Machine Learning
LABORATORY: GAN In Class

NAME:	STUDENT ID#:

Objectives:

- Understand the training dynamics of a **Generative Adversarial Network (GAN).**
- Implement the adversarial loss used to train **GANs.**
- Train a GAN to generate handwritten digit images using the **MNIST** dataset.
- Apply **PyTorch** operations to manually train both a generator and a discriminator.
- Visualize generated results and observe how outputs evolve over training.

Instructions:

- In this assignment, you will complete the training loop for a **Generative Adversarial Network (GAN)** using the code template provided in class.
- Your task is to:
- Train a **generator** that learns to produce handwritten digit images from random noise.
- Train a **discriminator** that learns to distinguish between real MNIST images and generated (fake) ones.
- Implement the GAN loss using **binary cross-entropy** to update both models.
- Train using **PyTorch** — do not use higher-level wrappers (like nn.GAN libraries).
- Visualize the output to observe how GAN works. Please compare the input image, the fake image, and the generated image.

Code Template.

Step	Procedure

1	#Load Dataset
	import torch
	import torch.nn as nn
	import torch.optim as optim
	from torchvision import datasets, transforms
	from torch.utils.data import DataLoader
	import matplotlib.pyplot as plt
	import numpy as np
	import torchvision
	
	# Load MNIST
	transform = transforms.Compose([
	    transforms.ToTensor(),
	    transforms.Normalize((0.5,), (0.5,)),
	])
	dataset = datasets.MNIST(root='./data', train=True, transform=transform, download=True)

	loader = DataLoader(dataset, batch_size=64, shuffle=True)
	
	target_digit = 3 # you can change this based on your last digit student number
	
	# Reload MNIST and filter to only target_digit
	dataset = MNIST(root='./data', train=True, transform=transform, download=True)
	filtered_indices = [i for i, (_, label) in enumerate(dataset) if label == target_digit]
	filtered_dataset = Subset(dataset, filtered_indices[:5000]) # limit to 5000 samples
	
	loader = DataLoader(filtered_dataset, batch_size=64, shuffle=True)
	
	# Visualize filtered samples
	examples = next(iter(loader))[0][:32]
	examples = examples * 0.5 + 0.5
	grid = torchvision.utils.make_grid(examples, nrow=8, padding=2, normalize=True)
	plt.figure(figsize=(8, 8))
	plt.imshow(np.transpose(grid, (1, 2, 0)))
	plt.title(f"Real MNIST Digit '{target_digit}' Only (Before Training)")
	plt.axis("off")
	plt.show()

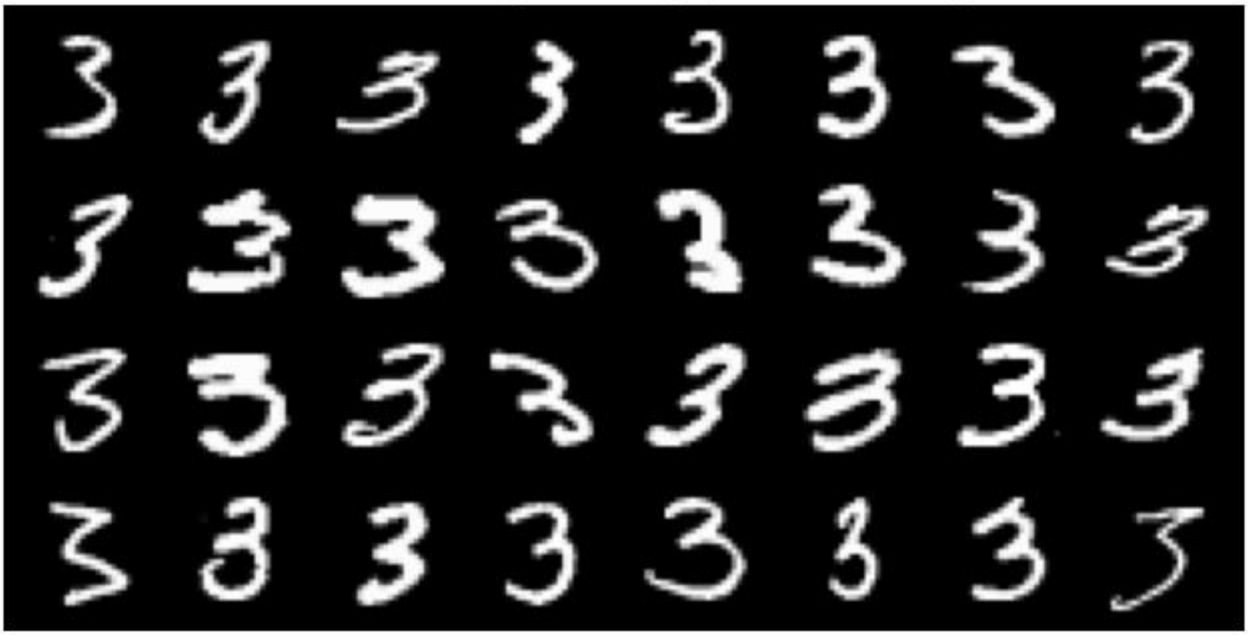
2	# ====== Generator and Discriminator Definitions ======
	# Define the Generator
	class Generator(nn.Module):
	    def __init__(self, z_dim=100, img_dim=784):
	        super().__init__()
	        self.gen = nn.Sequential(
	            # Define your generator architecture here
	        )
	
	    def forward(self, x):
	        return self.gen(x)
	
	# Define the Discriminator
	class Discriminator(nn.Module):
	    def __init__(self, img_dim=784):
	        super().__init__()
	        self.disc = nn.Sequential(
	            # Define your discriminator architecture here
	        )
	
	    def forward(self, x):
	        return self.disc(x)

3	# ========== Training Setup ==========
	# Initialize networks and optimizers, you can adjust the parameters
	z_dim = 100
	lr = 0.0002

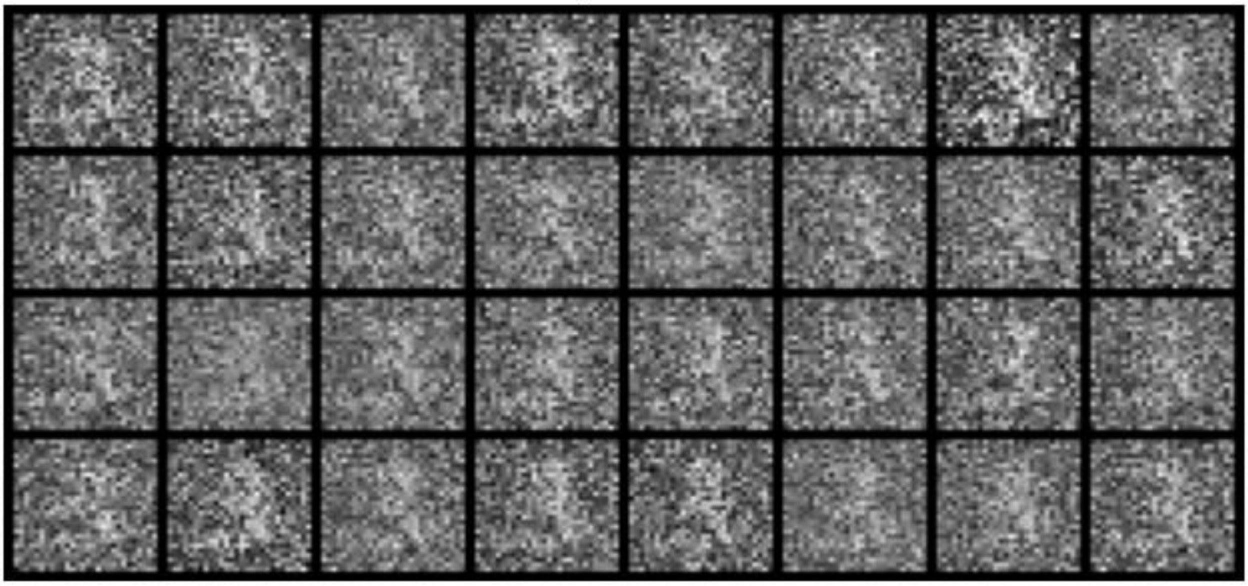
	gen = Generator(z_dim) disc = Discriminator() criterion = nn.BCELoss() opt_gen = optim.Adam(gen.parameters(), lr) opt_disc = optim.Adam(disc.parameters(), lr)
4	# Write training loop with GAN adversarial loss here # TODO: implement training loop with real/fake labels, forward passes, and optim steps # Example training loop (pseudo-code): # 1. Loop over epochs and batches: # for epoch in range(num_epochs): # for batch_idx, (real, _) in enumerate(loader): # 2. Flatten and move real images to the device: # real = real.view(-1, 784).to(device) # batch_size = real.size(0) # 3. Generate fake images from noise: # noise = torch.randn(batch_size, z_dim).to(device) # fake = gen(noise) # 4. Train Discriminator # 5. Train Generator # 6. Print epoch loss values: # print(f"Epoch [{epoch+1}/{num_epochs}] Loss D: {loss_disc:.4f}, Loss G: {loss_gen:.4f}") # 7. Visualize generated samples after training

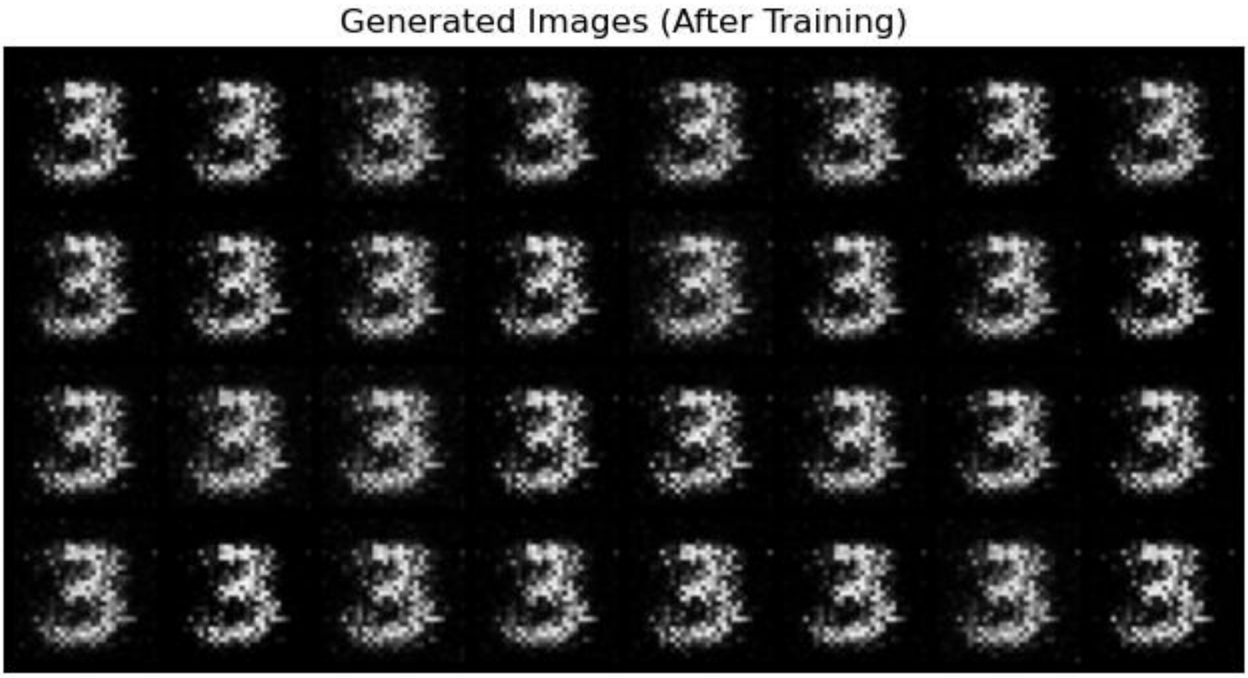
Example Output:

Real MNIST Digit '3' Only (Before Training)



Epoch 1





Grading Assignment & Submission (30% Max)

Implementation:

1. **(10%) GAN Training Loop Implementation**
   Implement the training loop using PyTorch:
   a. Generator: generates fake images from noise
   b. Discriminator: classifies real vs. fake images
   c. Loss: Binary Cross Entropy for both models
2. **(10%) Code Runs Without Errors + Successful Training Output**
   a. The model runs end-to-end without errors
   b. Trains using the MNIST dataset
   c. Displays GAN loss values per epoch
   d. Outputs generated samples as images
3. **(5%) Generated Style Focus: Use Last Digit of Student ID as Target Style**
   a. Train your GAN only on a **single digit** (e.g., if ID ends in 7 → use class '7')
   b. The generator should produce fake images that resemble that digit

Question:

1. **(5%) Briefly discuss your results.**

Submission:

1. Report: Answer all conceptual questions. Include screenshots of your results in the last pages of this PDF File.
2. Code: Submit your complete Python script in either .py or .ipynb format.
3. Upload both your report and code to the E3 system (**Lab8 In Class Assignment**). Name your files correctly:
   a. Report: StudentID_Lab8_InClass.pdf
   b. Code: StudentID_Lab8_ InClass.py or StudentID_Lab8_InClass.ipynb
4. Deadline: 16:20 PM

5. Plagiarism is **strictly prohibited**. Submitting copied work from other students will result in penalties.

Answer: