Generative Adversarial Network (GAN) for Handwritten Digit Generation

1. Objective

The objective of this assignment is to design, implement, and train a **Generative Adversarial Network (GAN)** using **TensorFlow (Keras API)** to generate realistic handwritten digit images similar to those in the **MNIST dataset**.
This assignment aims to strengthen understanding of **adversarial learning**, **deep generative models**, and **training dynamics of GANs**.

2. Brief Explanation of GANs

A **Generative Adversarial Network (GAN)** consists of two neural networks that are trained simultaneously in a competitive (adversarial) manner:

- **Generator (G):**
  Generates fake images from random noise vectors with the goal of producing outputs that resemble real data.
- **Discriminator (D):**
  Classifies input images as either **real** (from the dataset) or **fake** (generated by the generator).

During training:

- The generator tries to **fool the discriminator**
- The discriminator tries to **correctly identify real vs fake images**

This adversarial process continues until the generator produces images that are indistinguishable from real data.

3. Dataset and Preprocessing

- **Dataset:** MNIST handwritten digit dataset
- **Image size:** 28 × 28 (grayscale)
- **Preprocessing steps:**
  - Pixel values normalized to the range **[-1, 1]**
  - Images reshaped to **(28, 28, 1)** to match convolutional input format

Normalization improves training stability and works well with the **Tanh** activation used in the generator's output layer.

4. Architecture Summary

4.1 Generator Network

The generator takes a **100-dimensional random noise vector** as input and generates a 28×28 grayscale image.

Architecture:

- Dense layer
- Batch Normalization
- LeakyReLU activation
- Reshape layer
- Conv2DTranspose layers
- Output layer with **Tanh** activation

Purpose:
Upsamples noise into meaningful image structures.

4.2 Discriminator Network

The discriminator takes an image as input and outputs a probability indicating whether the image is real or fake.

Architecture:

- Conv2D layers
- LeakyReLU activation
- Dropout layers
- Flatten layer
- Dense output layer with **Sigmoid** activation

Purpose:
Acts as a binary classifier distinguishing real images from generated images.

5. Training Details

- **Loss Function:** Binary Cross-Entropy (BCE)
- **Optimizer:** Adam
    - Learning rate = 0.0002
    - $\beta_1 = 0.5$
- **Epochs:** 20

- **Batch size:** 256

Two losses are tracked during training:

- **Generator Loss:** Measures how well the generator fools the discriminator
- **Discriminator Loss:** Measures how accurately the discriminator classifies real and fake images

6. Generated Outputs and Loss Curves

- A fixed random noise vector is used to generate sample images at regular intervals.
- Generated images are displayed in a **4×4 grid** after every few epochs.
- Loss curves for both the generator and discriminator are plotted to analyze training behavior.

Loss Graphs

- Initially, discriminator loss is low while generator loss is high.
- Over time, losses stabilize, indicating balanced adversarial learning.

Sample Generated Images

- Early epochs show noisy and unclear digit shapes.
- As training progresses, digit structures become more recognizable.

7. Observations

- GAN training is highly sensitive to hyperparameters.
- Image quality improves gradually with epochs.
- Visual inspection is essential, as loss values alone do not fully represent GAN performance.
- Generator and discriminator reach a competitive balance after several epochs.

8. Challenges Faced

- **Training instability:** GANs can oscillate and fail to converge.
- **Mode collapse risk:** Generator may produce limited variety of outputs.
- **Hyperparameter tuning:** Learning rate and optimizer settings significantly affect results.
- **Evaluation difficulty:** GANs lack straightforward quantitative evaluation metrics.

9. Conclusion

This assignment demonstrates the implementation of a GAN for handwritten digit generation using TensorFlow. The adversarial training framework successfully enables the generator to learn meaningful image representations. Despite challenges such as instability and evaluation difficulty, GANs remain powerful tools for generative modeling tasks.