Assignment-3

Train a GAN (Generative Adversarial Network) model using PyTorch on the MNIST dataset to generate digit images. The MNIST dataset consists of grayscale handwritten digit images with a resolution of 28×28 pixels.

Part-1

In this paert, you are required to complete the following steps:

1. Read the codebook of this question: **Assignment_3_GAN.ipynb**.
2. **Architecture:** define multiple linear layers in the Generator and Discriminator, respectively. The architecture is provided in Figure 1. You are asked to use a sequential way to implement each model (**10 pts**) (TODO-1)
3. **Optimization:** define the loss function and optimizers. **(5 pts)** (TODO-2)
4. **Training:** complete the training loop, including: (TODO-3)
   - Labels for real images and fake images when training the discriminator. **(5 pts)**
   - Calculation of the discriminator's loss for real images. **(5 pts)**
   - Calculation of the discriminator's loss for fake images. **(5 pts)**
   - Calculation of the generator's loss. **(5 pts)**
   - Freeze or unfreeze the generator or discriminator when appropriate. **(5 pts)**
5. **Plots:** plot the training loss curves. **(5 pts)** (TODO-4)
6. **Evaluation:** Generate and save 25 images using the trained generator. **(5 pts)** (TODO-5)

Generator

Layers	Input Dim.	Output Dim.
Layer-1	100	256
Layer-2	256	512
Layer-3	512	1024
Layer-4	1024	1*28*28
Last Layer	nn.Tanh()	

Discriminator

Layers	Input Dim.	Output Dim.
Layer-1	1*28*28	1024
Layer-2	1024	512
Layer-3	512	256
Layer-4	256	1
Last Layer	nn.Sigmoid()	

Figure 1: Structure of the GAN

Part-2

In this part, you are required to conduct further experimental analysis based on the code in **Part-1**. Note that you may need to train multiple models under different parameter settings as necessary.

1. Conduct a comprehensive analysis of the generated image quality across different training stages (e.g., every 5 epochs). The analysis should include appropriate visualizations. **(10 pts)**
2. Investigate the impact of batch size (specifically, sizes 128, 256, 512, and 1024) on the training stability of the GAN and the quality of the generated images. **(20 pts)**
3. In the last question, you might have found the training unstable in some cases. Implement the following learning rate scheduling strategies: Step Decay and Warmup + Cosine Annealing, and attempt to use one of them to resolve the instability issues encountered in the previous question. Analyze how these strategies affect model convergence. **(20 pts)**
   - **Step Decay**: The learning rate is reduced by a fixed factor (e.g., multiplied by 0.1) at predefined intervals (e.g., every 20 epochs). This provides a structured, step-wise decrease in the learning rate.
   - **Warmup + Cosine Annealing**: This strategy first gradually increases the learning rate during a warmup phase, then smoothly decreases it following a cosine curve. This helps stabilize early training and often leads to better model performance.

Note

For Part I, you need to fill in the code in Assignment_3_GAN.ipynb as required, complete the training, and visualize the loss and images. For this part, you only need to submit this single Jupyter file.

For Part II, you need to refer to the code in Assignment_3_GAN.ipynb and create three separate Python or Jupyter files as required. Write the corresponding code and conduct experiments accordingly. For this part, you need to submit the three code files and an experimental report. The experimental report should include visualizations and appropriate analysis (limited to 3 pages in PDF format).

When submitting the assignment, **be sure to remove unnecessary data**, including the MNIST dataset and checkpoints.