```
from torch.utils.data import DataLoader,Dataset
import torch
import torchvision.transforms as transforms
from torchvision import datasets
import torchvision.models as models
import torchvision
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda
```

```
train_data = datasets.CIFAR10(root='./data',train=True,download=True,transform=
test_data = datasets.CIFAR10(root='./data',train=False,download=True,transform=
train_loader = DataLoader(train_data,batch_size=64,shuffle=True)
test_loader = DataLoader(test_data,batch_size=64,shuffle=True)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

```
class Generator(torch.nn.Module):
    def __init__(self):
        super(Generator,self).__init__()
        self.model = torch.nn.Sequential(
              torch.nn.Linear(100,192),
              torch.nn.LeakyReLU(0.2, inplace=True),
              torch.nn.Unflatten(1,(3,8,8)),
              torch.nn.ConvTranspose2d(3,32,kernel_size=4,stride=2,padding=1),
              torch.nn.BatchNorm2d(32),
              torch.nn.LeakyReLU(0.2, inplace=True),
              torch.nn.ConvTranspose2d(32,64,kernel_size=4,stride=2,padding=1),
              torch.nn.BatchNorm2d(64),
              torch.nn.LeakyReLU(0.2, inplace=True),
              torch.nn.ConvTranspose2d(64,3,kernel_size=5,stride=1,padding=2),
              torch.nn.Tanh(),
          )


    def forward(self,x):
        return self.model(x)

class Discriminator(torch.nn.Module):
    def __init__(self):
        super(Discriminator,self).__init__()
        self.model = torch.nn.Sequential(
              torch.nn.Conv2d(3,32,kernel_size=4,stride=2,padding=1),
              torch.nn.LeakyReLU(0.2, inplace=True),
              torch.nn.Conv2d(32,64,kernel_size=4,stride=2,padding=1),
```

```
            torch.nn.BatchNorm2d(64),
            torch.nn.LeakyReLU(0.2, inplace=True),
            torch.nn.Flatten(),
            torch.nn.Linear(64*8*8,1),
            torch.nn.Sigmoid()
        )

    def forward(self,x):
      return self.model(x)



generator = Generator().to(device)
discriminator = Discriminator().to(device)
gen_optim = torch.optim.Adam(generator.parameters(),lr=0.0002)
dis_optim = torch.optim.Adam(discriminator.parameters(),lr=0.0002)
loss = torch.nn.BCELoss()
epochs = 50


for epoch in range(epochs):
    running_gen_loss = 0.0
    running_dis_loss = 0.0
    for (real_batch,_) in tqdm(train_loader):
        real_data = real_batch.to(device)
        real_label = torch.ones(real_data.size(0),1).to(device)
        fake_label = torch.zeros(real_data.size(0),1).to(device)
        noise = torch.randn(real_data.size(0),100).to(device)
        fake_data = generator(noise)

        # train discrimator
        discriminator.zero_grad()
        real_predict = discriminator(real_data)
        real_loss = loss(real_predict,real_label)
        fake_predict = discriminator(fake_data.detach())
        fake_loss = loss(fake_predict,fake_label)
        d_loss = real_loss + fake_loss
        d_loss.backward()
        dis_optim.step()


        generator.zero_grad()
        predict = discriminator(fake_data)
        g_loss = loss(predict,real_label)
        g_loss.backward()
        gen_optim.step()
        running_gen_loss += g_loss.item()
        running_dis_loss += d_loss.item()
    print(f"Epoch {epoch+1}/{epochs}, Discriminator Loss: {running_dis_loss}, G
```

```
100%|██████████| 782/782 [00:14<00:00, 55.07it/s]
Epoch 1/50, Discriminator Loss: 79.34735544864088, Generator Loss: 3498.272
100%|██████████| 782/782 [00:14<00:00, 55.64it/s]
```

```
Epoch 2/50, Discriminator Loss: 265.89982818067074, Generator Loss: 2297.77
100%|██████████| 782/782 [00:14<00:00, 53.89it/s]
Epoch 3/50, Discriminator Loss: 212.33174313046038, Generator Loss: 2387.64
100%|██████████| 782/782 [00:14<00:00, 54.67it/s]
Epoch 4/50, Discriminator Loss: 131.56030647084117, Generator Loss: 2853.75
100%|██████████| 782/782 [00:14<00:00, 55.69it/s]
Epoch 5/50, Discriminator Loss: 126.95083673112094, Generator Loss: 3119.74
100%|██████████| 782/782 [00:14<00:00, 55.53it/s]
Epoch 6/50, Discriminator Loss: 144.1408057268709, Generator Loss: 2926.517
100%|██████████| 782/782 [00:14<00:00, 55.55it/s]
Epoch 7/50, Discriminator Loss: 141.14397859945893, Generator Loss: 2927.65
100%|██████████| 782/782 [00:14<00:00, 55.62it/s]
Epoch 8/50, Discriminator Loss: 118.87103569274768, Generator Loss: 3286.87
100%|██████████| 782/782 [00:14<00:00, 54.64it/s]
Epoch 9/50, Discriminator Loss: 134.0993997938931, Generator Loss: 3191.206
100%|██████████| 782/782 [00:14<00:00, 54.33it/s]
Epoch 10/50, Discriminator Loss: 149.42568738525733, Generator Loss: 3325.5
100%|██████████| 782/782 [00:14<00:00, 55.15it/s]
Epoch 11/50, Discriminator Loss: 131.2643554098904, Generator Loss: 3400.28
100%|██████████| 782/782 [00:14<00:00, 55.06it/s]
Epoch 12/50, Discriminator Loss: 129.6974500133656, Generator Loss: 3293.84
100%|██████████| 782/782 [00:14<00:00, 55.40it/s]
Epoch 13/50, Discriminator Loss: 135.2629120964557, Generator Loss: 3223.02
100%|██████████| 782/782 [00:14<00:00, 55.59it/s]
Epoch 14/50, Discriminator Loss: 140.99911763891578, Generator Loss: 3557.9
100%|██████████| 782/782 [00:14<00:00, 55.62it/s]
Epoch 15/50, Discriminator Loss: 150.6282909400761, Generator Loss: 3238.09
100%|██████████| 782/782 [00:14<00:00, 53.91it/s]
Epoch 16/50, Discriminator Loss: 146.5069995643571, Generator Loss: 3474.39
100%|██████████| 782/782 [00:14<00:00, 55.64it/s]
Epoch 17/50, Discriminator Loss: 143.24071638472378, Generator Loss: 3068.7
100%|██████████| 782/782 [00:14<00:00, 53.67it/s]
Epoch 18/50, Discriminator Loss: 147.0724831968546, Generator Loss: 3280.58
100%|██████████| 782/782 [00:34<00:00, 22.72it/s]
Epoch 19/50, Discriminator Loss: 122.43681250081863, Generator Loss: 3649.2
100%|██████████| 782/782 [00:14<00:00, 55.38it/s]
Epoch 20/50, Discriminator Loss: 181.5030162539333, Generator Loss: 3193.81
100%|██████████| 782/782 [00:14<00:00, 55.33it/s]
Epoch 21/50, Discriminator Loss: 104.07328116567805, Generator Loss: 3907.5
100%|██████████| 782/782 [00:14<00:00, 55.57it/s]
Epoch 22/50, Discriminator Loss: 174.1619120799005, Generator Loss: 3109.25
100%|██████████| 782/782 [00:14<00:00, 54.20it/s]
Epoch 23/50, Discriminator Loss: 181.06164005957544, Generator Loss: 3191.7
100%|██████████| 782/782 [00:14<00:00, 54.62it/s]
Epoch 24/50, Discriminator Loss: 220.4662074521184, Generator Loss: 3074.73
100%|██████████| 782/782 [00:14<00:00, 55.83it/s]
Epoch 25/50, Discriminator Loss: 226.19803060870618, Generator Loss: 3002.2
100%|██████████| 782/782 [00:14<00:00, 55.53it/s]
Epoch 26/50, Discriminator Loss: 195.27199198305607, Generator Loss: 3132.8
100%|██████████| 782/782 [00:14<00:00, 55.45it/s]
Epoch 27/50, Discriminator Loss: 224.12189589627087, Generator Loss: 2801.1
100%|██████████| 782/782 [00:14<00:00, 55.52it/s]
Epoch 28/50, Discriminator Loss: 173.87523538433015, Generator Loss: 3211.0
100%|██████████| 782/782 [00:14<00:00, 54.95it/s]
Epoch 29/50, Discriminator Loss: 155.3940852675587, Generator Loss: 3389.16
```

```
with torch.no_grad():
    z = torch.randn(16, 100).to(device)
    samples = generator(z).cpu()
```

```
    samples = generator(z).cpu()

fig, axes = plt.subplots(4, 4, figsize=(8, 8))
for i, ax in enumerate(axes.flatten()):
    ax.imshow(samples[i].permute(1, 2, 0))
    ax.axis("off")
plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
```