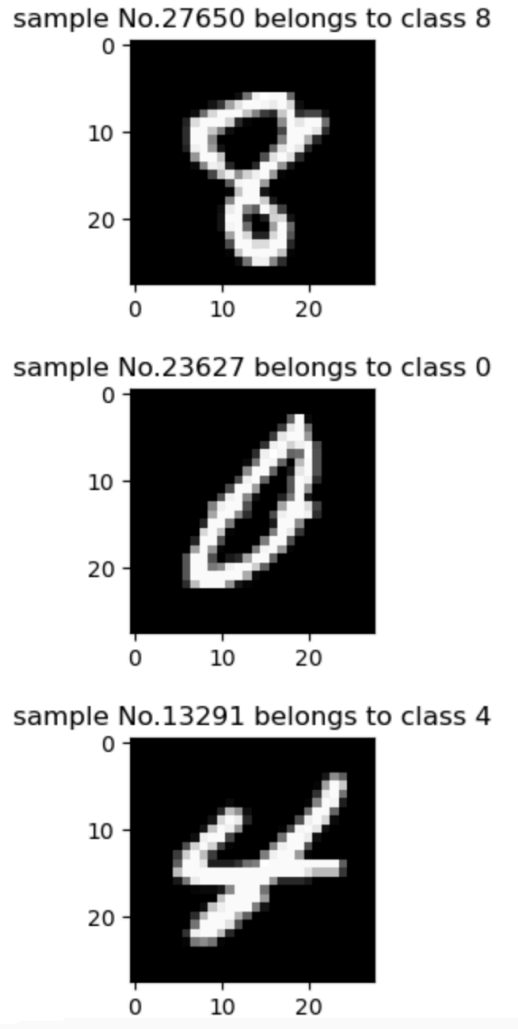
Deep Convolution GAN (DCGAN) for MNIST dataset

Deep Learning Final Project
May, 2024

(Aarti Veernala)

1. Objective

Train the GAN model to **generate images** as **close to the real handwritten images** as possible.



2. Dataset Used

Name : **MNIST** data.

This dataset that comprises of the original handwritten digits ranging from 0 to 9.

Data Attributes :

Number of samples = 60,000

Features = 28 x 28 pixel **black and white** images

2. DCGAN Model Description

- **Type of model** :

  Deep Convolutional (DC) layers applied to a Generative Adversarial Neural Network (GAN)

- **Model Details :**

  1. We use a GAN comprising of two deep convolutional neural networks

     A. Generator for upsampling the data from noise to a AI generated 28x28 image)

        - Activation function : tanh (to keep the values of the generated image between -1 and 1
        - Number of layers : 11 layers

     B. Discriminator (for downsampling the generated data to a single (1x1) output

        - Activation function : Sigmoid activation is applied to the discriminator via the sigmoid Binary Cross Entropy (BCE) Loss function
        - Number of layers : 9 layers

  2. We use a kernel size of 5x5 for the generator and discriminator

  3. Loss function : BCE (Binary Cross Entropy)

3. Data Pre-Processing

- Shape of Original Data : (70000, 28, 28)

  comprises of 60000 samples of 28 x 28 pixel BW images of handwritten digits.

- Pre-processing -
  - The original dataset is split into
    - Train data of shape (60000, 28, 28)
    - Test data of shape (10000, 28, 28)
  - Rescaled : The train and test data are rescaled to carrying float values between [-1,1]
  - Tensor : The rescaled train and test data are converted into a tensor slices that can be passed into the GAN

4. Project Analysis/Hyper-parameter Tuning

- Once the model is built, compiled and ready to go, we need to **tune our model** to ensure the **best possible outcome (BPO)**. The BPO is often measured in terms of :
  - Efficiency : runtime, computational power/cost etc
  - Accuracy : how well the model performs on test data. In our case, we will measure this in terms of how well the discriminator network identifies real test images (as real) and the generated images created by the generator network (as fake).

- The BPO is heavily influenced by the hyper-parameters of model such as
  - number of epochs
  - batch size
  - depth of the convolutional neural network (CNN)
  - choice of noise used by the generator network
  - etc

5. Hyper-parameter Tuning

We tune our model by varying the following hyper-parameters (one at a time, keeping all else fixed) and study the impact on the run time and the accuracy of our model.

- ⋆ Epochs
- ⋆ Batch Size
- ⋆ Depth of CNN or the number of layers

5a.

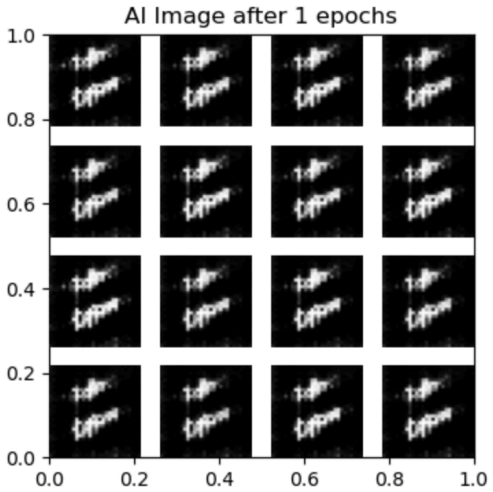
Epochs

Vs

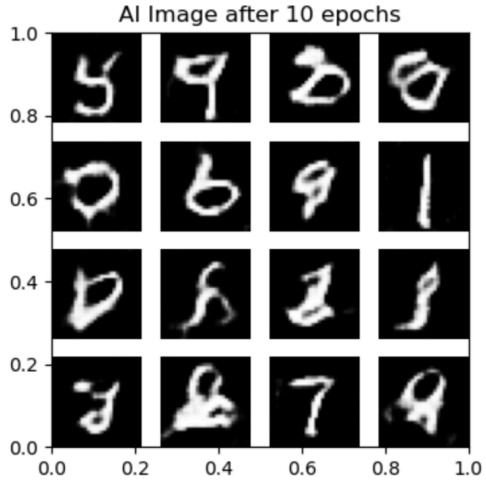
Visual Clarity

(Batch size = 256)

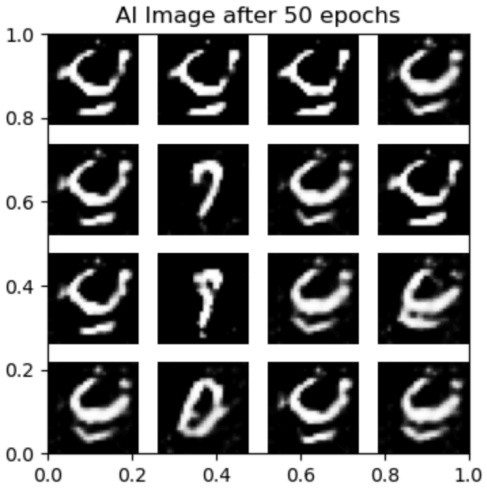
Epochs = 1



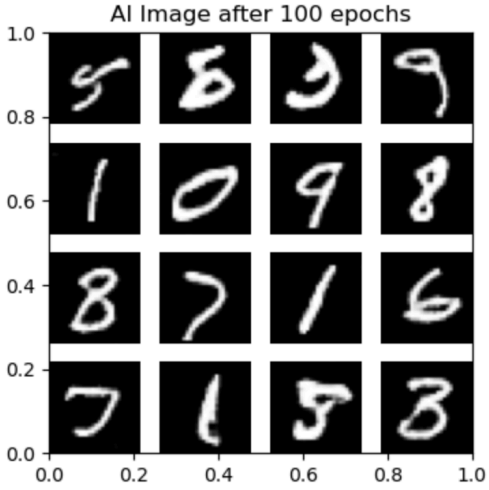
Epochs = 10



Epochs = 50



Epochs = 100



Result:

Image looks better and much closer to the original with larger number of epochs !

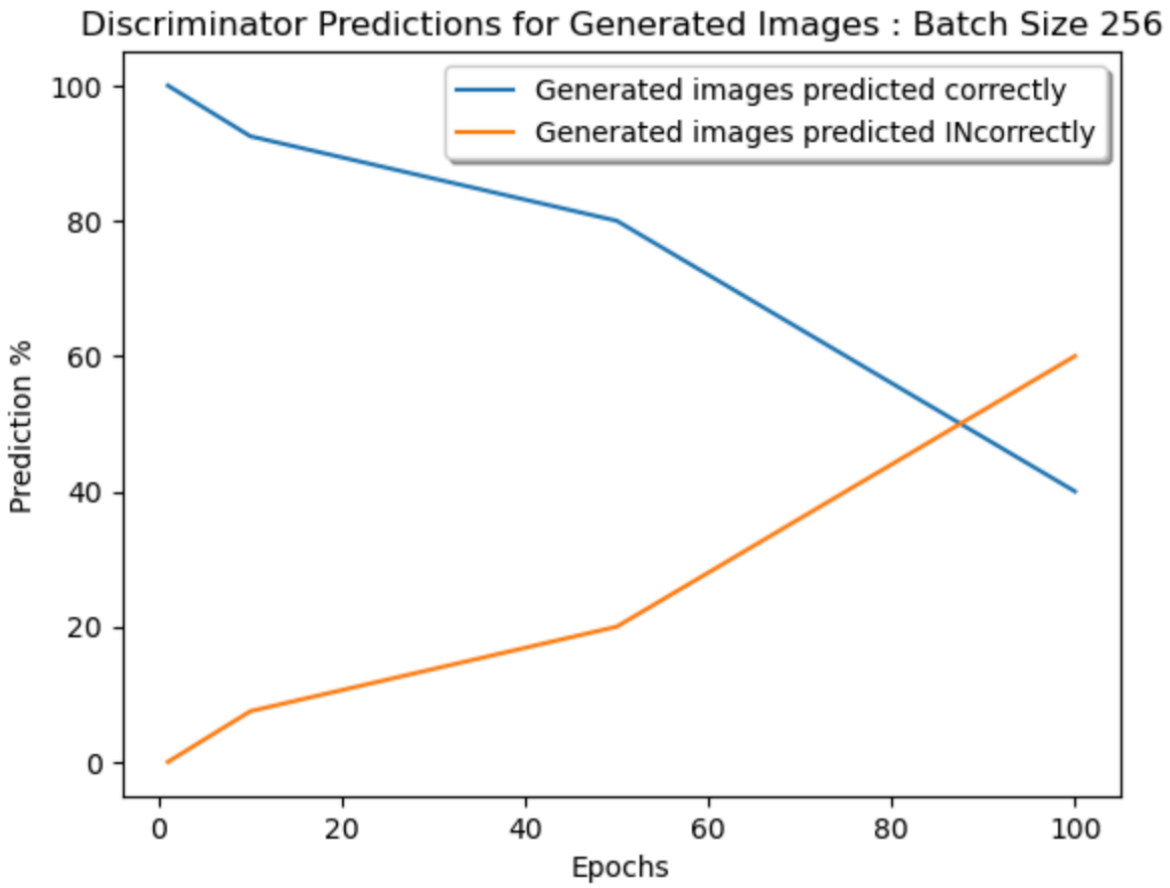
5b.

Epochs

Vs

Prediction Accuracy

This plot shows the discriminator network's ability to correctly identify generated images from real (test) images.



Result

As the number of epochs increase:

1. The discriminator's ability to predict generated images correctly drops (blue curve).
2. This shows that with increasing number of epochs, the model (generator) is learning better and more adept at generating images closer to the original.
3. The orange curve shows that with more learning the discriminator is getting 'tricked' more often by the generator.

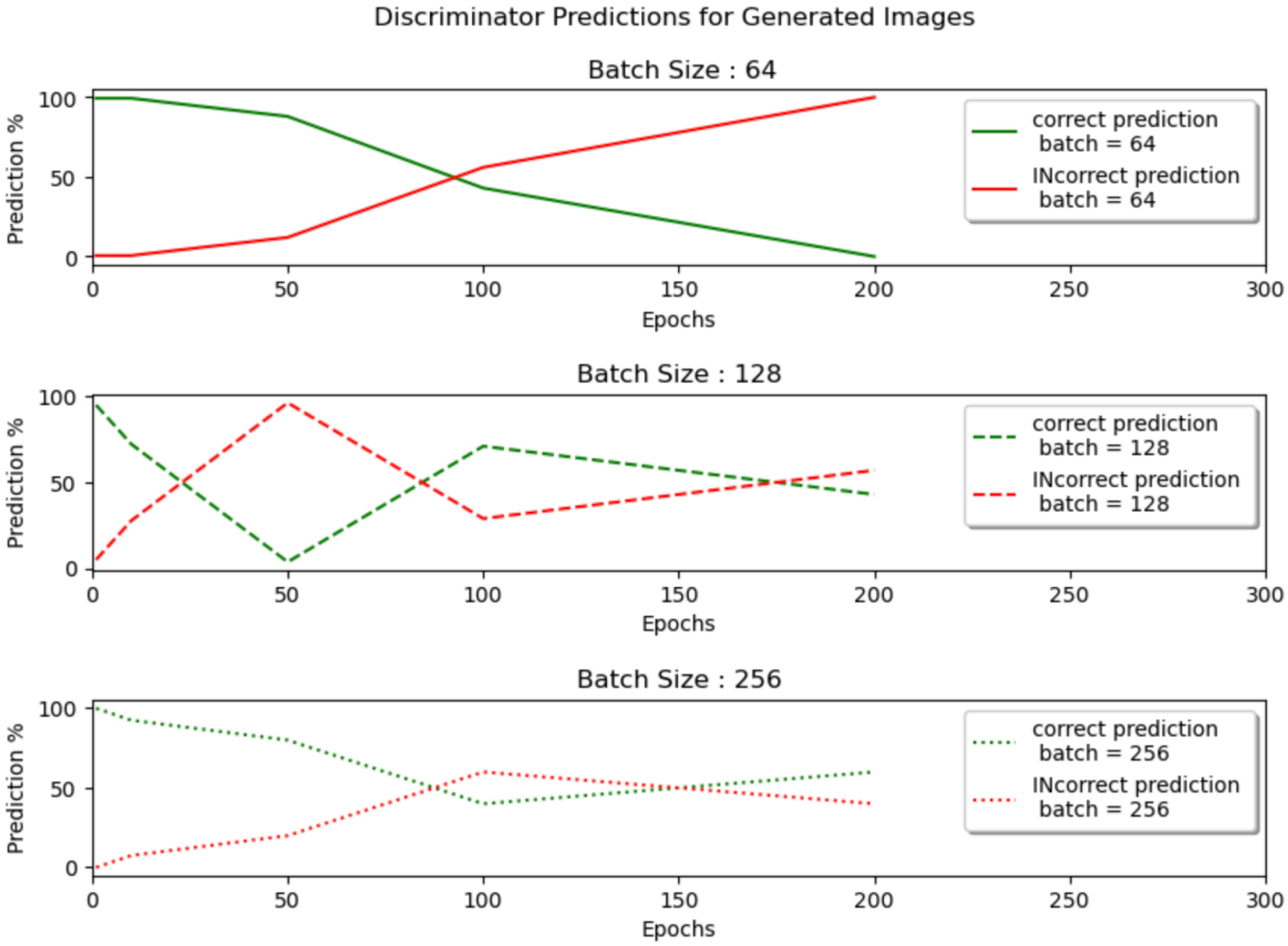
5c.

Batch Size

Vs

Prediction %

(**Over a range of epochs**)



Red : This curve represents the number of INcorrect predictions by the discriminator. This implies the **better performance of the generator**

Green : This curve represents the number of correct predictions by the discriminator. This implies the **poorer performance of the generator**

RESULT : With increasing epochs (upto 200), the performance of the discriminator and the generator both seem to get better. The higher incorrect predictions indicate that the generator is getting better with creating images close to the original image. However, the fact that the number of correct predictions is not consistently dropping indicates that the discriminator is learning to better identify real from generated images.

5d.

Choice of Noise

Vs

Prediction %

(**Fixed epoch, fixed batch size**)

Prediction % comparable

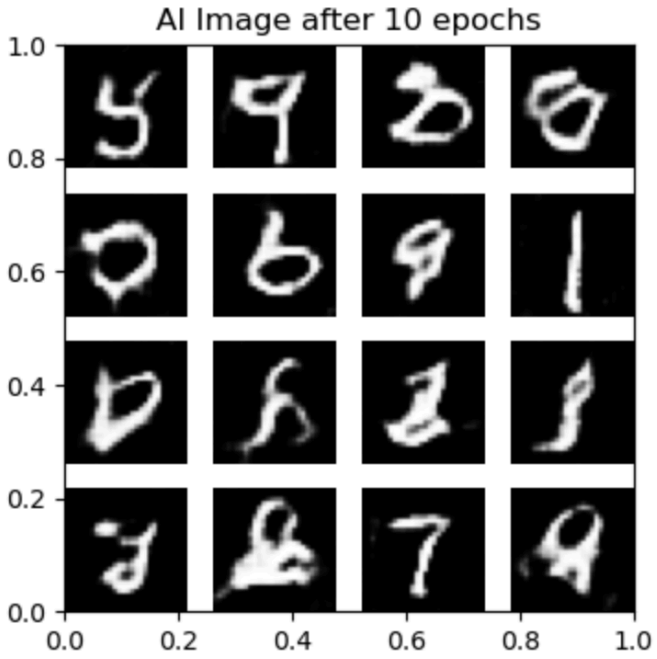
between noise sampled from

normal and Poisson distributions.

Discriminator prediction for generated images

Correct predictions = 92.5 %

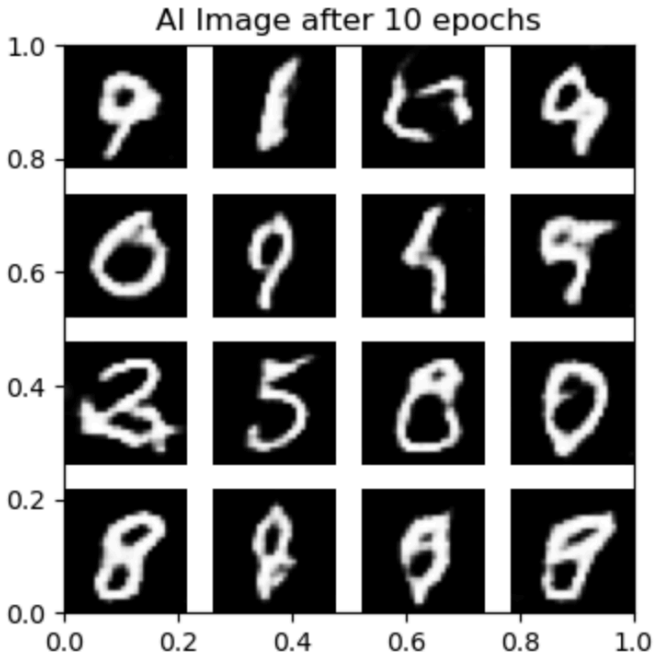
INcorrect predictions = 7.5 %



Randomn Normal

Correct predictions = 90 %

INcorrect predictions = 10 %



Randomn Poisson

6. Key findings

1. Epoch Variation:

   1.1. Image looks better and much closer to the original with larger number of epoch

   1.2. Discriminator outperforms generator for a small number of epochs.

   1.3. However, as the number of epochs increase, the generator's performance improves and the discriminators' ability to identify the generated from the real images drops.

   1.4. Further increasing the number of epochs, improves the performance of both the generator and the discriminator. The prediction % is seen to fluctuate between the correct and incorrect predictions.

2. Batch size Variation

   2.1. At batch size of 64 (epochs=200), the generator seems to have outperformed the discriminator, where the generated images to get MISidentified off as a real ones, nearly 100% of the times.

   2.2.The larger batch sizes (>64) did not seem to affect the performance (image quality, prediction) of the model in any significant manner.

   2.3. Reducing or increasing the batch size did not affect the total run time. All the batch size ran at the rate of around 3 minutes/ epoch.

3. Choice of noise

   No change observed in prediction ability or run time when the noise was sampled from a Poisson distribution instead of a normal distribution.

7. Best Choice

Best choice of model based on this study :

- Batch Size = 64
- Number of epochs : at least 200
- Normal distribution for noise sampling (easy and intuitive)

8. Issues/Scope

- **Issue/Scope**

  Run time limits upper end of epochs :

  The observed run time of 3 minutes per epoch limits the highest number of epoch we realistically train our model with. It takes 10 hours to run 200 epochs (all else fixed and constant).

- **Next Steps**

  1. Parallel processing! Total run time can be reduced if we were to switch over to GPUs. We can then train our model for much larger number of epochs.

  2. Study the impact of network depth (the number of hidden/deep layers) on run time, prediction ability etc.