**HANDWRITTEN DIGIT RECOGNITION USING GAN**

**SOURCE CODE**

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Reshape, Flatten

from tensorflow.keras.optimizers import Adam

# Load the MNIST dataset

(X\_train, \_), (\_, \_) = mnist.load\_data()

X\_train = (X\_train.astype(np.float32) - 127.5) / 127.5 # Normalize the images to [-1, 1]

X\_train = X\_train.reshape(X\_train.shape[0], 28\*28)

# Generator model

generator = Sequential([

Dense(256, input\_dim=100, activation='relu'),

Dense(512, activation='relu'),

Dense(28\*28, activation='tanh'),

Reshape((28, 28))

])

# Discriminator model

discriminator = Sequential([

Flatten(input\_shape=(28, 28)),

Dense(512, activation='relu'),

Dense(256, activation='relu'),

Dense(1, activation='sigmoid')

])

# Compile the discriminator

discriminator.compile(loss='binary\_crossentropy', optimizer=Adam(lr=0.0002, beta\_1=0.5))

# Combined GAN model

discriminator.trainable = False

gan\_input = Input(shape=(100,))

gan\_output = discriminator(generator(gan\_input))

gan = Model(gan\_input, gan\_output)

gan.compile(loss='binary\_crossentropy', optimizer=Adam(lr=0.0002, beta\_1=0.5))

# Training loop

batch\_size = 128

epochs = 10000

for epoch in range(epochs):

# Generate random noise

noise = np.random.normal(0, 1, (batch\_size, 100))

# Generate fake images

fake\_images = generator.predict(noise)

# Combine real and fake images

X\_combined = np.concatenate([X\_train[np.random.randint(0, X\_train.shape[0], batch\_size)], fake\_images])

# Labels for real and fake images

y\_combined = np.concatenate([np.ones((batch\_size, 1)), np.zeros((batch\_size, 1))])

# Train the discriminator

d\_loss = discriminator.train\_on\_batch(X\_combined, y\_combined)

# Generate noise for the generator

noise = np.random.normal(0, 1, (batch\_size, 100))

y\_fake = np.ones((batch\_size, 1))

# Train the generator (via the combined model)

g\_loss = gan.train\_on\_batch(noise, y\_fake)

# Print progress

if epoch % 100 == 0:

print(f"Epoch: {epoch}, Discriminator Loss: {d\_loss}, Generator Loss: {g\_loss}")

# Generate and display generated images

noise = np.random.normal(0, 1, (10, 100))

generated\_images = generator.predict(noise)

plt.figure(figsize=(10, 10))

for i in range(10):

plt.subplot(1, 10, i+1)

plt.imshow(generated\_images[i], cmap='gray')

plt.axis('off')

plt.show()