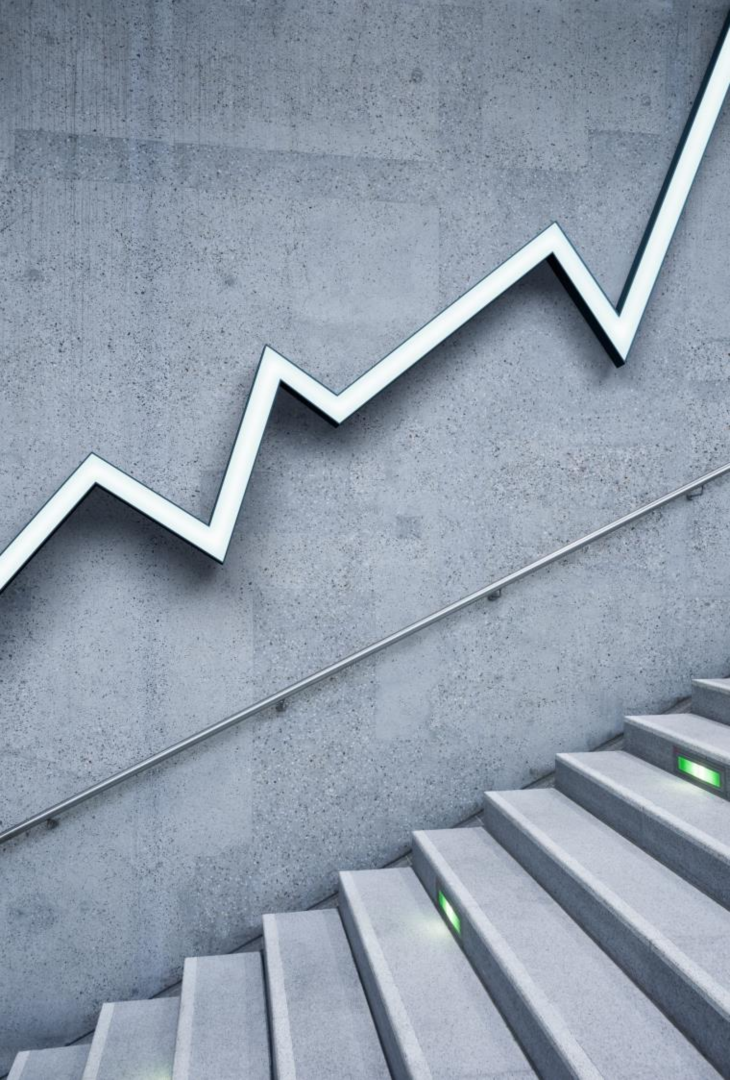


Image Generation Using GANs

Manish Ray

Presented By -
MANISH RAY(222123031)

Under the Guidance of

Prof Dr. Prasant Patil

Assistant professor

Mehta Family School of Data Science and Artificial Intelligence

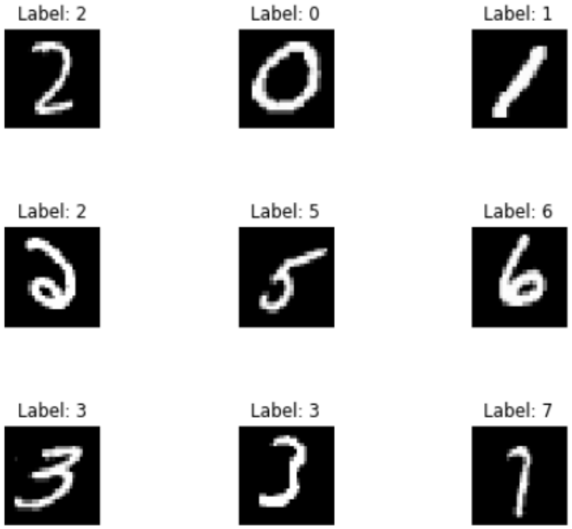
IIT Guwahati

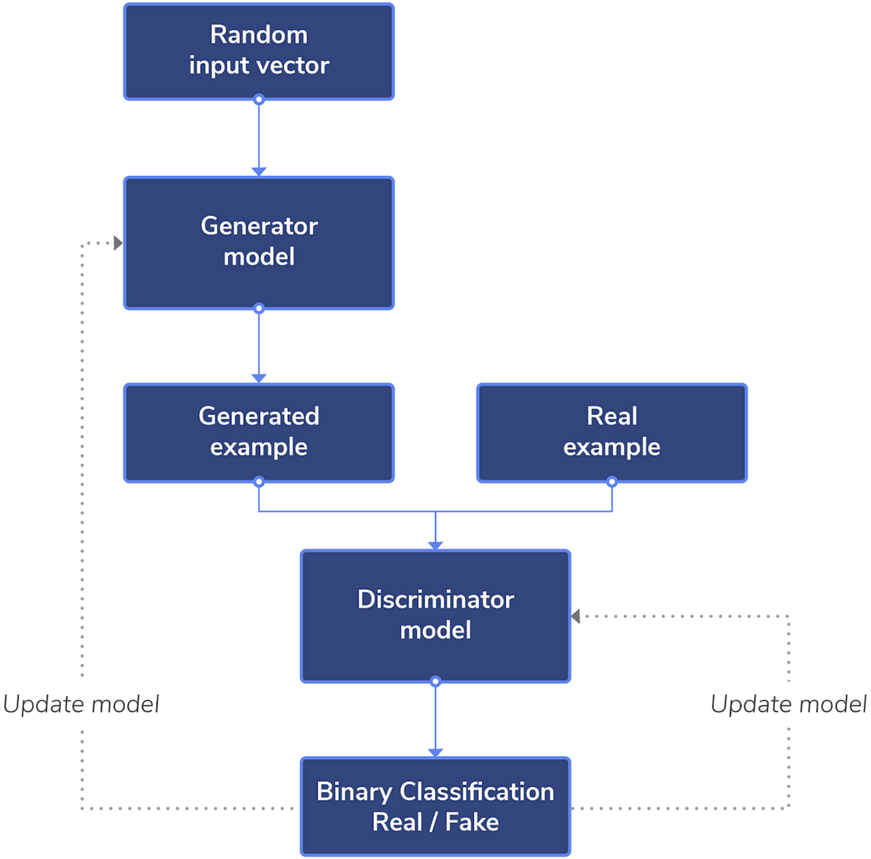
Objective

- implementing a Generative Adversarial Network (GAN) using TensorFlow to generate handwritten digit images from the MNIST dataset
- We define the generator and discriminator model. The models are configured with loss functions and optimizers.
- During training, sample images are generated and saved periodically.

Data Augmentation: It can be used to generate additional handwritten digit images to augment datasets for training machine learning models, especially in tasks like digit recognition. Augmented datasets can help improve model generalization and performance

MNIST Dataset





Random input vector

Generator model

Generated example

Real example

Discriminator model

Binary Classification Real / Fake

Update model

Update model

Generator model

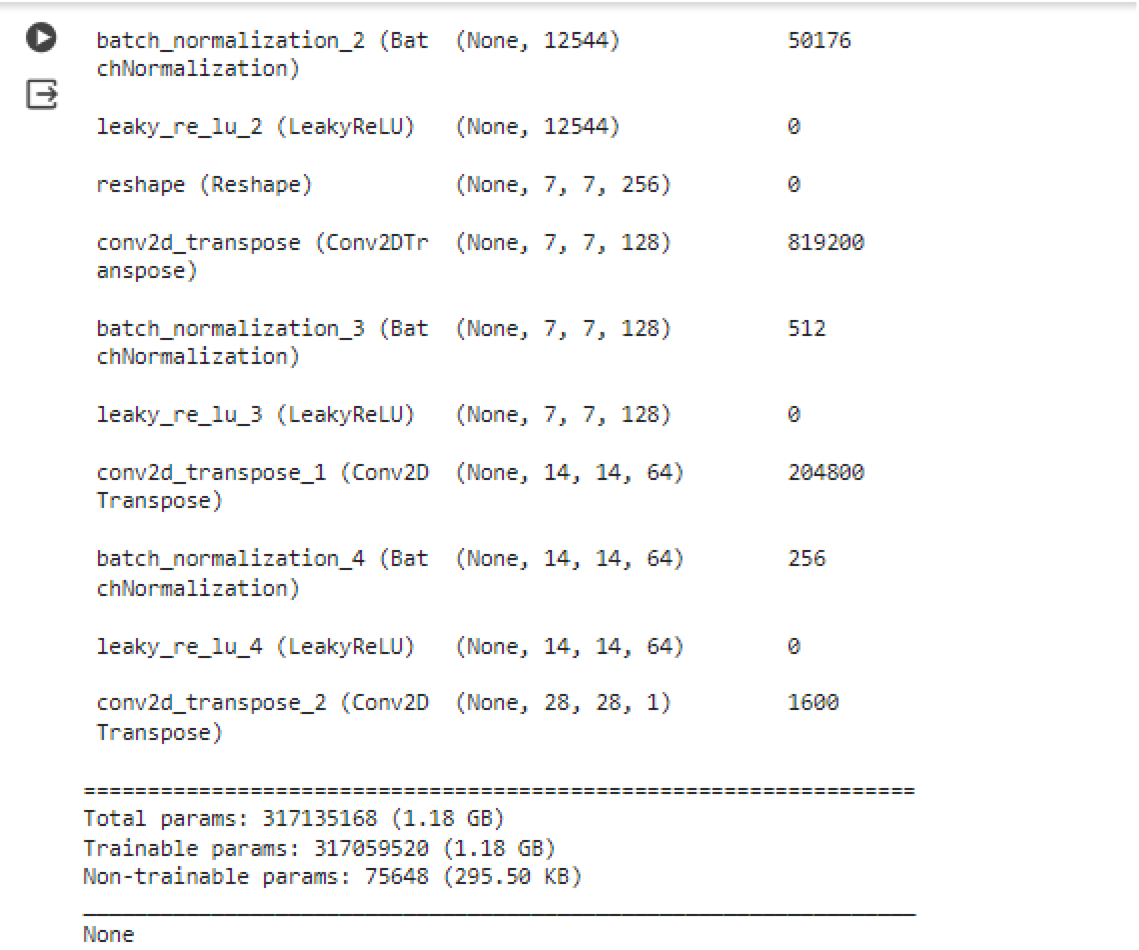
- **Input Layer**: The model starts with a densely connected layer (Dense) that takes a random noise vector (of size 100) as input. This noise vector is essentially random numbers used to generate unique images.

- **Normalization and Activation**: After the input layer, Batch Normalization is applied to standardize the activations. Then, LeakyReLU activation is used to introduce non-linearity, allowing the model to learn complex patterns

- **Transposed Convolutional Layers**: The model consists of several transposed convolutional layers (Conv2DTranspose), which learn to upsample the input noise vector into a meaningful image.

- **Output Layer**: Finally, the output layer produces the generated image. It uses a transposed convolutional layer with a 'tanh' activation function to ensure that the pixel values of the output image are in the range [-1, 1]

Generator has approximately **317 M** parameters



```
batch_normalization_2 (Bat  (None, 12544)             50176
chNormalization)

leaky_re_lu_2 (LeakyReLU)   (None, 12544)             0

reshape (Reshape)           (None, 7, 7, 256)         0

conv2d_transpose (Conv2DTr  (None, 7, 7, 128)         819200
anspose)

batch_normalization_3 (Bat  (None, 7, 7, 128)         512
chNormalization)

leaky_re_lu_3 (LeakyReLU)   (None, 7, 7, 128)         0

conv2d_transpose_1 (Conv2D  (None, 14, 14, 64)        204800
Transpose)

batch_normalization_4 (Bat  (None, 14, 14, 64)        256
chNormalization)

leaky_re_lu_4 (LeakyReLU)   (None, 14, 14, 64)        0

conv2d_transpose_2 (Conv2D  (None, 28, 28, 1)         1600
Transpose)

=================================================================
Total params: 317135168 (1.18 GB)
Trainable params: 317059520 (1.18 GB)
Non-trainable params: 75648 (295.50 KB)
_________________________________________________________________
None
```

Discriminator model

- **Convolutional Layers**: The model begins with a series of convolutional layers (Conv2D) that learn to extract features from input images.

- **Activation and Dropout**: After each convolutional layer, LeakyReLU activation is applied to introduce non-linearity. Additionally, Dropout is used to randomly deactivate some neurons during training, which helps prevent overfitting by adding noise to the network

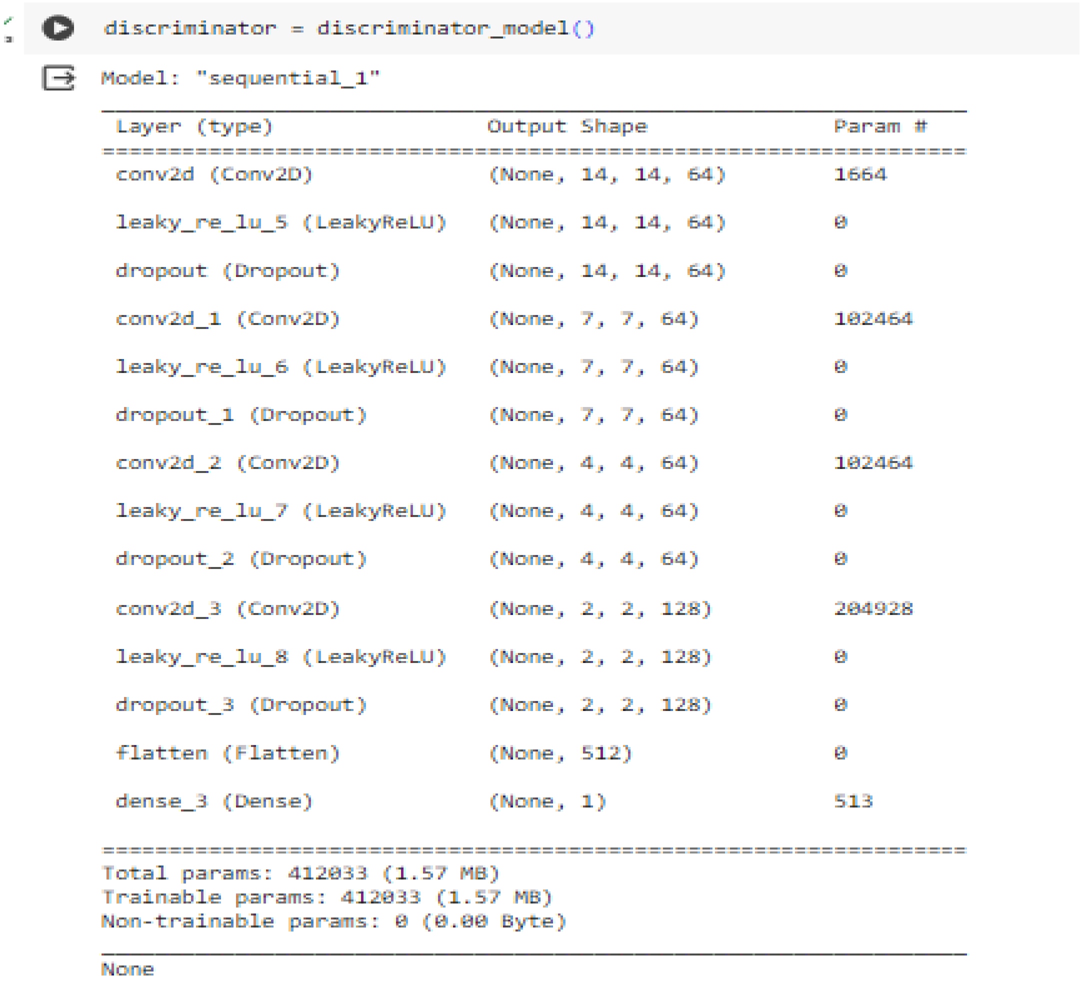
- **Flattening**: The output of the convolutional layers is flattened into a 1D vector to prepare for the final classification step.

- **Dense Layer**: Following the flattening, a densely connected layer (Dense) is used for classification. This layer learns to map the extracted features to a single output value, indicating whether the input image is real or fake.

Calculate Loss Function:
Cross- Entropy

Optimizer: Adam

Discriminator Description

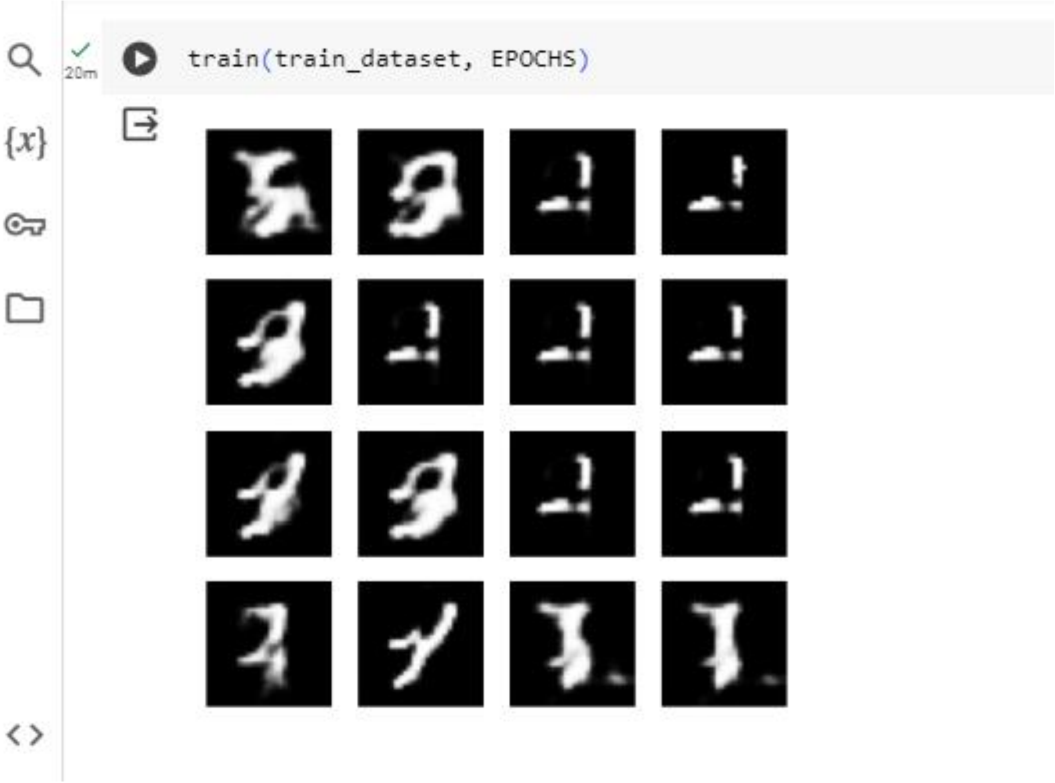


```
discriminator = discriminator_model()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 14, 14, 64)        1664

 leaky_re_lu_5 (LeakyReLU)   (None, 14, 14, 64)        0

 dropout (Dropout)           (None, 14, 14, 64)        0

 conv2d_1 (Conv2D)           (None, 7, 7, 64)          102464

 leaky_re_lu_6 (LeakyReLU)   (None, 7, 7, 64)          0

 dropout_1 (Dropout)         (None, 7, 7, 64)          0

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          102464

 leaky_re_lu_7 (LeakyReLU)   (None, 4, 4, 64)          0

 dropout_2 (Dropout)         (None, 4, 4, 64)          0

 conv2d_3 (Conv2D)           (None, 2, 2, 128)         204928

 leaky_re_lu_8 (LeakyReLU)   (None, 2, 2, 128)         0

 dropout_3 (Dropout)         (None, 2, 2, 128)         0

 flatten (Flatten)           (None, 512)               0

 dense_3 (Dense)             (None, 1)                 513

=================================================================
Total params: 412033 (1.57 MB)
Trainable params: 412033 (1.57 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Results

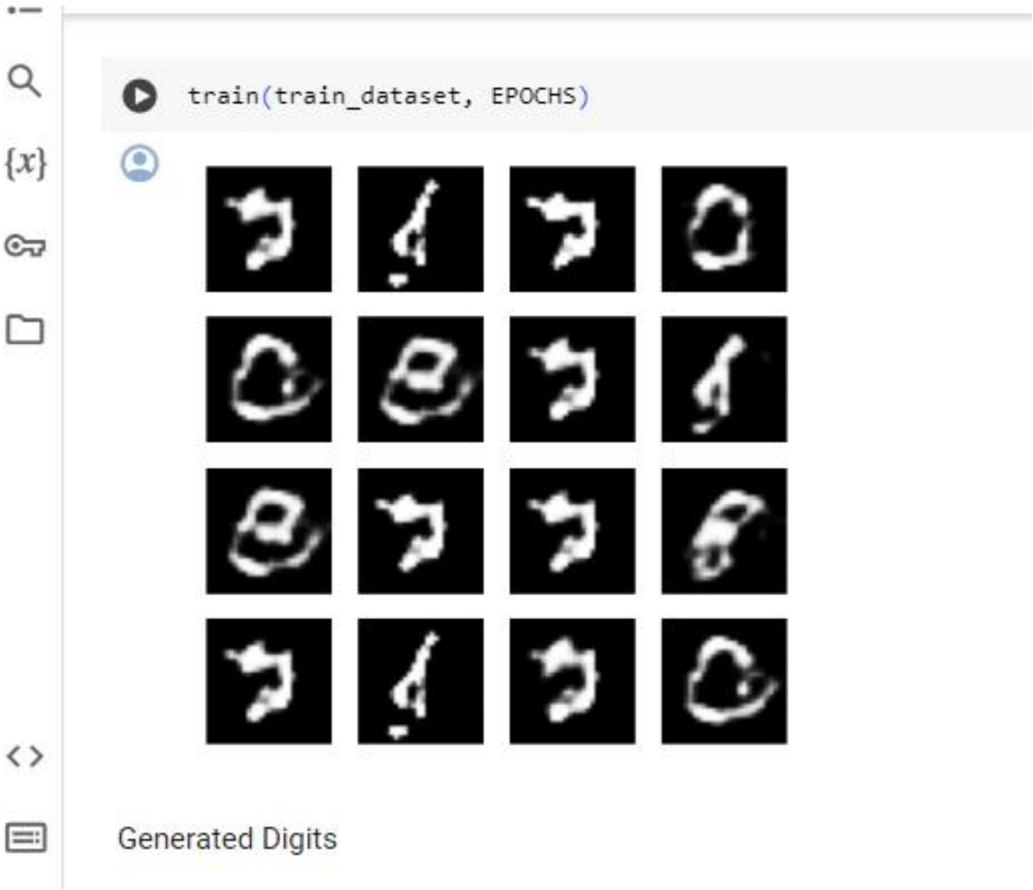
70 Epochs



```
train(train_dataset, EPOCHS)
```

Results

100+ Epochs

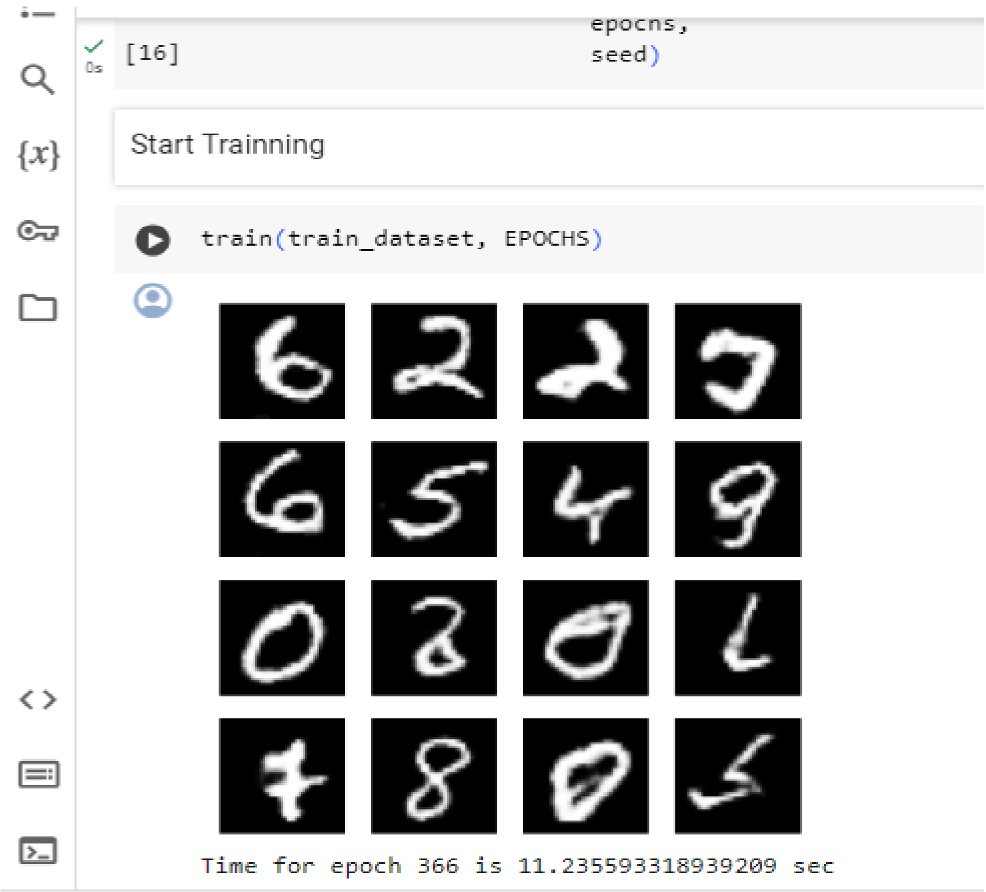


```
train(train_dataset, EPOCHS)
```

Generated Digits

Results

300+ Epochs



```
[16]                                        epochs,
                                            seed)
```

Start Trainning

```
train(train_dataset, EPOCHS)
```

```
Time for epoch 366 is 11.235593318939209 sec
```

Summary

- My project implements a Generative Adversarial Network (GAN) using TensorFlow to generate handwritten digit images from the MNIST dataset. The GAN consists of a Generator model and a Discriminator model, which are trained alternately to generate realistic-looking handwritten digits
- After training, the Generator can generate new handwritten digit images that closely resemble those in the MNIST dataset.

Future Work:

- **Improving Image Quality**: Experiment with different architectures, loss functions, and training strategies to improve the quality of generated images
- Experimenting with different datasets is a promising avenue for future work in my project.