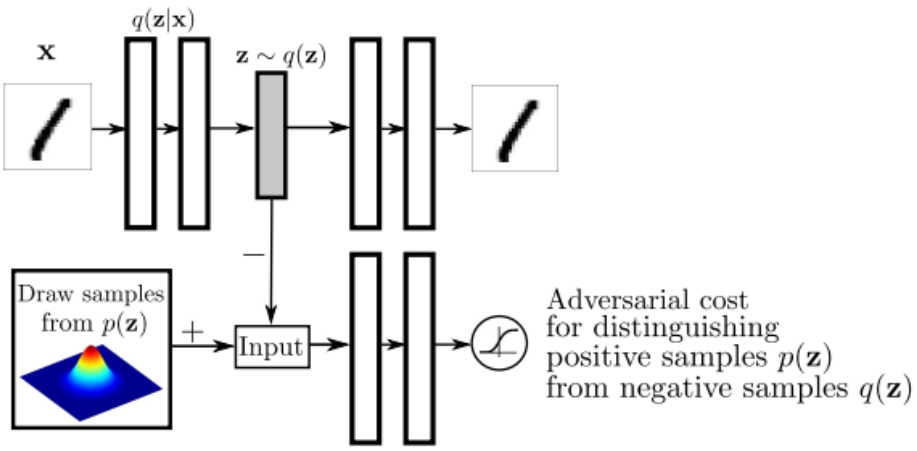

Figure 3. Architecture of the Proposed Adversarial Autoencoder (Makhzani et al., 2016)

The paper "Adversarial Autoencoders" proposes a model that transforms an autoencoder into a generative model by training it on an adversarial training criterion in addition to the traditional reconstruction error criterion of the autoencoder. Figure 2A of the paper shows the latent space of the hidden code z of the autoencoder fit to a 2-D Gaussian. Figure 2E of the paper shows the images generated by uniformly sampling from the 2-D Gaussian-fit latent space of the autoencoder.

To implement this architecture, we created an adversarial autoencoder model using three basic model blocks: an encoder, decoder, and discriminator. The encoder takes in the 28x28x1 images and compresses the input into the 2-D Gaussian latent space. The decoder takes the output of the encoder and attempts to reconstruct the 28x28x1 images. These two building blocks are the autoencoder component of the adversarial autoencoder. The discriminator also takes in real and fake input and learns to discriminate between them. The fake data is sampled from the encoder's latent space q(z) and the real data is sampled from a prior distribution p(z). This is the adversarial part of the network and it guides q(z) to match p(z). The adversarial autoencoder model was then built to minimize two losses: The loss of the autoencoder (the reconstruction error), which consists of the encoder and the decoder, and the loss of the discriminator—such that the discriminator tries to tell apart the real and fake data and the generative autoencoder tries to trick the discriminator.

To reproduce figure 2A, the MNIST test set was run through the encoder. The encoded test set was then plotted as a scatterplot. This visualizes the latent space of the autoencoder. As seen in figure 4, the numbers in the MNIST dataset create Gaussian blobs in latent space and similar numbers are physically close to each other.

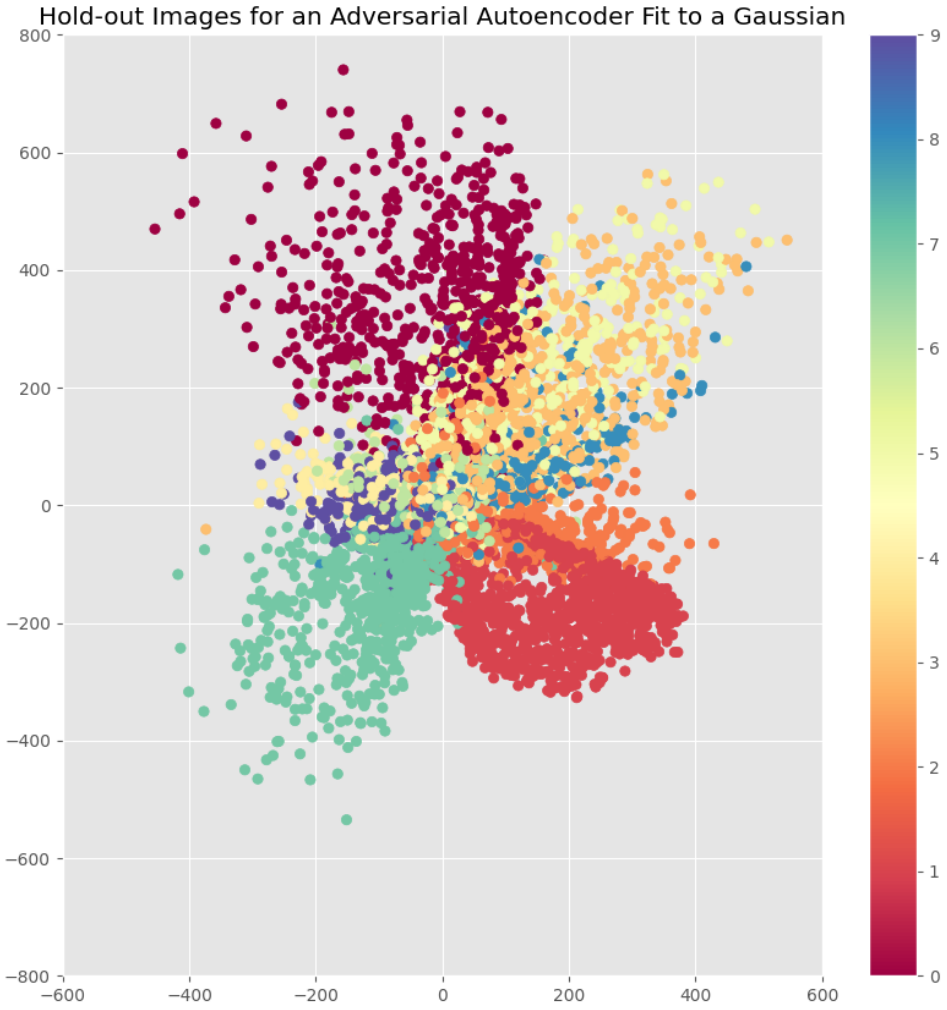

Figure 4. Reproduction of figure 2A

To reproduce figure 2E, images were generated by the decoder by uniformly sampling from the hidden code dimension z.

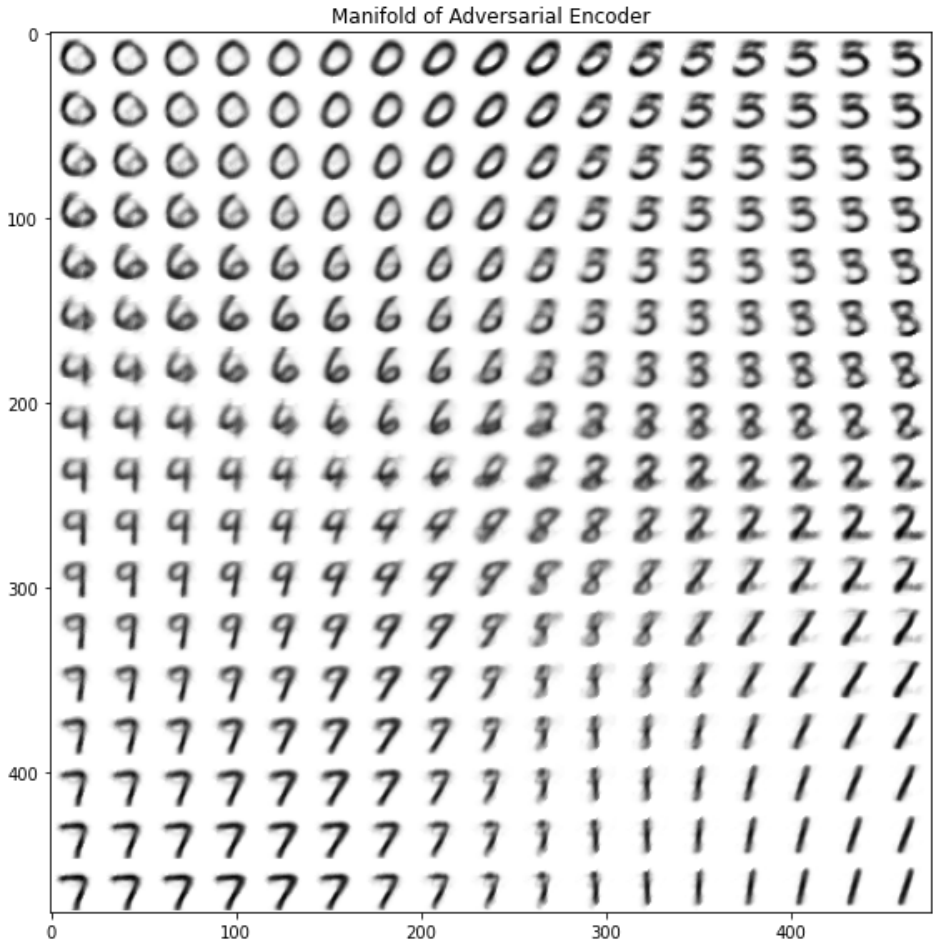

Figure 5. Reproduction of figure 2E

Ablation Experiments:

We were unsure how to determine if the encoder, in fact, fits a Guassian. Based on the diagram of the model architecture from the paper (figure 1), we understood that the prior distribution of the input samples, p(z), fed to the discriminator is Gaussian, and that the distribution of the hidden code/latent space of the autoencoder, q(z), is also Guassian. So, we coded the prior distribution of the discriminator to be Gaussian. For the encoder, we referenced (github link) which seemed to map the inputs to a Gaussian latent space. But based on the paper's diagram, we were unsure if this was necessary. So we decided to conduct ablation experiments and had the encoder map to an undefined 2D latent space. We wanted to see if the encoder's latent space could fit a 2D Gaussian distribution just from the regularization by the discriminator.

This resulted in figures 6 and 7. The distribution of the encoder's latent space still looks somewhat Gaussian and blobby.

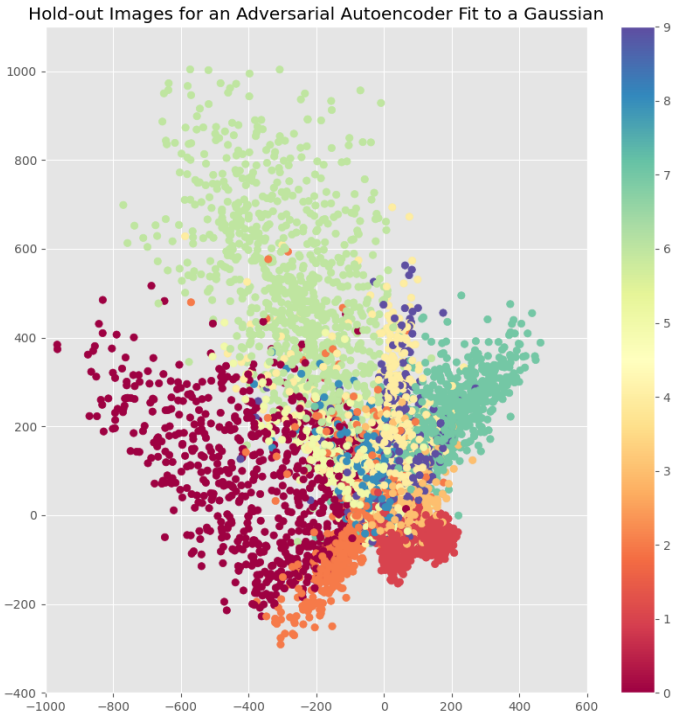

Figure 6. Ablation Experiment Reproduction of Figure 2A

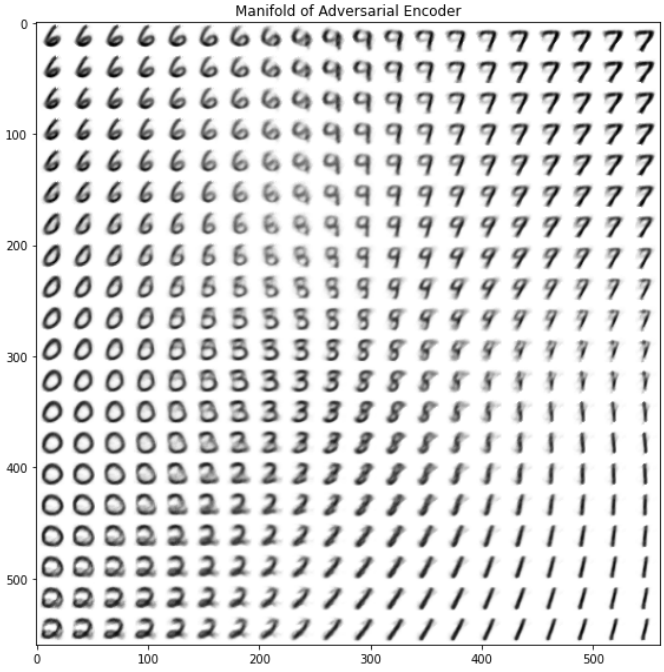

Figure 7. Ablation Experiment Reproduction of Figure 2E

References:

Makhzani, Alireza, et al. “Adversarial Autoencoders.” *International Conference on Learning Representations*, 2016, http://arxiv.org/abs/1511.05644.