2024

GAN

GENERATIVE ADVERSERIAL NETWORK

MUHAMMAD YAQOOB

Table of Contents

Table of Figures

1 Introduction

1.1 Objective

The purpose of this assignment is to develop an architecture for the GAN to generate natural looking images of the handwritten digit from the MNIST dataset. We also want to know the validation of the Generative Adversarial Network model in the aspect of image quality and confirm the stability of the both generator and the discriminator's losses.

1.2 Problem Domain

Generative modeling concept is critical in artificial intelligence across images synthesis and data augmentation. This is however has been revolutionized by Goodfellow et al's research paper at GANs which have made adversarial learning possible. This analysis identifies a practical requirement for synthesizing realistic handwritten digits that is important for handwriting recognition systems, tutorial tools, and document verification.

1.3 Method Rationale

GANs are suitable for this task because they leverage a competitive training approach, where a generator learns to create realistic images while a discriminator learns to distinguish between real and fake samples. This adversarial process aligns well with the objective of generating realistic data from noise distributions. The chosen architecture is designed to balance capacity and computational efficiency, ensuring convergence within 50 epochs.

2 Analysis

2.1 Dataset

I have used MNIST handwritten digit database. The MNIST database of handwritten digits, available over ATT Labs. It has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set for handwritten digits. The size of each image is 28×28.

Here is the detail of variables or classes in the dataset.

Feature	Class	Shape
Image	Image	(28, 28, 1)
label	ClassLabel	0,1,2, 3…,9

Here is the detail of each Feature Structure within the dataset:

```
FeaturesDict({
    'image': Image(shape=(28, 28, 1), dtype=uint8),
    'label': ClassLabel(shape=(), dtype=int64, num_classes=10),
})
```

Images are grayscale and these images were normalized to values between −1 and 1, and reshaped to include a single channel. This preprocessing will ensure the compatibility with the GAN architecture.

2.2 Exploratory Analysis

Sample Images from the MNIST dataset are:

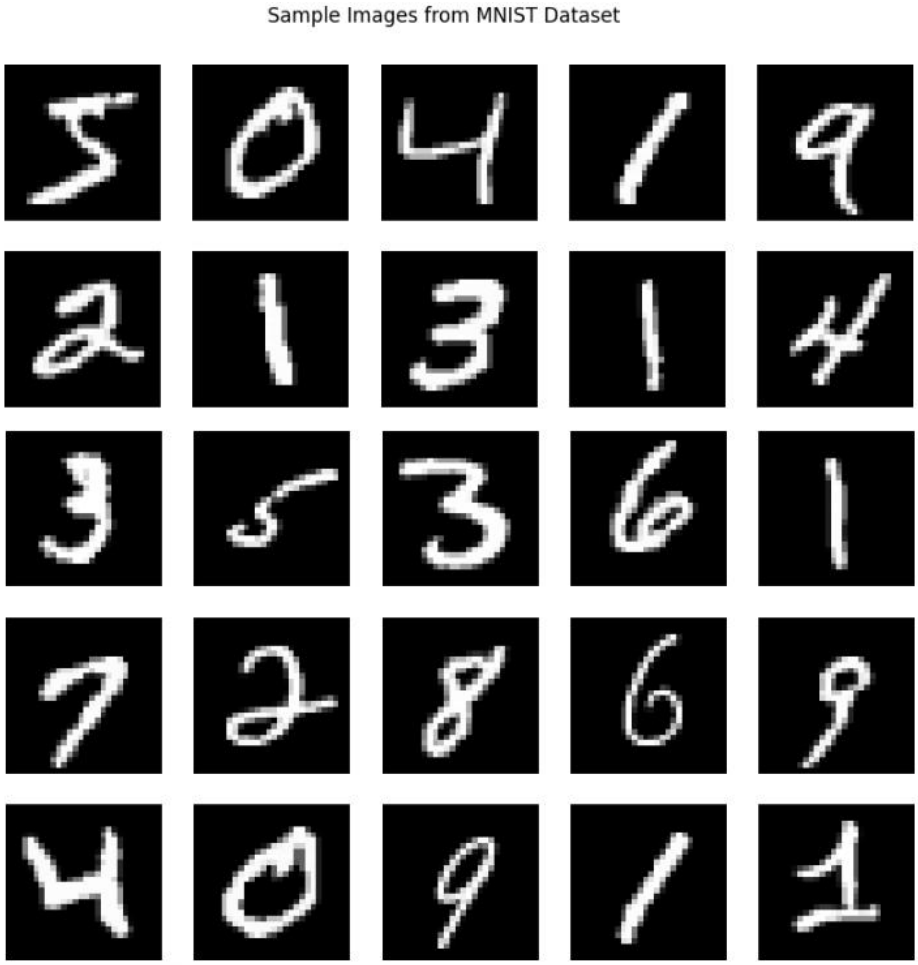

Figure 1: MNIST sample Images

Let's see the classes and their distribution through the Bar chart:

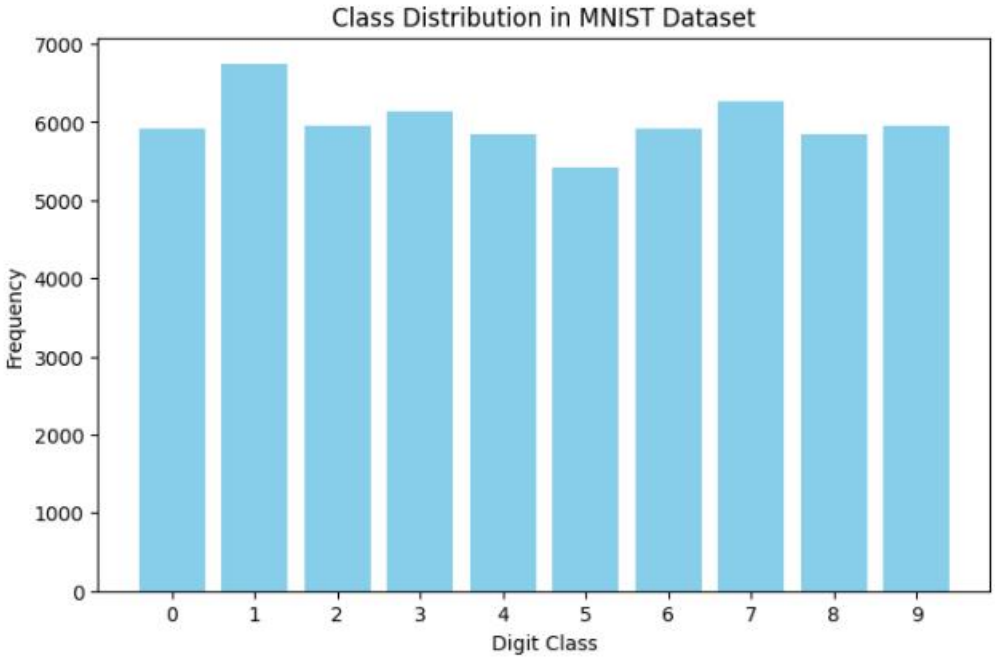

Figure 2: Class distribution in MNIST dataset

2.3 Preprocessing

Data preprocessing includes normalization of pixel values to the range [−1,1] and reshaping for model compatibility. I have also shuffled the dataset, so, I can get the best efficiency at the training time.

The pixel values were normalized to a range of -1 to 1 by subtracting 127.5 and dividing by 127.5. Normalization is important for stable training of GANs, as it ensures that the data distribution aligns well with the range of the generator's output (tanh activation).

Since the generator and discriminator models expect input data with a channel dimension, the images were expanded from a shape of (28, 28) to (28, 28, 1) using TensorFlow's tf.expand_dims() function. This additional dimension represents the grayscale channel.

2.4 Algorithm Intuition

In the Generative Adversarial Network, the word:

- **Generative** means we will generate some probability distribution which becomes close to the original data that we want to approximate.
- **Adversarial:** Adversarial mean conflict or opposition.

There will be two networks, which we call them as discriminator and generator. These two neural networks will be fighting against each other in order to learn the probability distribution function.

GAN are deep neural net architectures comprised of two neural networks competing one against the other. That's why we say them Adversarial.

GAN are networks that are trained in adversarial manner to generate data mimicking some distribution.

There are two classes of models in machine learning.

a) **Discriminative model:** It is the one that discriminates between two different classes of data:
   - Weather face is fake or not.
   - Output will be 0 if it is fake and 1 if it is not fake
   - It is a classification model

b) **Generative mode:** A generative model (G) to be trained on training data (X) sampled from some true distribution (D) is the one which given some standard random distribution(z) produces a distribution(D') which is close to D according to some closeness metric.

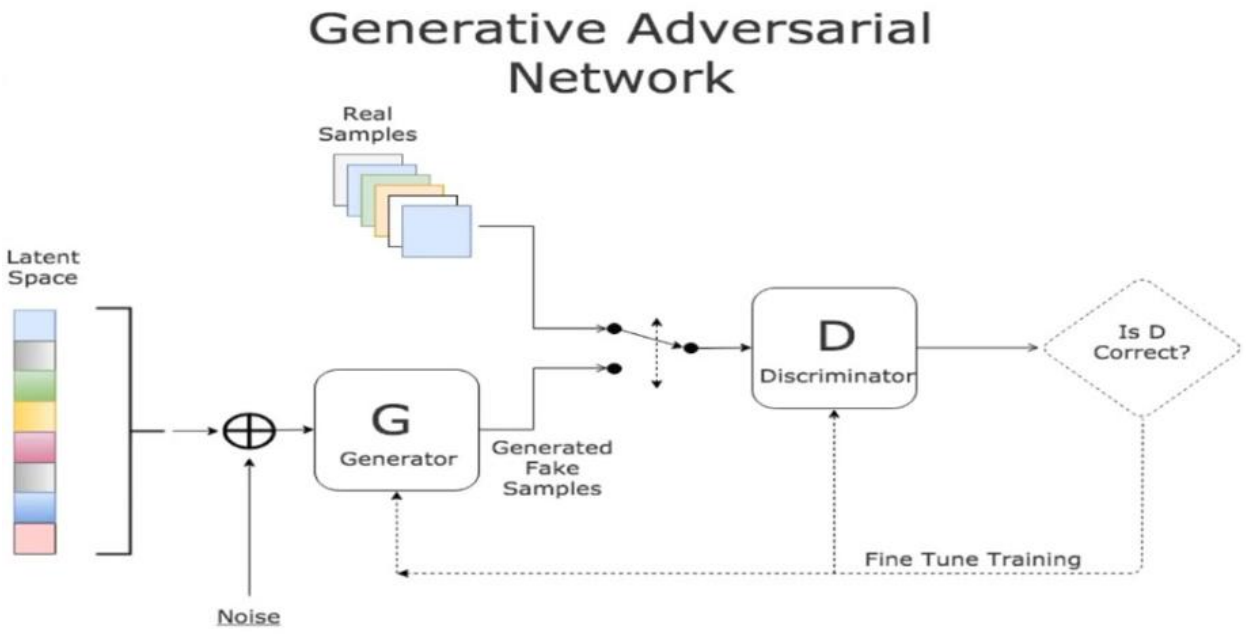

Figure 3: Generative Adversarial Network

GANs consist of two neural networks:

- Generator: Maps random noise vectors (z) to image-like outputs. Uses dense layers, batch normalization, and activation functions like LeakyReLU and Tanh.
- Discriminator: Classifies inputs as real or fake. Employs dense layers with LeakyReLU activation and a sigmoid output.

The adversarial loss functions ensure the generator improves its image quality while the discriminator refines its ability to detect fakes.

Let's see the internal working for the MNIST dataset for the GAN:

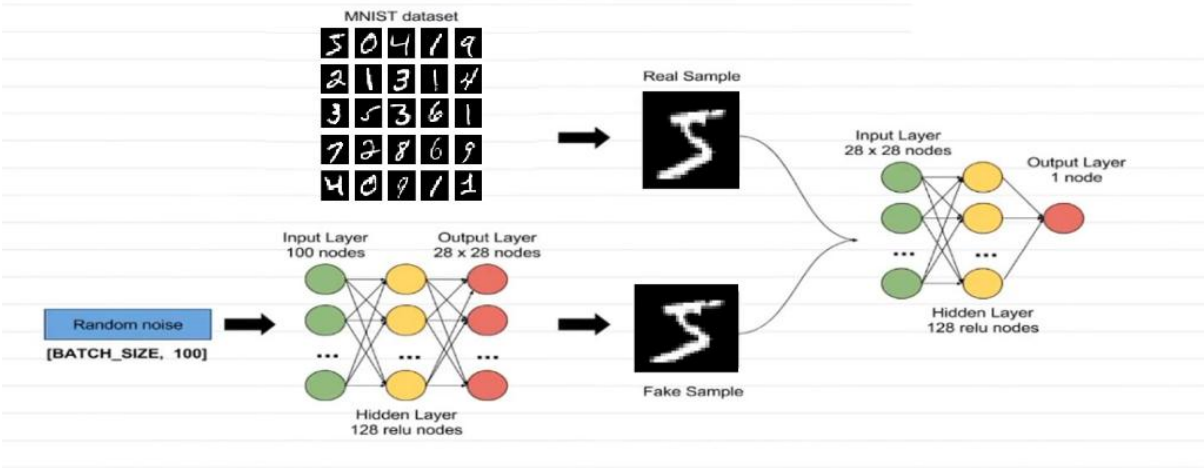

Figure 4: GAN for the MNIST dataset

2.5 Model Fitting

The GAN was trained using:

Generator Loss: Binary cross-entropy with labels 1 for fake outputs.

- Discriminator Loss: Sum of real and fake classification losses.
- Optimizers: Adam optimizers with a learning rate of 1×10^{-4}.

Loss values were monitored across 50 epochs, and hyperparameters were tuned for convergence stability.

Let's look at the equations of the loss functions of the Generator and Discriminator:

Discriminator's role is to distinguish between actual data and fake data. While Generator's role is to create data in such a way so that it can fool the discriminator. Now, let's see the equation associated with the binary cross entropy:

$$L(y', y) = ylogy' + (1 - y)\log(1 - y') \_\_\_\_\_(A)$$

For actual data image, we will have y = 1 and y' = D(x), So, the above equation will become:

$$L(D(x), 1) = \log(D(x)) \_\_\_\_\_(B)$$

For data from generator we will have, y = 0 and y' = D(G(z))

$$L(D(G(z)), 0) = \log\left(1 - D(G(z))\right) \_\_\_\_\_(C)$$

Objective of the discriminator is to correctly classify fake vs the real dataset. For this We need to maximize equation B and C. So,

$$\max\{\quad \log(D(x) + log\left(1 - D(G(z))\right)\} \_\_\_\_\_(D)$$

To fool the discriminator generator needs to produce D(G(z)) = 1

$$\min\{\quad \log(D(x) + log\left(1 - D(G(z))\right)\} \_\_\_\_\_(E)$$

By minimizing we are forcing D(G(z)) = 1, Finally

$$\min(G)\max(D)\{\quad \log\left(D(x) + log\left(1 - D(G(z))\right)\right\} \_\_\_\_\_(F)$$

3 Results

3.1 Output

I have added the generated images by the generator. All the images at the epoch 50 are almost same as they were real. So, it can be can be concluded that the generator has successfully done its learning. Generator's role was to create data in such a way, so it can foll the discriminator. So, it has succussed in its task.

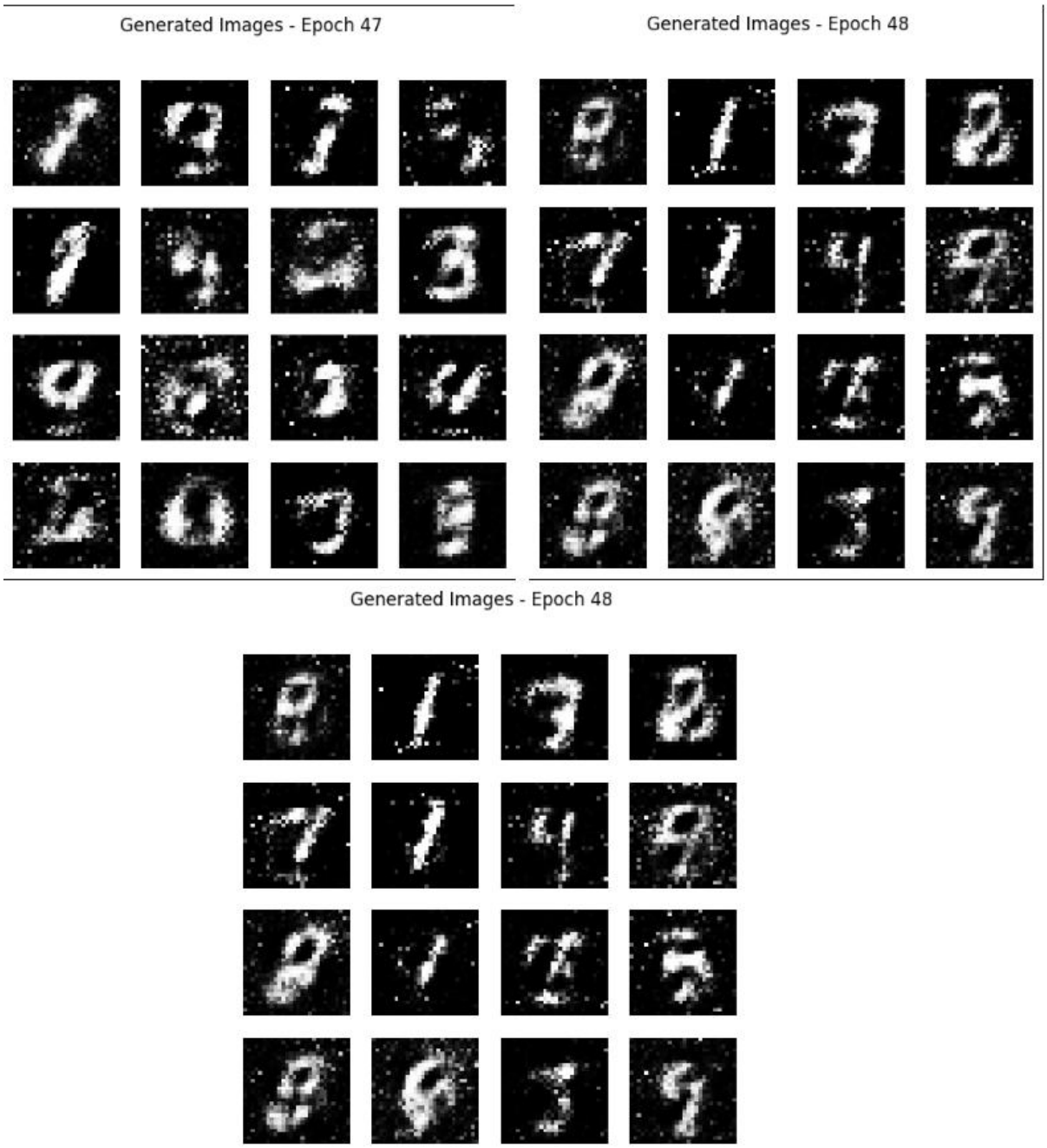

Figure 5: Generated Images by Generator

3.2 Model Properties

- Generator: It is a neural network having the Sequential model and it has three dense layers, batch normalization, and Tanh activation. And the Role of the generator was to classify the images weather they are fake or not.
- Discriminator: Second neural network was built with the Sequential model with three dense layers, LeakyReLU activation, and a sigmoid output.

So, we these two networks in adversarial manner. Also, GAN are neural networks that are trained in a adversarial manner to generated the same distribution as the actual images has.

Model weights were initialized randomly and updated through backpropagation. I have trained the Generator Adversarial Network up to the 50 epochs.

3.3 Evaluation

Quantitative metrics include:

- Generator Loss: It can be seen from the training loss of the Generator, which is in the blue line that it is about 0.7 and its around. It is achieved at the epoch 50. The generator learned it after the 50 successive approaches.
- Discriminator Loss: Discriminator loss at the epoch 50 is also about 0.6. So, it is very close to the Generator Loss.
- Visual Analysis: Generated images closely resembled real MNIST digits. It shows that the model's success in achieving its objective.
- Below is the training loss for the generator and discriminator loss.

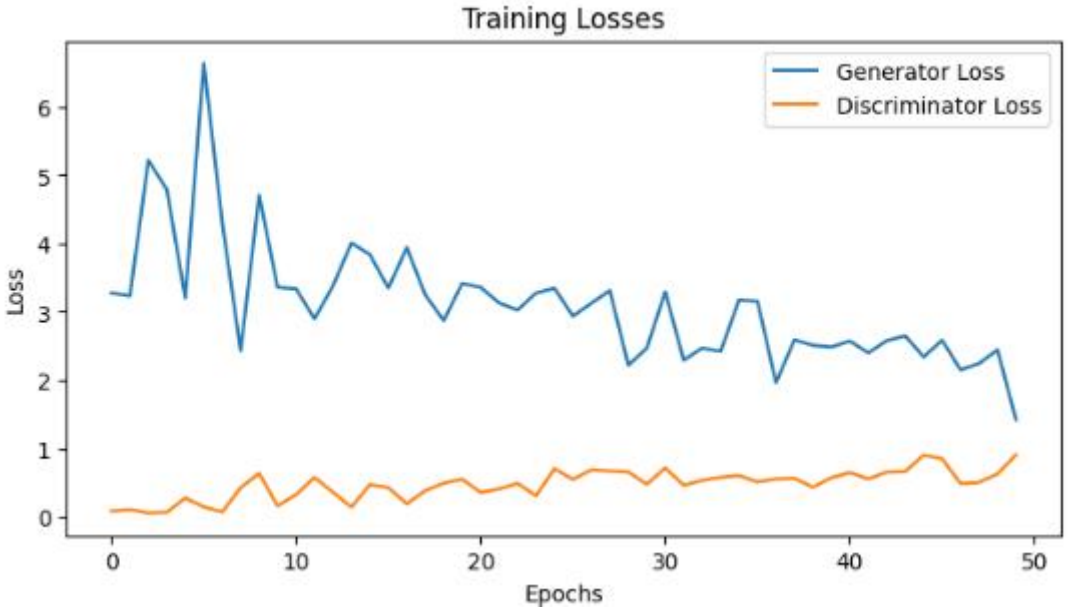

Figure 6: Generator and Discriminator Loss

Above loss plot shows that, how the generator learned during the epochs. I got the equilibrium state in the Training Losses. And it was required.

4 Conclusion

4.1 Summary

The GAN successfully generated realistic MNIST-like digits, with generator and discriminator losses converging by epoch 50. And I have achieved the equilibrium state between the Generator and the Discriminator.

4.2 Limitations

- There is some noise in the generated data.
- The generator occasionally produced low-quality or indistinct images.
- The quality of the generated images may be improved and distribution by the generator can be better, if I change the learning rate or batch size.
- It is always learning and getting better. Investing more time, I can increase its efficiency.
- It was the success for this dataset. But maybe it is complex for the other large datasets.
- It may fail where we need to produce pure distribution.

4.3 Improvement Areas

- Change in the learning rate and batch size may improve the quality of distribution.
- Also, Deep Convolutional GANs (DCGANs) will be best option for the higher-quality outputs.
- Change in the loss functions such as Wasserstein loss, will be helpful in the enhancement of the generator.
- We can extend the model to handle colored datasets or larger image resolutions.

5 References

- Goodfellow, I., et al. (2014). Generative Adversarial Networks.
- Lecun, Y., et al. (1998). MNIST Handwritten Digit Database.