Amardeep Singh

E23CSEU2189

```
import tensorflow as tf
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt
import os

# ================= PARAMETERS =================
epochs = 50
batch_size = 128
noise_dim = 100
save_interval = 5

# Learning rates
d_lr = 0.0002
g_lr = 0.0001

# ================= FOLDERS =================
os.makedirs("generated_samples", exist_ok=True)
os.makedirs("final_generated_images", exist_ok=True)

# ================= LOAD MNIST =================
(x_train, _), _ = tf.keras.datasets.mnist.load_data()

x_train = (x_train.astype("float32") - 127.5) / 127.5
x_train = np.expand_dims(x_train, axis=-1)

dataset = tf.data.Dataset.from_tensor_slices(x_train)\
          .shuffle(60000).batch(batch_size, drop_remainder=True)

# ================= GENERATOR =================
def build_generator():
    return tf.keras.Sequential([
        layers.Dense(256, input_dim=noise_dim),
        layers.LeakyReLU(0.2),

        layers.Dense(512),
        layers.LeakyReLU(0.2),

        layers.Dense(1024),
        layers.LeakyReLU(0.2),

        layers.Dense(28 * 28, activation="tanh"),
        layers.Reshape((28, 28, 1))
    ])

# ================= DISCRIMINATOR (NO SIGMOID!) =================
def build_discriminator():
    return tf.keras.Sequential([
        layers.Flatten(input_shape=(28, 28, 1)),

        layers.Dense(512),
        layers.LeakyReLU(0.2),

        layers.Dense(256),
        layers.LeakyReLU(0.2),

        layers.Dense(1)   # LOGITS OUTPUT
    ])

generator = build_generator()
discriminator = build_discriminator()

# ================= LOSSES =================
bce = tf.keras.losses.BinaryCrossentropy(from_logits=True)

def d_loss_fn(real_logits, fake_logits):
    real_loss = bce(tf.ones_like(real_logits), real_logits)
    fake_loss = bce(tf.zeros_like(fake_logits), fake_logits)
    return real_loss + fake_loss

def g_loss_fn(fake_logits):
    return bce(tf.ones_like(fake_logits), fake_logits)

# ================= OPTIMIZERS =================
d_optimizer = tf.keras.optimizers.Adam(d_lr, beta_1=0.5)
```

```
g_optimizer = tf.keras.optimizers.Adam(g_lr, beta_1=0.5)

# ================= TRAIN STEP =================
@tf.function
def train_step(real_images):
    batch = tf.shape(real_images)[0]
    noise = tf.random.normal([batch, noise_dim])

    with tf.GradientTape() as d_tape, tf.GradientTape() as g_tape:
        fake_images = generator(noise, training=True)

        real_logits = discriminator(real_images, training=True)
        fake_logits = discriminator(fake_images, training=True)

        d_loss = d_loss_fn(real_logits, fake_logits)
        g_loss = g_loss_fn(fake_logits)

    d_grads = d_tape.gradient(d_loss, discriminator.trainable_variables)
    g_grads = g_tape.gradient(g_loss, generator.trainable_variables)

    d_optimizer.apply_gradients(zip(d_grads, discriminator.trainable_variables))
    g_optimizer.apply_gradients(zip(g_grads, generator.trainable_variables))

    return d_loss, g_loss

# ================= SAVE IMAGE GRID =================
def save_images(epoch):
    noise = tf.random.normal([25, noise_dim])
    imgs = generator(noise, training=False)
    imgs = (imgs + 1) / 2

    fig, axs = plt.subplots(5, 5, figsize=(5, 5))
    idx = 0
    for i in range(5):
        for j in range(5):
            axs[i, j].imshow(imgs[idx, :, :, 0], cmap="gray")
            axs[i, j].axis("off")
            idx += 1

    plt.savefig(f"generated_samples/epoch_{epoch:02d}.png")
    plt.close()

# ================= TRAIN LOOP =================
for epoch in range(1, epochs + 1):
    for real_imgs in dataset:
        d_loss, g_loss = train_step(real_imgs)

    print(
        f"Epoch {epoch}/{epochs} | "
        f"D_loss: {d_loss:.3f} | "
        f"G_loss: {g_loss:.3f}"
    )

    if epoch % save_interval == 0:
        save_images(epoch)

# ================= FINAL IMAGE GENERATION =================
noise = tf.random.normal([100, noise_dim])
final_imgs = generator(noise, training=False)
final_imgs = (final_imgs + 1) / 2

for i in range(100):
    plt.imsave(
        f"final_generated_images/img_{i}.png",
        final_imgs[i, :, :, 0],
        cmap="gray"
    )

print("\n✅ Training complete. G_loss will NOT be zero.")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_sl
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `
  super().__init__(**kwargs)
Epoch 1/50 | D_loss: 0.746 | G_loss: 1.645
Epoch 2/50 | D_loss: 0.646 | G_loss: 0.839
Epoch 3/50 | D_loss: 0.392 | G_loss: 1.839
Epoch 4/50 | D_loss: 0.378 | G_loss: 2.006
Epoch 5/50 | D_loss: 0.422 | G_loss: 2.519
Epoch 6/50 | D_loss: 0.443 | G_loss: 3.036
Epoch 7/50 | D_loss: 0.591 | G_loss: 1.409
Epoch 8/50 | D_loss: 0.523 | G_loss: 2.179
Epoch 9/50 | D_loss: 0.451 | G_loss: 1.663
Epoch 10/50 | D_loss: 0.624 | G_loss: 1.560
Epoch 11/50 | D_loss: 0.499 | G_loss: 1.304
Epoch 12/50 | D_loss: 0.651 | G_loss: 3.716
Epoch 13/50 | D_loss: 0.347 | G_loss: 2.286
Epoch 14/50 | D_loss: 0.391 | G_loss: 2.395
Epoch 15/50 | D_loss: 0.535 | G_loss: 2.943
Epoch 16/50 | D_loss: 0.471 | G_loss: 1.876
Epoch 17/50 | D_loss: 0.594 | G_loss: 1.516
Epoch 18/50 | D_loss: 0.500 | G_loss: 2.544
Epoch 19/50 | D_loss: 0.505 | G_loss: 1.646
Epoch 20/50 | D_loss: 0.653 | G_loss: 1.884
Epoch 21/50 | D_loss: 0.718 | G_loss: 1.430
Epoch 22/50 | D_loss: 1.090 | G_loss: 2.571
Epoch 23/50 | D_loss: 0.915 | G_loss: 0.945
Epoch 24/50 | D_loss: 0.825 | G_loss: 1.354
Epoch 25/50 | D_loss: 0.933 | G_loss: 1.349
Epoch 26/50 | D_loss: 0.942 | G_loss: 1.579
Epoch 27/50 | D_loss: 1.005 | G_loss: 2.109
Epoch 28/50 | D_loss: 0.954 | G_loss: 0.995
Epoch 29/50 | D_loss: 0.828 | G_loss: 1.180
Epoch 30/50 | D_loss: 0.996 | G_loss: 1.714
Epoch 31/50 | D_loss: 0.949 | G_loss: 1.607
Epoch 32/50 | D_loss: 1.009 | G_loss: 0.898
Epoch 33/50 | D_loss: 0.850 | G_loss: 1.483
Epoch 34/50 | D_loss: 0.898 | G_loss: 1.435
Epoch 35/50 | D_loss: 0.922 | G_loss: 1.210
Epoch 36/50 | D_loss: 0.923 | G_loss: 1.309
Epoch 37/50 | D_loss: 0.953 | G_loss: 1.160
Epoch 38/50 | D_loss: 1.041 | G_loss: 1.446
Epoch 39/50 | D_loss: 0.973 | G_loss: 1.069
Epoch 40/50 | D_loss: 0.965 | G_loss: 1.082
Epoch 41/50 | D_loss: 1.050 | G_loss: 1.281
Epoch 42/50 | D_loss: 0.917 | G_loss: 1.129
Epoch 43/50 | D_loss: 0.988 | G_loss: 0.920
Epoch 44/50 | D_loss: 0.961 | G_loss: 1.449
Epoch 45/50 | D_loss: 0.968 | G_loss: 1.238
Epoch 46/50 | D_loss: 1.002 | G_loss: 0.991
Epoch 47/50 | D_loss: 1.068 | G_loss: 0.780
Epoch 48/50 | D_loss: 1.174 | G_loss: 1.864
Epoch 49/50 | D_loss: 1.000 | G_loss: 1.257
Epoch 50/50 | D_loss: 0.933 | G_loss: 1.591
```

```python
import shutil
import os
from IPython.display import FileLink, display

# Zip generated_samples folder
shutil.make_archive(
    "generated_samples",   # zip file name
    'zip',
    "generated_samples"    # folder to zip
)

# Zip final_generated_images folder
shutil.make_archive(
    "final_generated_images",
    'zip',
    "final_generated_images"
)

print("✅ ZIP files created successfully!")

# ===== DOWNLOAD LINKS =====
display(FileLink("generated_samples.zip"))
display(FileLink("final_generated_images.zip"))
```

```
✅ ZIP files created successfully!
generated_samples.zip
final_generated_images.zip
```