Image Generation with GAN

Michio Sun

2023/12/1

1 Introduction

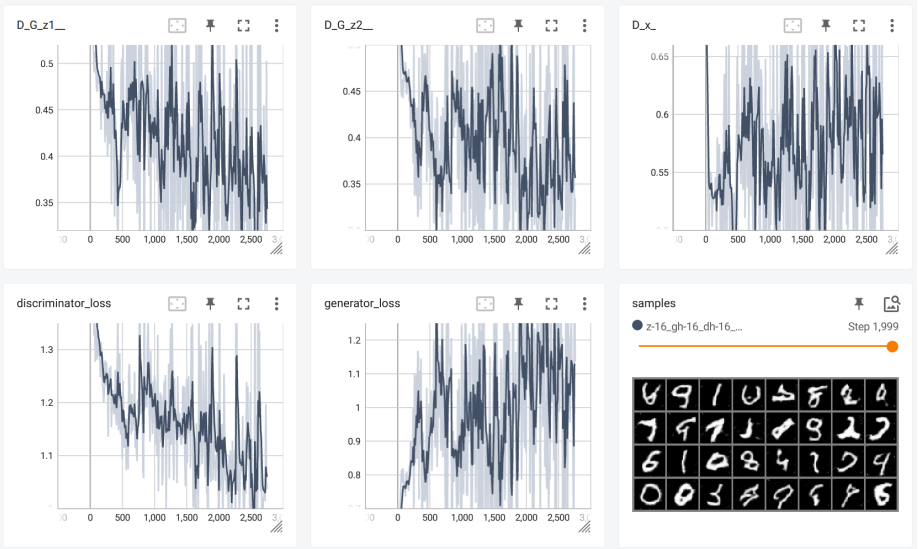
This assignment utilizes DCGAN, a variant of the generative adversarial network that utilizes convolutional layers in the discriminator and transposed convolutional layers in the generator, to classify and generate images. Specifically, this assignment uses the MNIST dataset.

2 Training Results

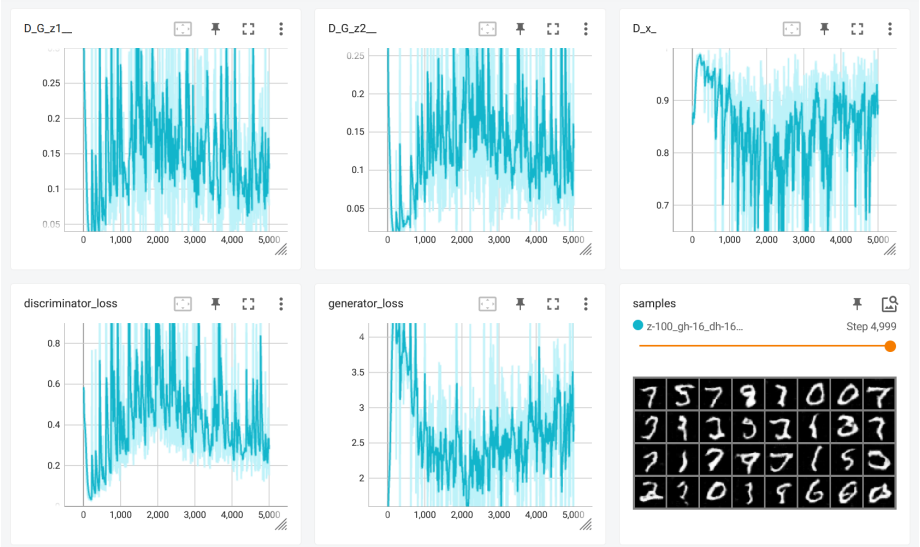
Six networks of varying hyper-parameters were trained. Below are the hyper-parameters and their results.

In the images below, z is the latent dimension, gh is the generator hidden size, and dh is the discriminator hidden size. The batch size is 64 and steps is 5000.

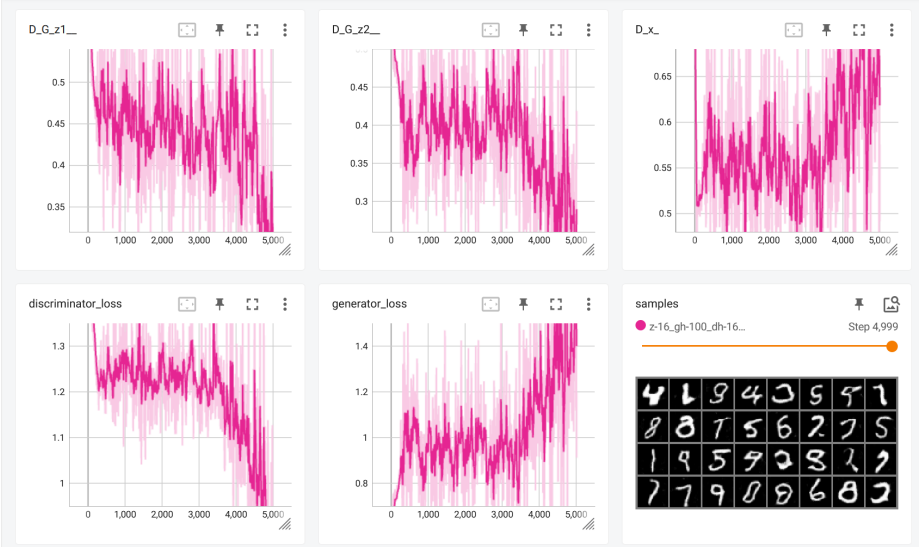
$z = 16, gh = 16, dh = 16$



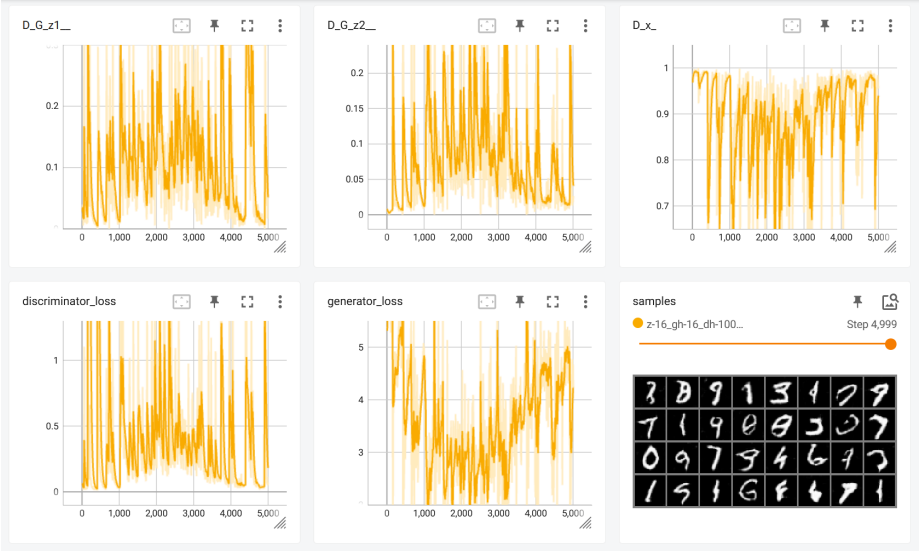
$z = 100, gh = 16, dh = 16$



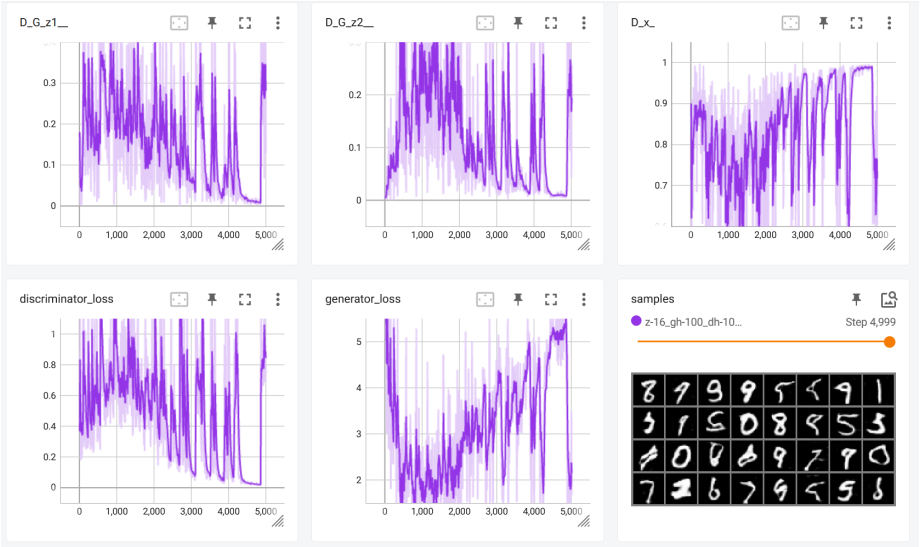
$z = 16, gh = 100, dh = 16$



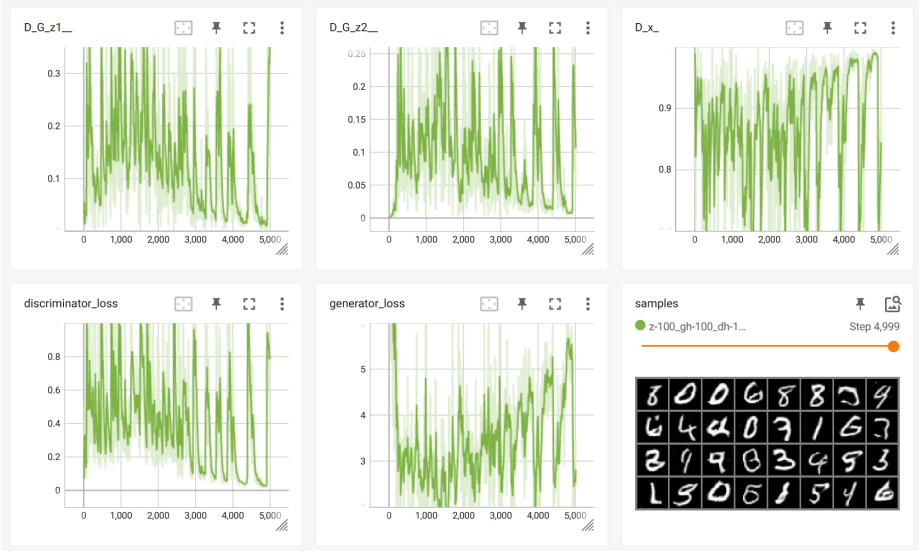
$z = 16, gh = 16, dh = 100$



$z = 16, gh = 100, dh = 100$



$z = 100, gh = 100, dh = 100$



2.1 FID Scores

Below are the FID scores for each model.

z	gh_dim	dh_dim	FID
16	16	16	112.3
16	100	16	44.8
16	16	100	44.66
16	100	100	33.84
100	16	16	65.18
100	100	100	24.61

The model with both the largest latent dimension size and hidden dimension size performed the best.

3 Dimensions and Performance

3.1 Latent Dimension

The latent dimension size greatly affected the performance of the GAN. Holding all other parameters equal, increasing z from 16 to 100 reduced the FID score from 112.3 to 65.18. The latent space on its own does not have any meaning, but through training, the generator learns how to produce accurate images from different inputs in the latent space.

A smaller latent dimension limits the expressiveness and diversity of the generator, with less traits for the generator to consider due to the lack of ability to capture complex patterns, which is evident in the generated image results from the model with $z = 16$.

A latent dimension that is too large may result in overfitting, since the model may start to memorize patterns rather than learning general patterns, leading to poor generalization to new data. Additionally, a large latent dimension will take a longer time to train, and will use up more computational resources, including memory and processing units.

Due to the difficulty of training a GAN and the number of hyper-parameters needed for tuning, the successful results of past models, and previous empirical data, most models tend to agree on using a hyper-parameter of 100 for convenience.

3.2 Hidden Dimension

The hidden dimension of a GAN refers to the size of the hidden layers in the generator and discriminator networks. This parameter plays a significant role in determining the capacity, expressiveness, and performance of the GAN.

As seen in the FID score results, increasing the dimension size of either gh_dim or dh_dim from 16 to 100 led to a increase in performance, from 112.3 to 44.8 and 44.66 respectively. When both were set to 100, there was an even bigger increase in performance, to 33.84.

A larger hidden dimension size leads to better pattern recognition abilities and increased capacity, which results in better performance.

4 Convergence

4.1 Discriminator Optimal Solution Derivation

$D(x)$ represents the probability that x came from the data rather than p_g. To optimize this, we need to look at the loss equation.
For G fixed, the optimal discriminator D^* is

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)}$$

where $p_r(x)$ and $p_g(x)$ are the reference distribution and the generator's distribution respectively. Below is the proof.
If

$$y = a \log y + b \log 1 - y$$

one can find the values of a and b when the derivative of y is 0 to obtain the optimal point, y^*.

$$\begin{aligned} y' &= \frac{a}{y} - \frac{b}{1-y} \\ \frac{a}{y^*} &= \frac{b}{1-y^*} \\ \frac{1-y^*}{y^*} &= \frac{b}{a} \\ \frac{1}{y^*} &= \frac{a+b}{a} \\ y^* &= \frac{a}{a+b} \end{aligned}$$

Optimizing for $D(x) = p_r(x) \log D(x) + p_g(x) \log 1 - D(x)$:

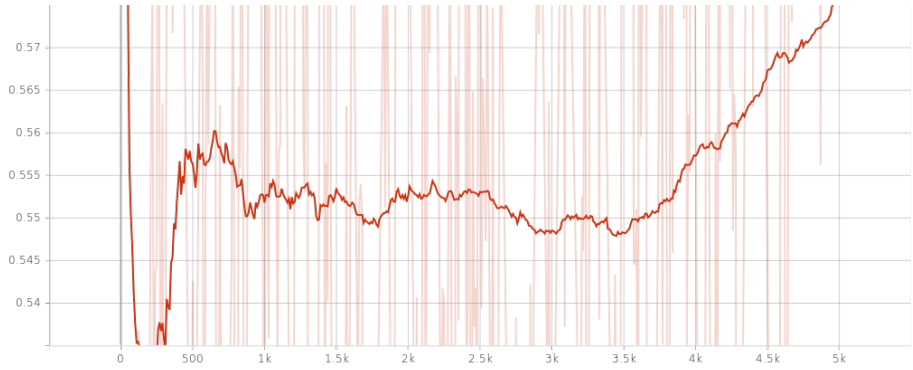
$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)} \tag{1}$$

This means that to get $D^*(G) = \frac{1}{2}$, the following has to be true:

$$p_r(x) = p_g(x) \tag{2}$$

4.2 Nash Equilibrium

As seen in the curves for $D\_x\_$, none of the models reached equilibrium. In fact, none of the models were approaching convergence. The closest model to 0.5 was the model with $z = 16, gh\_dim = 100, dh\_dim = 16$.



When training, this model has a much better generator in the beginning, but as training progresseed, the generator began to overfit as the discriminator became better, resulting in divergence.

This reflects the reality of training GANs: they are difficult to train with stability and often doesn't reach the desired convergence.
DCGAN is already an improvement on normal GAN, which removes max pooling, since it destroys spatial information, and repalces it with convolutional stride.

Some additional improvements may include new cost function penalties, new cost functions, New penalties may include those for mode collapse, which will be discussed in a later section. The discriminator can also be modified so that if the gradient vanishes or destabilizes, the training will fail. so that the emphasis is on the process of training.

5 Interpolation

A qualitative way to evaluate latent representations of a GAN is through interpolation. From two randomly generated latent input vectors, denoted z_1, z_2, evaluation is possible by observing the smooth transition between corresponding images.

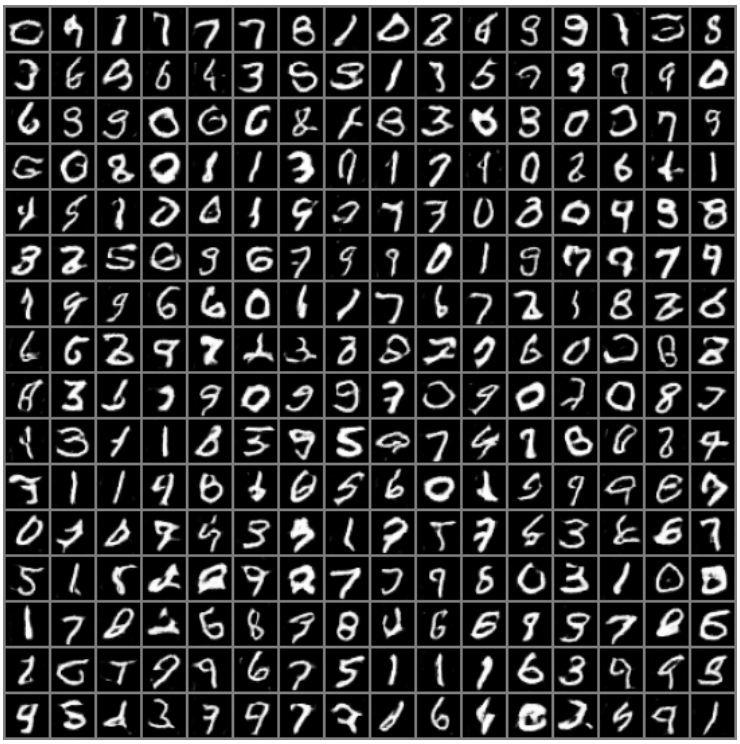
The images generated in between comes from $z_1 + \frac{i}{K}(z_2 - z_1)$ for $i \in$

$\{0, 1, ..., K\}$. In this evaluation, 5 pairs of z_1 and z_2 were used, with $K = 10$.

The images show that the network perceives the latent space relatively well, with a smooth transition in between. The generated images maintain a consistent style and structure throughout the process, with sufficient diversity.

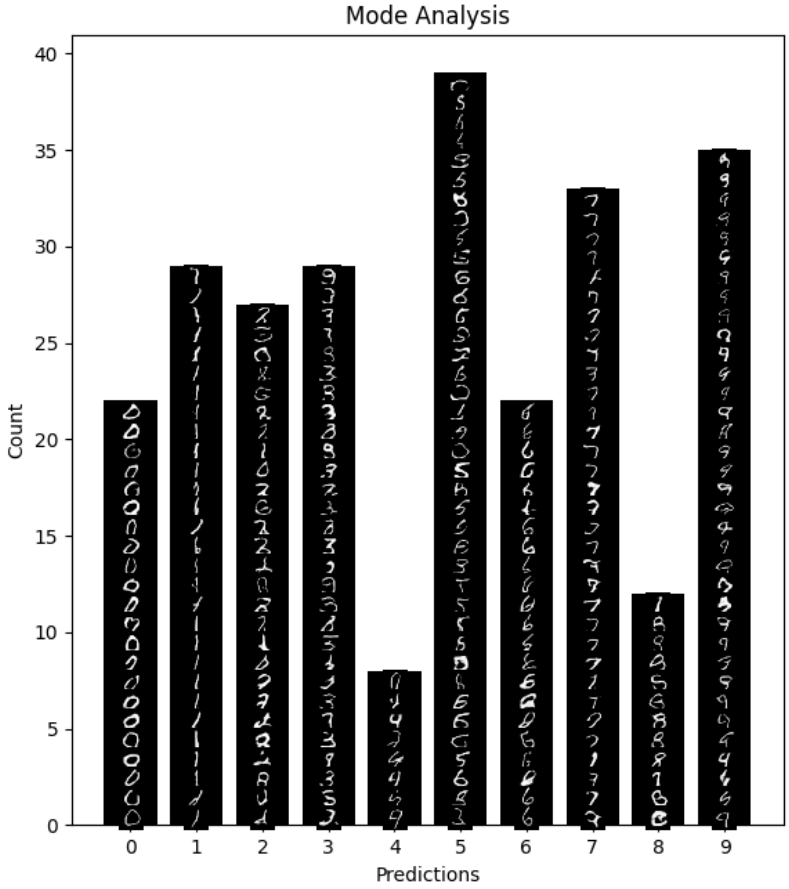
6 Mode Collapse (Bonus)

Mode collapse is a phenomenon where a GAN may only produce a limiting set of samples, ignoring or collapsing other modes in the distribution. Using the best model, $z = 100, gh = 100, dh = 100$, 256 random samples were generated.



These samples were then analyzed by a MLP model, trained in homework #1. The model used was a MLP with double hidden layers, ReLU activation, and hinge loss, which achieved 97% accuracy in validation.

A chart was also generated to show the distribution of generated numbers.



As seen in the chart above, there is a significant uneven distribution of numbers generated. Specifically, the numbers 4, 6, and 8 seem to be lower. This may be due to the fact that they are all composed by loops, and perhaps there may be a tendency to generate 9s when it comes to loops.

7 Conclusion

GANs may not be the state of the art model in 2023 due to training instability and mode collapse, but they are still an effective method

to generate data when tuned correctly and meticulously. Additionally, the structure itself, two adversarial networks in a zero-sum game, is extremely intuitive and even arguably beautiful.