Name - Akshat D Jain PRN - 22070126136 AIML B3

GAN Assignment4

```
!pip install medmnist
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting medmnist
  Downloading medmnist-3.0.2-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from medmnist) (1.24.3)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (from medmnist) (1.5.3)
Requirement already satisfied: scikit-learn in c:\programdata\anaconda3\lib\site-packages (from medmnist) (1.3.0)
Requirement already satisfied: scikit-image in c:\programdata\anaconda3\lib\site-packages (from medmnist) (0.20.0)
Requirement already satisfied: tqdm in c:\users\jmdgo\appdata\roaming\python\python311\site-packages (from medmnist) (4.66.5)
Requirement already satisfied: Pillow in c:\programdata\anaconda3\lib\site-packages (from medmnist) (10.3.0)
Collecting fire (from medmnist)
  Downloading fire-0.7.0.tar.gz (87 kB)
     ---------------------------------------- 0.0/87.2 kB ? eta -:--:--
     ---------------------------- ----------- 61.4/87.2 kB 1.6 MB/s eta 0:00:01
     ------------------------------------ -- 81.9/87.2 kB 1.5 MB/s eta 0:00:01
     ------------------------------------ -- 81.9/87.2 kB 1.5 MB/s eta 0:00:01
     -------------------------------------- 87.2/87.2 kB 545.7 kB/s eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: torch in c:\users\jmdgo\appdata\roaming\python\python311\site-packages (from medmnist) (2.5.0)
Requirement already satisfied: torchvision in c:\users\jmdgo\appdata\roaming\python\python311\site-packages (from medmnist) (0
Requirement already satisfied: termcolor in c:\users\jmdgo\appdata\roaming\python\python311\site-packages (from fire->medmnist
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\anaconda3\lib\site-packages (from pandas->medmnist) (2
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas->medmnist) (2022.7)
Requirement already satisfied: scipy>=1.8 in c:\programdata\anaconda3\lib\site-packages (from scikit-image->medmnist) (1.10.1)
Requirement already satisfied: networkx>=2.8 in c:\programdata\anaconda3\lib\site-packages (from scikit-image->medmnist) (3.1)
Requirement already satisfied: imageio>=2.4.1 in c:\programdata\anaconda3\lib\site-packages (from scikit-image->medmnist) (2.2
Requirement already satisfied: tifffile>=2019.7.26 in c:\programdata\anaconda3\lib\site-packages (from scikit-image->medmnist)
Requirement already satisfied: PyWavelets>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from scikit-image->medmnist) (
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-image->medmnist) (23
Requirement already satisfied: lazy_loader>=0.1 in c:\programdata\anaconda3\lib\site-packages (from scikit-image->medmnist) (0
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn->medmnist) (1.2.
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn->medmnist
Requirement already satisfied: filelock in c:\programdata\anaconda3\lib\site-packages (from torch->medmnist) (3.9.0)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\jmdgo\appdata\roaming\python\python311\site-packages (from
Requirement already satisfied: jinja2 in c:\programdata\anaconda3\lib\site-packages (from torch->medmnist) (3.1.2)
Requirement already satisfied: fsspec in c:\users\jmdgo\appdata\roaming\python\python311\site-packages (from torch->medmnist)
Requirement already satisfied: sympy==1.13.1 in c:\users\jmdgo\appdata\roaming\python\python311\site-packages (from torch->med
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\programdata\anaconda3\lib\site-packages (from sympy==1.13.1->torch->me
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from tqdm->medmnist) (0.4.6)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas->me
Requirement already satisfied: MarkupSafe>=2.0 in c:\programdata\anaconda3\lib\site-packages (from jinja2->torch->medmnist) (2
Downloading medmnist-3.0.2-py3-none-any.whl (25 kB)
Building wheels for collected packages: fire
  Building wheel for fire (setup.py): started
  Building wheel for fire (setup.py): finished with status 'done'
  Created wheel for fire: filename=fire-0.7.0-py3-none-any.whl size=114263 sha256=19fea2bc76e773e9ff6c5cdeec92afd5002999a6b8f0
  Stored in directory: c:\users\jmdgo\appdata\local\pip\cache\wheels\46\54\24\1624fd5b8674eb1188623f7e8e17cdf7c0f6c24b609dfb8a
Successfully built fire
Installing collected packages: fire, medmnist
Successfully installed fire-0.7.0 medmnist-3.0.2

[notice] A new release of pip is available: 24.1.2 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.utils as vutils
from torch.utils.data import DataLoader
from medmnist import ChestMNIST
from torch.utils.tensorboard import SummaryWriter
```

```
# Load ChestMNIST dataset
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

dataset = ChestMNIST(split='train', transform=transform, download=True)
dataloader = DataLoader(dataset, batch_size=64, shuffle=True)
```

```
Downloading https://zenodo.org/records/10519652/files/chestmnist.npz?download=1 to C:\Users\jmdgo\.medmnist\chestmnist.npz
100%|████████████████████████████████████████████████████████████████████████| 82.8M/82.8M [00:50<00:00, 1.64MB/s]
```

```
# Define a simple CNN-based Generator
class Generator(nn.Module):
    def __init__(self, latent_dim=100):
        super(Generator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(latent_dim, 128),
            nn.ReLU(),
            nn.Linear(128, 256),
            nn.ReLU(),
            nn.Linear(256, 512),
            nn.ReLU(),
            nn.Linear(512, 28*28),
            nn.Tanh()
        )

    def forward(self, z):
        img = self.model(z)
        img = img.view(img.size(0), 1, 28, 28)
        return img

# Define CNN-based Discriminator for LS-GAN and WGAN
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.LeakyReLU(0.2),
            nn.Linear(512, 256),
            nn.LeakyReLU(0.2),
            nn.Linear(256, 1)
        )

    def forward(self, img):
        img_flat = img.view(img.size(0), -1)
        validity = self.model(img_flat)
        return validity

# Initialize models
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
generator = Generator().to(device)
discriminator = Discriminator().to(device)
```

```
# Optimizers
optimizer_G = optim.Adam(generator.parameters(), lr=0.0002, betas=(0.5, 0.999))
optimizer_D = optim.Adam(discriminator.parameters(), lr=0.0002, betas=(0.5, 0.999))

# Loss function for LS-GAN
criterion_ls = nn.MSELoss()

# TensorBoard Writer
writer = SummaryWriter()

def train_lsgan():
    for epoch in range(50):
        for i, (imgs, _) in enumerate(dataloader):
            imgs = imgs.to(device)
            batch_size = imgs.size(0)
            valid = torch.ones(batch_size, 1).to(device)
            fake = torch.zeros(batch_size, 1).to(device)

            # Train Generator
            optimizer_G.zero_grad()
            z = torch.randn(batch_size, 100).to(device)
            gen_imgs = generator(z)
            g_loss = criterion_ls(discriminator(gen_imgs), valid)
            g_loss.backward()
            optimizer_G.step()

            # Train Discriminator
            optimizer_D.zero_grad()
            real_loss = criterion_ls(discriminator(imgs), valid)
            fake_loss = criterion_ls(discriminator(gen_imgs.detach()), fake)
            d_loss = (real_loss + fake_loss) / 2
            d_loss.backward()
            optimizer_D.step()

        # Log losses and generated images
        writer.add_scalar("Loss/Generator", g_loss.item(), epoch)
        writer.add_scalar("Loss/Discriminator", d_loss.item(), epoch)

        with torch.no_grad():
            z = torch.randn(64, 100).to(device)
            gen_imgs = generator(z)
            writer.add_images('Generated Images', gen_imgs, epoch)
            vutils.save_image(gen_imgs, f'generated_epoch_{epoch}.png', normalize=True)
```

```
def train_wgan():
    for epoch in range(50):
        for i, (imgs, _) in enumerate(dataloader):
            imgs = imgs.to(device)
            batch_size = imgs.size(0)

            # Train Generator
            optimizer_G.zero_grad()
            z = torch.randn(batch_size, 100).to(device)
            gen_imgs = generator(z)
            g_loss = -torch.mean(discriminator(gen_imgs))
            g_loss.backward()
            optimizer_G.step()

            # Train Discriminator
            optimizer_D.zero_grad()
            d_loss = torch.mean(discriminator(gen_imgs.detach())) - torch.mean(discriminator(imgs))
            d_loss.backward()
            optimizer_D.step()

            # Weight Clipping for WGAN
            for p in discriminator.parameters():
                p.data.clamp_(-0.01, 0.01)

        writer.add_scalar("Loss/Generator", g_loss.item(), epoch)
        writer.add_scalar("Loss/Discriminator", d_loss.item(), epoch)

        with torch.no_grad():
            z = torch.randn(64, 100).to(device)
            gen_imgs = generator(z)
            writer.add_images('Generated Images', gen_imgs, epoch)
            vutils.save_image(gen_imgs, f'generated_epoch_{epoch}.png', normalize=True)
```

```
def train_wgan_gp():
    lambda_gp = 10
    for epoch in range(50):
        for i, (imgs, _) in enumerate(dataloader):
            imgs = imgs.to(device)
            batch_size = imgs.size(0)

            # Train Generator
            optimizer_G.zero_grad()
            z = torch.randn(batch_size, 100).to(device)
            gen_imgs = generator(z)
            g_loss = -torch.mean(discriminator(gen_imgs))
            g_loss.backward()
            optimizer_G.step()

            # Train Discriminator
            optimizer_D.zero_grad()
            alpha = torch.rand(batch_size, 1, 1, 1).to(device)
            interpolates = (alpha * imgs + (1 - alpha) * gen_imgs.detach()).requires_grad_(True)
            d_interpolates = discriminator(interpolates)
            gradients = torch.autograd.grad(outputs=d_interpolates, inputs=interpolates,
                                            grad_outputs=torch.ones_like(d_interpolates),
                                            create_graph=True, retain_graph=True)[0]
            gradient_penalty = lambda_gp * ((gradients.norm(2, dim=1) - 1) ** 2).mean()
            d_loss = torch.mean(discriminator(gen_imgs.detach())) - torch.mean(discriminator(imgs)) + gradient_penal
            d_loss.backward()
            optimizer_D.step()

        writer.add_scalar("Loss/Generator", g_loss.item(), epoch)
        writer.add_scalar("Loss/Discriminator", d_loss.item(), epoch)

        with torch.no_grad():
            z = torch.randn(64, 100).to(device)
            gen_imgs = generator(z)
            writer.add_images('Generated Images', gen_imgs, epoch)
            vutils.save_image(gen_imgs, f'generated_epoch_{epoch}.png', normalize=True)
```

```
train_lsgan()
```

```
import os

log_dir = "C:/Users/jmdgo/Downloads/lsgan-epochs"
print("Log directory contents:", os.listdir(log_dir))
```

```
Log directory contents: ['events.out.tfevents.1743177778.5d1b67b5700f.31.0', 'generated_epoch_0.png', 'generated_epoch_1.png',
```

```
%load_ext tensorboard
```

```
%tensorboard --logdir "C:/Users/jmdgo/Downloads/lsgan-epochs"
```

image.png

image.png

image.png

```
train_wgan()
```

```
log_dir = "C:/Users/jmdgo/Downloads/wgan-epochs"
print("Log directory contents:", os.listdir(log_dir))
```

```
Log directory contents: ['events.out.tfevents.1743181093.DESKTOP-T6TE1HQ.420.0', 'gan-assignment-4.ipynb', 'generated_epoch_0.
```

```
%tensorboard --logdir "C:/Users/jmdgo/Downloads/wgan-epochs"
```

image.png

image.png

image.png

```
train_wgan_gp()
```

```
log_dir = "C:/Users/jmdgo/Downloads/wgan-gp-epochs"
print("Log directory contents:", os.listdir(log_dir))
```

```
Log directory contents: ['events.out.tfevents.1743181093.DESKTOP-T6TE1HQ.420.0', 'generated_epoch_0.png', 'generated_epoch_1.p
```

```
%tensorboard --logdir "C:/Users/jmdgo/Downloads/wgan-gp-epochs"
```

image.png

image.png

image.png

```
writer.close()
```