Introduction to Deep Learning (CS474)

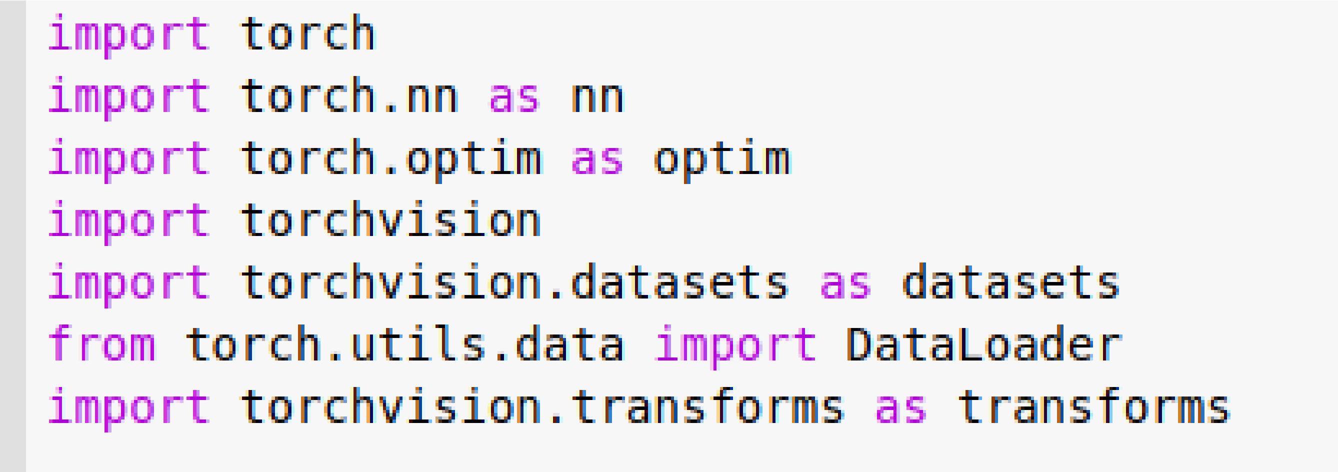
Lecture 29

Outline

Module 4

- Generative Adversarial Networks (GANs) in PyTorch

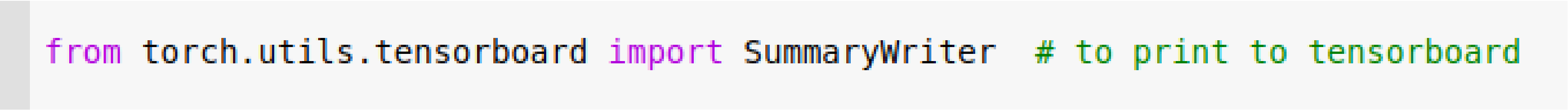
Example



```python
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.datasets as datasets
from torch.utils.data import DataLoader
import torchvision.transforms as transforms
```

Slide Credit: A. Persson

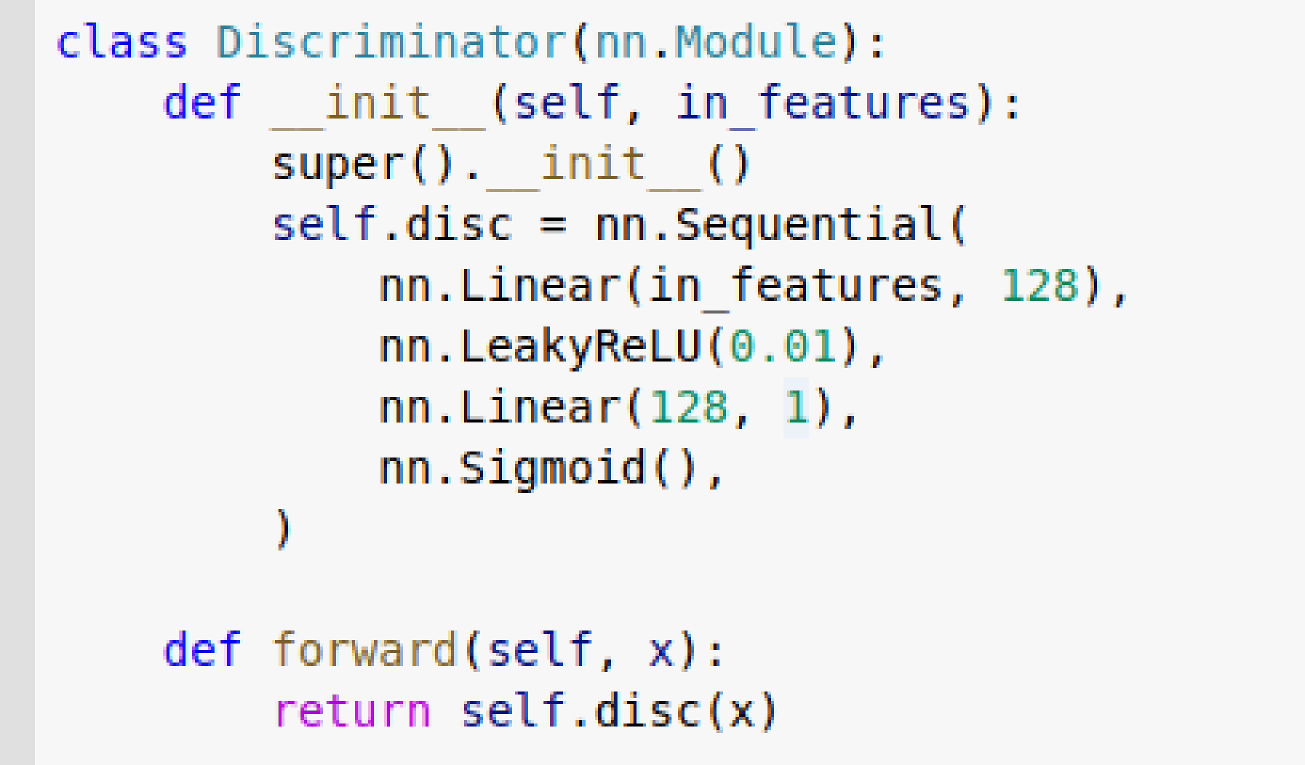
Example (contd.)



```
from torch.utils.tensorboard import SummaryWriter  # to print to tensorboard
```

Slide Credit: A. Persson

Example (contd.)

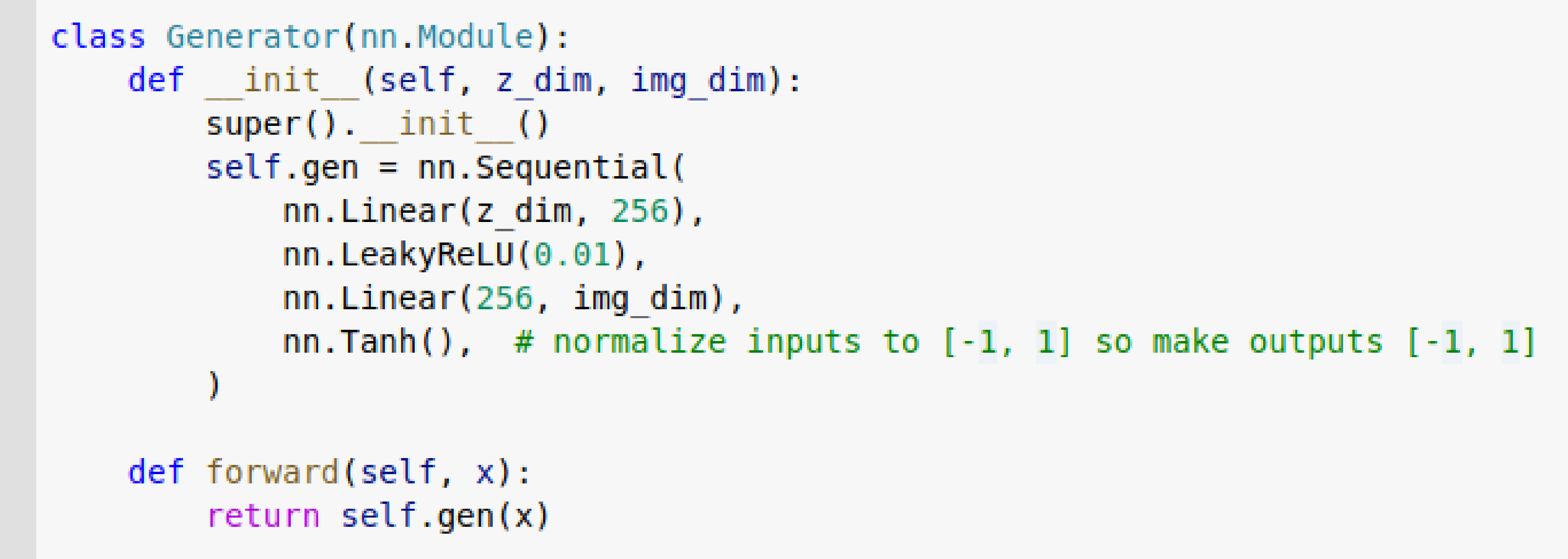


```python
class Discriminator(nn.Module):
    def __init__(self, in_features):
        super().__init__()
        self.disc = nn.Sequential(
            nn.Linear(in_features, 128),
            nn.LeakyReLU(0.01),
            nn.Linear(128, 1),
            nn.Sigmoid(),
        )

    def forward(self, x):
        return self.disc(x)
```

Slide Credit: A. Persson

Example (contd.)

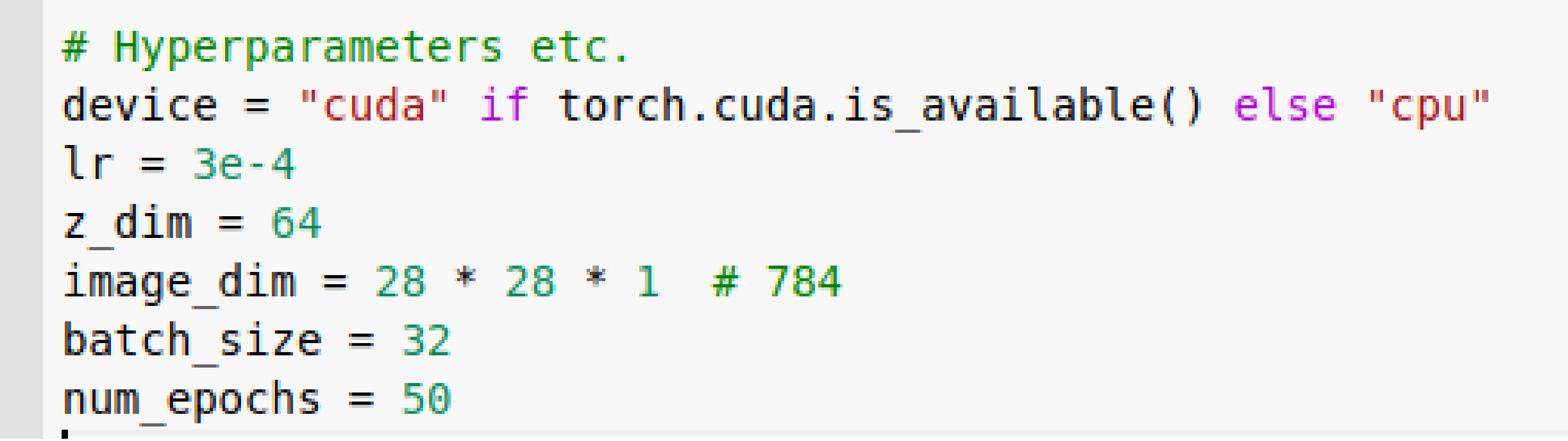


```
class Generator(nn.Module):
    def __init__(self, z_dim, img_dim):
        super().__init__()
        self.gen = nn.Sequential(
            nn.Linear(z_dim, 256),
            nn.LeakyReLU(0.01),
            nn.Linear(256, img_dim),
            nn.Tanh(),  # normalize inputs to [-1, 1] so make outputs [-1, 1]
        )

    def forward(self, x):
        return self.gen(x)
```

Slide Credit: A. Persson

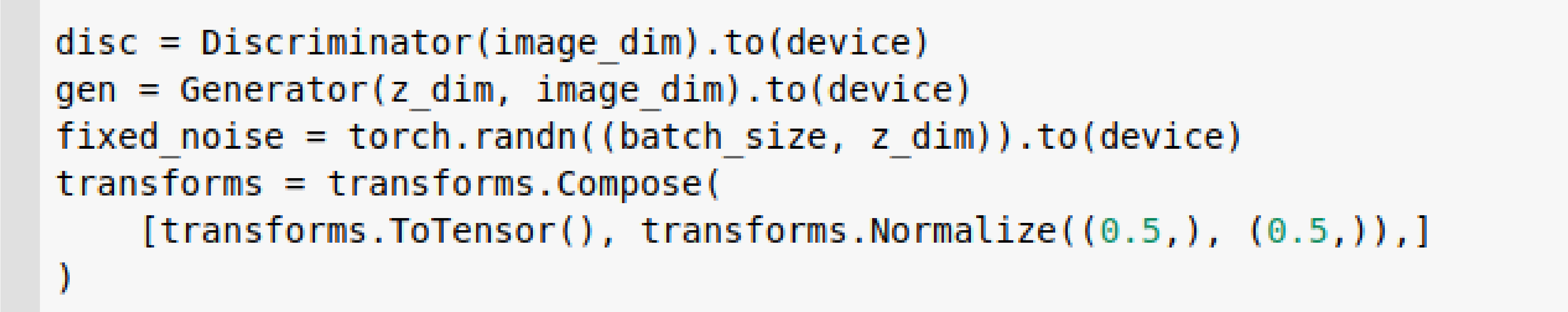
Example (contd.)



```
# Hyperparameters etc.
device = "cuda" if torch.cuda.is_available() else "cpu"
lr = 3e-4
z_dim = 64
image_dim = 28 * 28 * 1  # 784
batch_size = 32
num_epochs = 50
```

Slide Credit: A. Persson

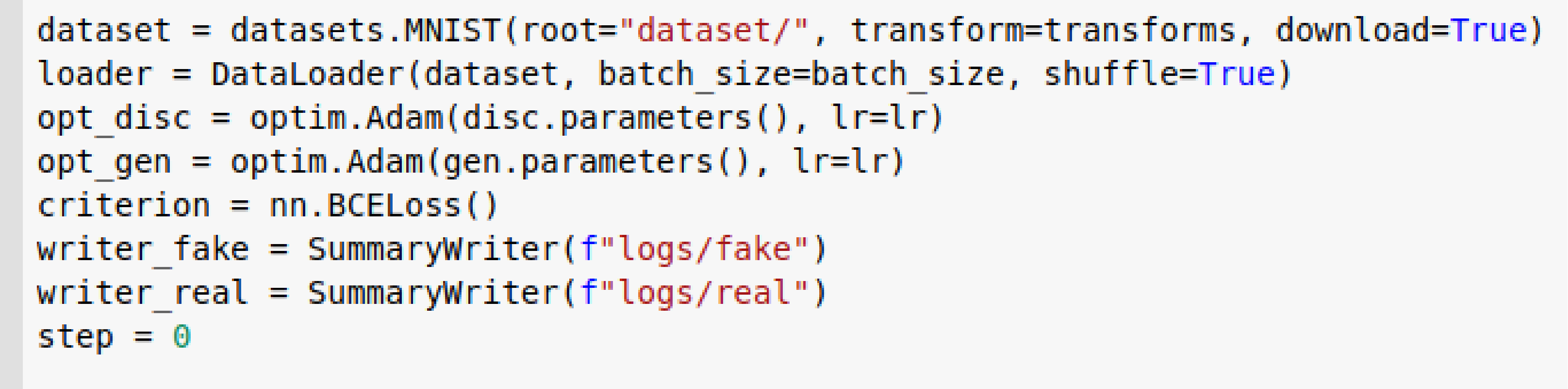
Example (contd.)



```
disc = Discriminator(image_dim).to(device)
gen = Generator(z_dim, image_dim).to(device)
fixed_noise = torch.randn((batch_size, z_dim)).to(device)
transforms = transforms.Compose(
    [transforms.ToTensor(), transforms.Normalize((0.5,), (0.5,)),]
)
```

Slide Credit: A. Persson

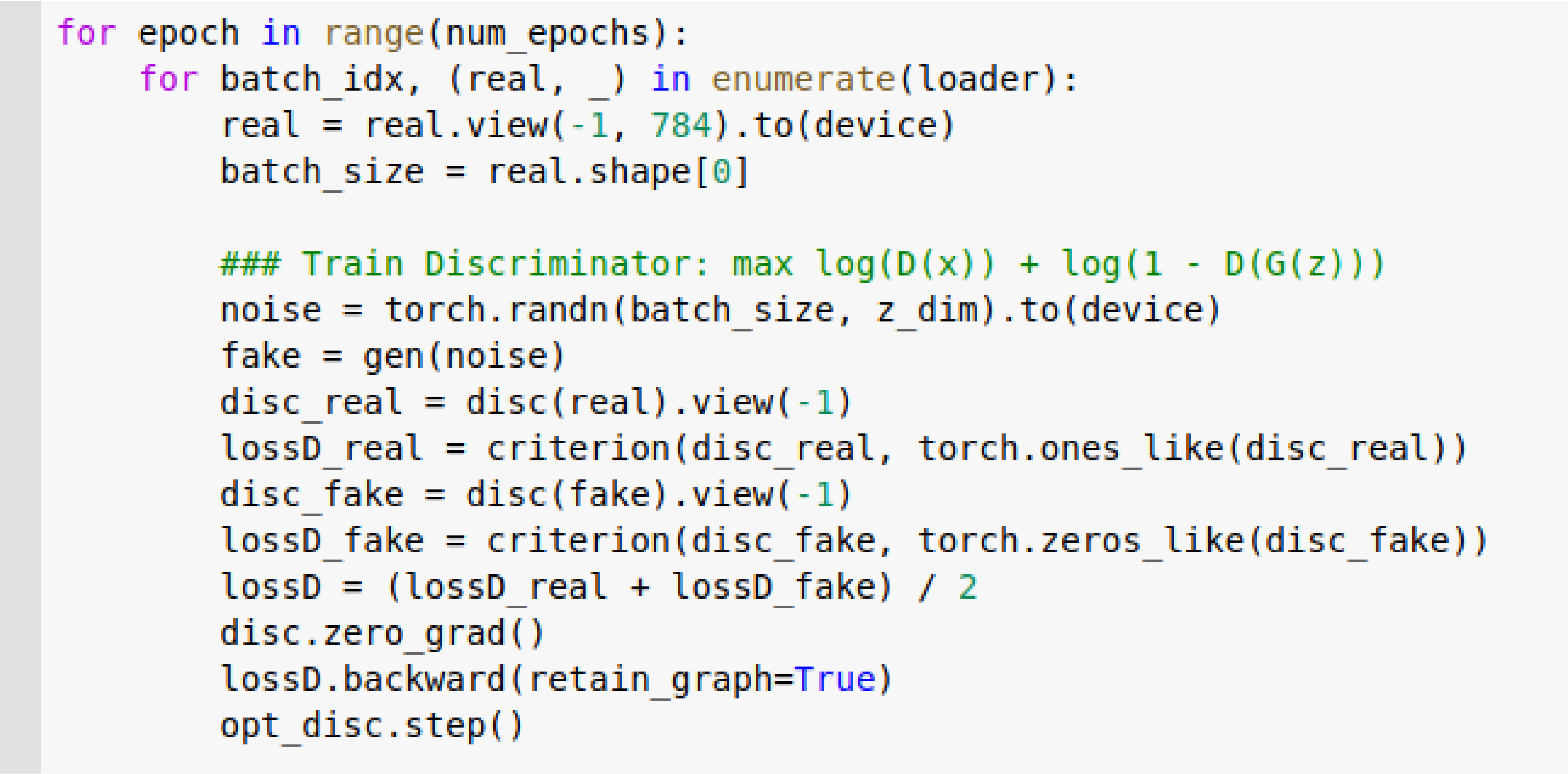
Example (contd.)



```python
dataset = datasets.MNIST(root="dataset/", transform=transforms, download=True)
loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)
opt_disc = optim.Adam(disc.parameters(), lr=lr)
opt_gen = optim.Adam(gen.parameters(), lr=lr)
criterion = nn.BCELoss()
writer_fake = SummaryWriter(f"logs/fake")
writer_real = SummaryWriter(f"logs/real")
step = 0
```

Slide Credit: A. Persson

Example (contd.)

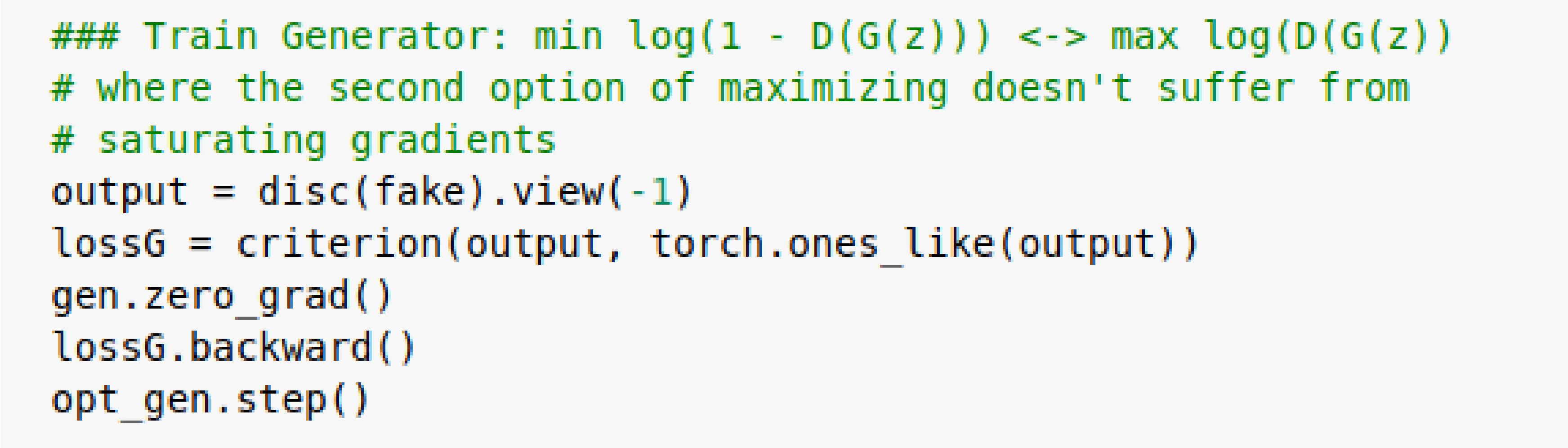


```python
for epoch in range(num_epochs):
    for batch_idx, (real, _) in enumerate(loader):
        real = real.view(-1, 784).to(device)
        batch_size = real.shape[0]

        ### Train Discriminator: max log(D(x)) + log(1 - D(G(z)))
        noise = torch.randn(batch_size, z_dim).to(device)
        fake = gen(noise)
        disc_real = disc(real).view(-1)
        lossD_real = criterion(disc_real, torch.ones_like(disc_real))
        disc_fake = disc(fake).view(-1)
        lossD_fake = criterion(disc_fake, torch.zeros_like(disc_fake))
        lossD = (lossD_real + lossD_fake) / 2
        disc.zero_grad()
        lossD.backward(retain_graph=True)
        opt_disc.step()
```

Slide Credit: A. Persson

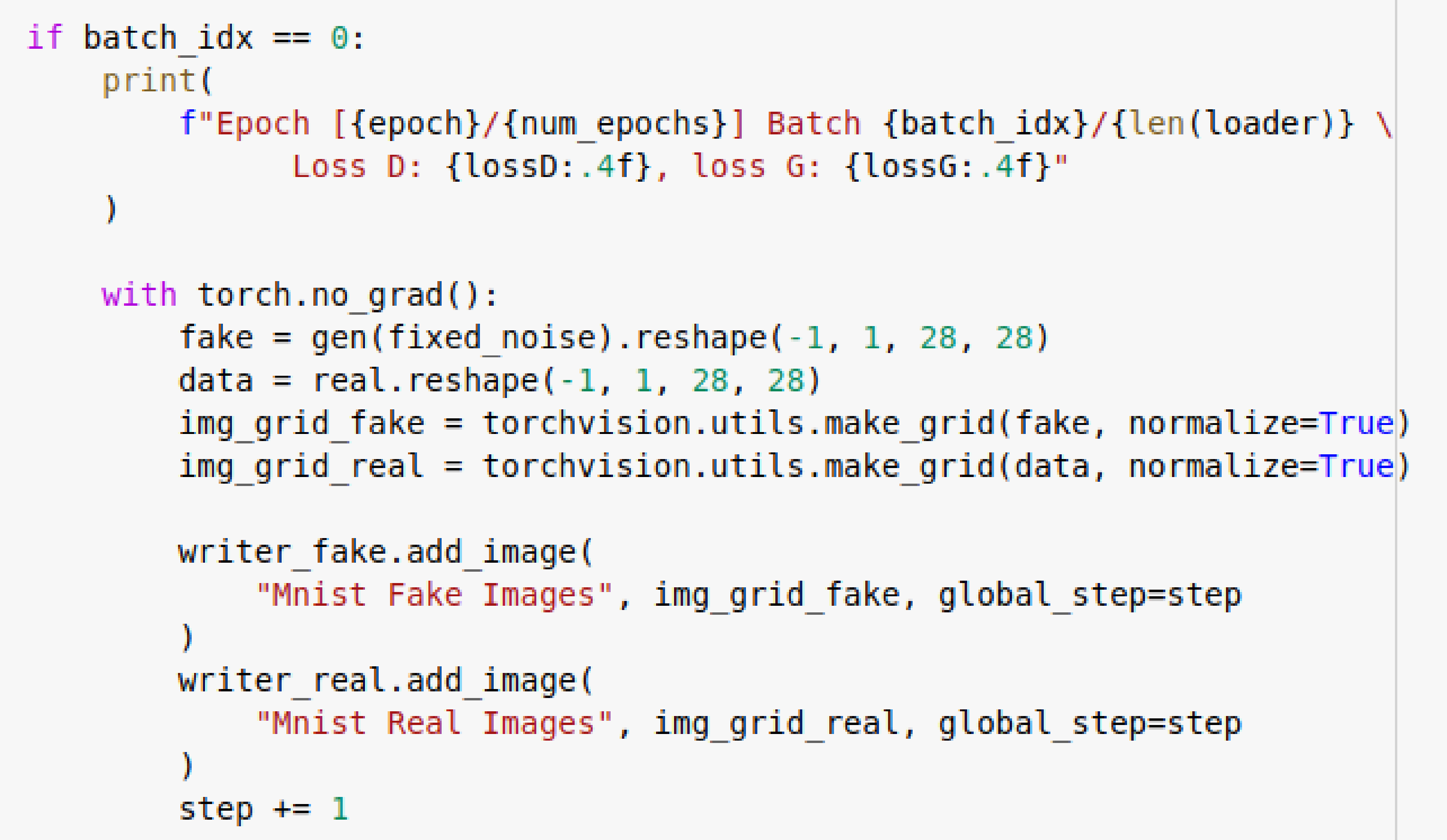
Example (contd.)



```
### Train Generator: min log(1 - D(G(z))) <-> max log(D(G(z))
# where the second option of maximizing doesn't suffer from
# saturating gradients
output = disc(fake).view(-1)
lossG = criterion(output, torch.ones_like(output))
gen.zero_grad()
lossG.backward()
opt_gen.step()
```

Slide Credit: A. Persson

Example (contd.)



```
if batch_idx == 0:
    print(
        f"Epoch [{epoch}/{num_epochs}] Batch {batch_idx}/{len(loader)} \
              Loss D: {lossD:.4f}, loss G: {lossG:.4f}"
    )

    with torch.no_grad():
        fake = gen(fixed_noise).reshape(-1, 1, 28, 28)
        data = real.reshape(-1, 1, 28, 28)
        img_grid_fake = torchvision.utils.make_grid(fake, normalize=True)
        img_grid_real = torchvision.utils.make_grid(data, normalize=True)

        writer_fake.add_image(
            "Mnist Fake Images", img_grid_fake, global_step=step
        )
        writer_real.add_image(
            "Mnist Real Images", img_grid_real, global_step=step
        )
        step += 1
```

Slide Credit: A. Persson

References

- All the contents present in the slides are taken from various online resources. Due credit is given in the respective slides.These slides are used for *academic* purposes only.