```
!ls
```

```
sample_data
```

```
from google.colab import drive
drive.mount('/content/drive')

import os

SAVE_DIR = "/content/drive/MyDrive/VAE_MNIST_Results"

os.makedirs(SAVE_DIR, exist_ok=True)
os.makedirs(SAVE_DIR + "/checkpoints", exist_ok=True)
os.makedirs(SAVE_DIR + "/generated", exist_ok=True)
os.makedirs(SAVE_DIR + "/recon", exist_ok=True)

print("Folders created in Drive!")
```

```
Mounted at /content/drive
Folders created in Drive!
```

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
import matplotlib.pyplot as plt
```

```
transform = transforms.Compose([
    transforms.ToTensor()
])

train_dataset = datasets.MNIST(
    root="./data",
    train=True,
    download=True,
    transform=transform
)

train_loader = DataLoader(train_dataset, batch_size=128, shuffle=True)

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

```
100%|██████████| 9.91M/9.91M [00:00<00:00, 17.9MB/s]
100%|██████████| 28.9k/28.9k [00:00<00:00, 483kB/s]
100%|██████████| 1.65M/1.65M [00:00<00:00, 4.47MB/s]
100%|██████████| 4.54k/4.54k [00:00<00:00, 16.2MB/s]Using device: cuda
```

```
latent_dim = 20   # perfect for MNIST

class VAE(nn.Module):
    def __init__(self):
        super(VAE, self).__init__()

        # Encoder
        self.encoder = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.ReLU()
        )

        self.fc_mu = nn.Linear(256, latent_dim)
        self.fc_logvar = nn.Linear(256, latent_dim)

        # Decoder
```

```
            self.decoder_fc = nn.Sequential(
                nn.Linear(latent_dim, 256),
                nn.ReLU(),
                nn.Linear(256, 512),
                nn.ReLU(),
                nn.Linear(512, 28*28),
                nn.Sigmoid()
            )

        def reparameterize(self, mu, logvar):
            std = torch.exp(0.5 * logvar)
            eps = torch.randn_like(std)
            return mu + eps * std

        def forward(self, x):
            x = x.view(-1, 28*28)
            h = self.encoder(x)

            mu = self.fc_mu(h)
            logvar = self.fc_logvar(h)

            z = self.reparameterize(mu, logvar)

            out = self.decoder_fc(z)
            out = out.view(-1, 1, 28, 28)

            return out, mu, logvar
```

```
    def vae_loss(recon, x, mu, logvar):
        recon_loss = F.binary_cross_entropy(recon, x, reduction="sum")
        kl_loss = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
        return recon_loss + kl_loss
```

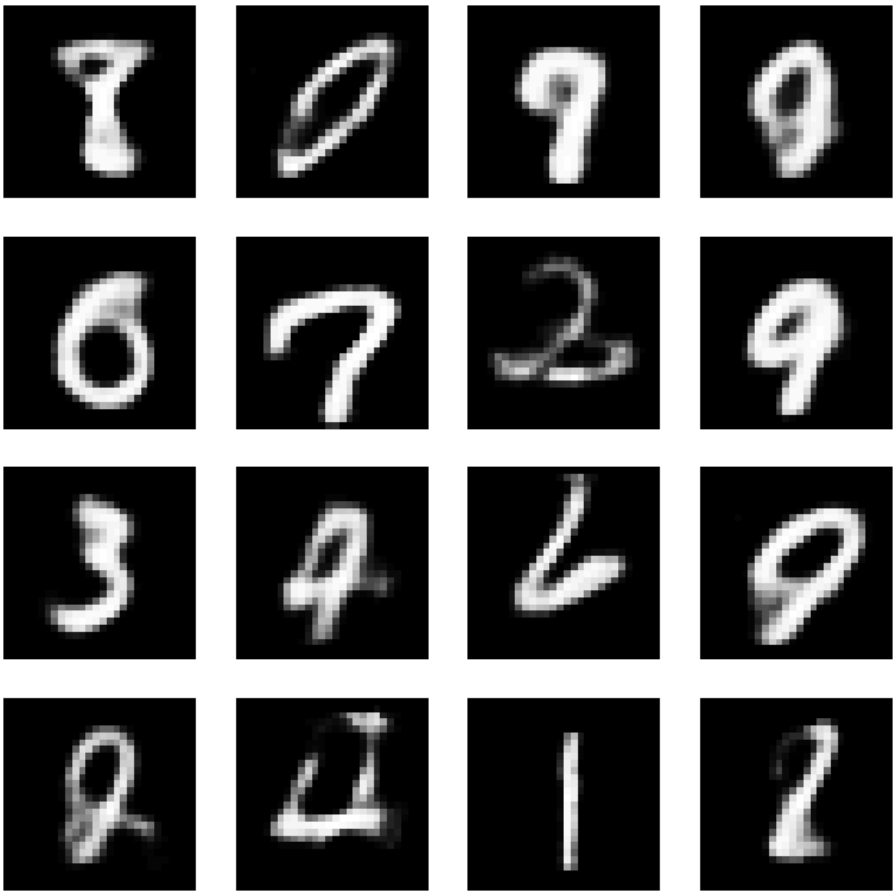
```
    model = VAE().to(device)
    optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

    epochs = 50   # MNIST converges beautifully in 40–60 epochs
    losses = []
```

```
    for epoch in range(epochs):
        model.train()
        total_loss = 0

        for imgs, _ in train_loader:
            imgs = imgs.to(device)

            recon, mu, logvar = model(imgs)
            loss = vae_loss(recon, imgs, mu, logvar)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            total_loss += loss.item()

        avg_loss = total_loss / len(train_dataset)
        losses.append(avg_loss)

        print(f"Epoch {epoch+1}/{epochs} | Loss: {avg_loss:.2f}")

        # Save checkpoint
        torch.save(model.state_dict(),
                   f"{SAVE_DIR}/checkpoints/vae_epoch_{epoch+1}.pth")

        # Save reconstruction images
        with torch.no_grad():
            sample_imgs = recon[:8].detach().cpu()
            original_imgs = imgs[:8].detach().cpu()

        plt.figure(figsize=(8,4))
```

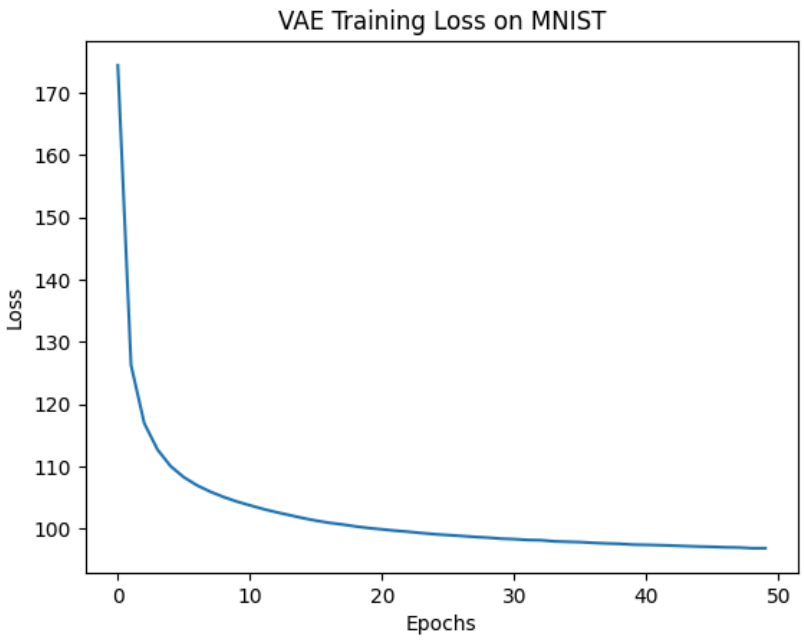
```
    for i in range(8):
        plt.subplot(2,8,i+1)
        plt.imshow(original_imgs[i][0], cmap="gray")
        plt.axis("off")

        plt.subplot(2,8,i+9)
        plt.imshow(sample_imgs[i][0], cmap="gray")
        plt.axis("off")

    plt.savefig(f"{SAVE_DIR}/recon/recon_epoch_{epoch+1}.png")
    plt.close()
```

```
Epoch 1/50 | Loss: 174.38
Epoch 2/50 | Loss: 126.26
Epoch 3/50 | Loss: 116.92
Epoch 4/50 | Loss: 112.69
Epoch 5/50 | Loss: 110.01
Epoch 6/50 | Loss: 108.24
Epoch 7/50 | Loss: 106.94
Epoch 8/50 | Loss: 105.92
Epoch 9/50 | Loss: 105.08
Epoch 10/50 | Loss: 104.35
Epoch 11/50 | Loss: 103.74
Epoch 12/50 | Loss: 103.15
Epoch 13/50 | Loss: 102.63
Epoch 14/50 | Loss: 102.16
Epoch 15/50 | Loss: 101.69
Epoch 16/50 | Loss: 101.28
Epoch 17/50 | Loss: 100.93
Epoch 18/50 | Loss: 100.66
Epoch 19/50 | Loss: 100.35
Epoch 20/50 | Loss: 100.08
Epoch 21/50 | Loss: 99.88
Epoch 22/50 | Loss: 99.67
Epoch 23/50 | Loss: 99.49
Epoch 24/50 | Loss: 99.29
Epoch 25/50 | Loss: 99.10
Epoch 26/50 | Loss: 98.96
Epoch 27/50 | Loss: 98.82
Epoch 28/50 | Loss: 98.67
Epoch 29/50 | Loss: 98.55
Epoch 30/50 | Loss: 98.39
Epoch 31/50 | Loss: 98.31
Epoch 32/50 | Loss: 98.18
Epoch 33/50 | Loss: 98.14
Epoch 34/50 | Loss: 97.96
Epoch 35/50 | Loss: 97.89
Epoch 36/50 | Loss: 97.83
Epoch 37/50 | Loss: 97.71
Epoch 38/50 | Loss: 97.62
Epoch 39/50 | Loss: 97.56
Epoch 40/50 | Loss: 97.43
Epoch 41/50 | Loss: 97.39
Epoch 42/50 | Loss: 97.34
Epoch 43/50 | Loss: 97.27
Epoch 44/50 | Loss: 97.19
Epoch 45/50 | Loss: 97.12
Epoch 46/50 | Loss: 97.07
Epoch 47/50 | Loss: 97.00
Epoch 48/50 | Loss: 96.97
Epoch 49/50 | Loss: 96.84
Epoch 50/50 | Loss: 96.85
```

```
model.eval()

with torch.no_grad():
    z = torch.randn(16, latent_dim).to(device)
    generated = model.decoder_fc(z).view(-1,1,28,28)

plt.figure(figsize=(8,8))
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.imshow(generated[i][0].cpu(), cmap="gray")
    plt.axis("off")

plt.savefig(f"{SAVE_DIR}/generated/generated_digits.png")
plt.show()
```



```
plt.plot(losses)
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("VAE Training Loss on MNIST")
plt.savefig(f"{SAVE_DIR}/loss_curve.png")
plt.show()
```




```
model = VAE().to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)  # smaller LR for fine tuning

checkpoint_path = f"{SAVE_DIR}/checkpoints/vae_epoch_50.pth"
model.load_state_dict(torch.load(checkpoint_path))

model.train()
```

```
print("Loaded checkpoint from epoch 50, resuming training...")
```

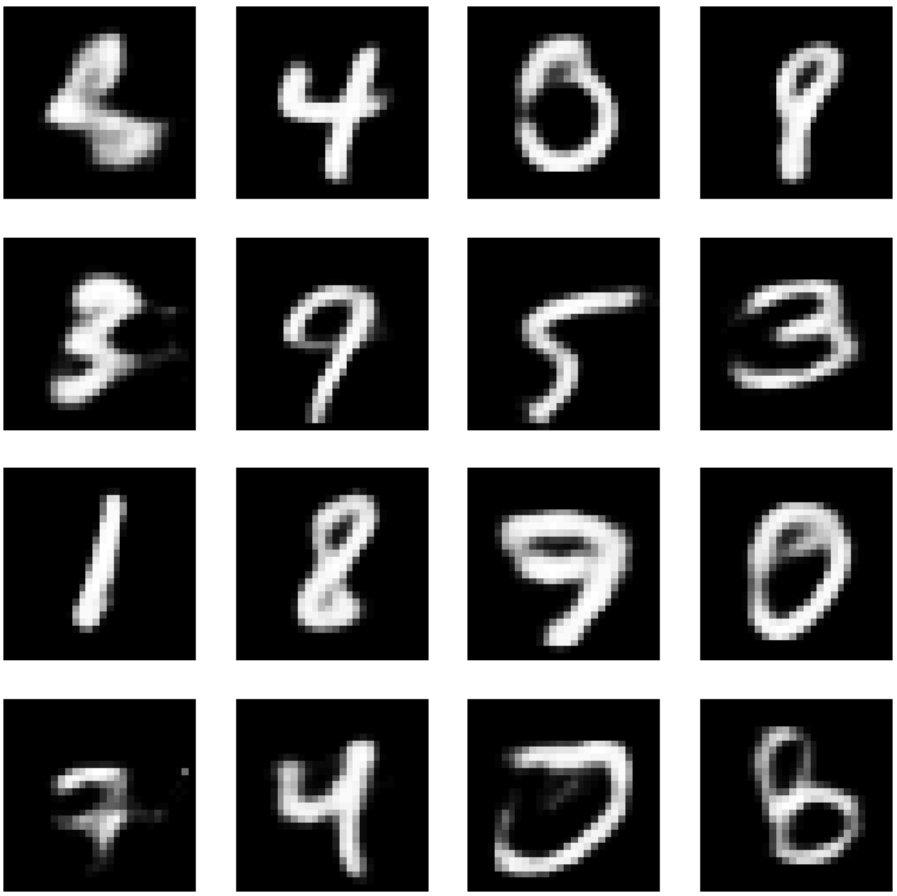
```
Loaded checkpoint from epoch 50, resuming training...
```

```
more_epochs = 50
start_epoch = 50   # last completed epoch

for epoch in range(more_epochs):
    total_loss = 0
    model.train()

    for imgs, _ in train_loader:
        imgs = imgs.to(device)

        recon, mu, logvar = model(imgs)
        loss = vae_loss(recon, imgs, mu, logvar)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    avg_loss = total_loss / len(train_dataset)
    current_epoch = start_epoch + epoch + 1

    print(f"Epoch {current_epoch}/100 | Loss: {avg_loss:.2f}")

    # Save checkpoint
    torch.save(model.state_dict(),
               f"{SAVE_DIR}/checkpoints/vae_epoch_{current_epoch}.pth")

    # Save reconstruction images (original vs reconstructed)
    with torch.no_grad():
        sample_recon = recon[:8].detach().cpu()
        sample_orig = imgs[:8].detach().cpu()

    plt.figure(figsize=(8,4))
    for i in range(8):
        # original
        plt.subplot(2,8,i+1)
        plt.imshow(sample_orig[i][0], cmap="gray")
        plt.axis("off")

        # reconstructed
        plt.subplot(2,8,i+9)
        plt.imshow(sample_recon[i][0], cmap="gray")
        plt.axis("off")

    plt.savefig(f"{SAVE_DIR}/recon/recon_epoch_{current_epoch}.png")
    plt.close()
```

```
Epoch 51/100 | Loss: 94.98
Epoch 52/100 | Loss: 94.67
Epoch 53/100 | Loss: 94.58
Epoch 54/100 | Loss: 94.54
Epoch 55/100 | Loss: 94.48
Epoch 56/100 | Loss: 94.43
Epoch 57/100 | Loss: 94.42
Epoch 58/100 | Loss: 94.38
Epoch 59/100 | Loss: 94.36
Epoch 60/100 | Loss: 94.34
Epoch 61/100 | Loss: 94.36
Epoch 62/100 | Loss: 94.32
Epoch 63/100 | Loss: 94.32
Epoch 64/100 | Loss: 94.30
Epoch 65/100 | Loss: 94.27
Epoch 66/100 | Loss: 94.25
Epoch 67/100 | Loss: 94.25
Epoch 68/100 | Loss: 94.21
Epoch 69/100 | Loss: 94.18
Epoch 70/100 | Loss: 94.21
Epoch 71/100 | Loss: 94.19
Epoch 72/100 | Loss: 94.20
Epoch 73/100 | Loss: 94.15
Epoch 74/100 | Loss: 94.14
```

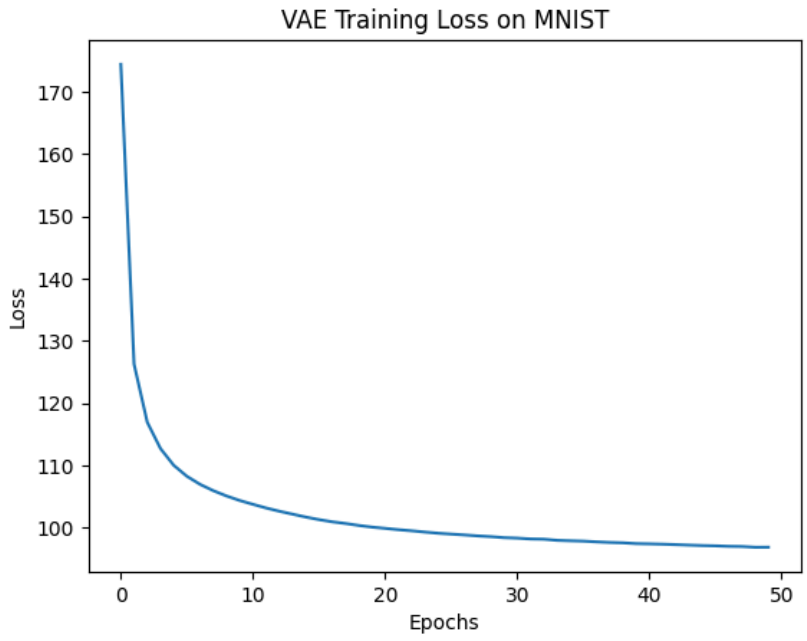
```
Epoch 75/100 | Loss: 94.12
Epoch 76/100 | Loss: 94.14
Epoch 77/100 | Loss: 94.13
Epoch 78/100 | Loss: 94.10
Epoch 79/100 | Loss: 94.09
Epoch 80/100 | Loss: 94.09
Epoch 81/100 | Loss: 94.07
Epoch 82/100 | Loss: 94.06
Epoch 83/100 | Loss: 94.02
Epoch 84/100 | Loss: 94.06
Epoch 85/100 | Loss: 94.03
Epoch 86/100 | Loss: 94.02
Epoch 87/100 | Loss: 94.02
Epoch 88/100 | Loss: 93.99
Epoch 89/100 | Loss: 94.00
Epoch 90/100 | Loss: 93.99
Epoch 91/100 | Loss: 93.99
Epoch 92/100 | Loss: 93.99
Epoch 93/100 | Loss: 93.97
Epoch 94/100 | Loss: 93.96
Epoch 95/100 | Loss: 93.96
Epoch 96/100 | Loss: 93.93
Epoch 97/100 | Loss: 93.94
Epoch 98/100 | Loss: 93.92
Epoch 99/100 | Loss: 93.93
Epoch 100/100 | Loss: 93.91
```

```
model.eval()

with torch.no_grad():
    z = torch.randn(16, latent_dim).to(device)
    generated = model.decoder_fc(z).view(-1,1,28,28)

plt.figure(figsize=(8,8))
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.imshow(generated[i][0].cpu(), cmap="gray")
    plt.axis("off")

plt.savefig(f"{SAVE_DIR}/generated/generated_digits_epoch100.png")
plt.show()
```



```
imgs, _ = next(iter(train_loader))
imgs = imgs.to(device)

with torch.no_grad():
    recon, mu, logvar = model(imgs)

orig = imgs[:8].cpu()
recon_img = recon[:8].detach().cpu()

plt.figure(figsize=(10,4))
for i in range(8):
    plt.subplot(2,8,i+1)
    plt.imshow(orig[i][0], cmap="gray")
    plt.axis("off")

    plt.subplot(2,8,i+9)
    plt.imshow(recon_img[i][0], cmap="gray")
    plt.axis("off")

plt.show()
```

```
plt.plot(losses)
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("VAE Training Loss on MNIST")
plt.savefig(f"{SAVE_DIR}/loss_curve.png")
plt.show()
```




```
model = VAE().to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)  # small LR for fine tuning

checkpoint_path = f"{SAVE_DIR}/checkpoints/vae_epoch_100.pth"
model.load_state_dict(torch.load(checkpoint_path))
```

```
model.train()
print("Loaded checkpoint from epoch 100. Continuing training...")
```

```
Loaded checkpoint from epoch 100. Continuing training...
```

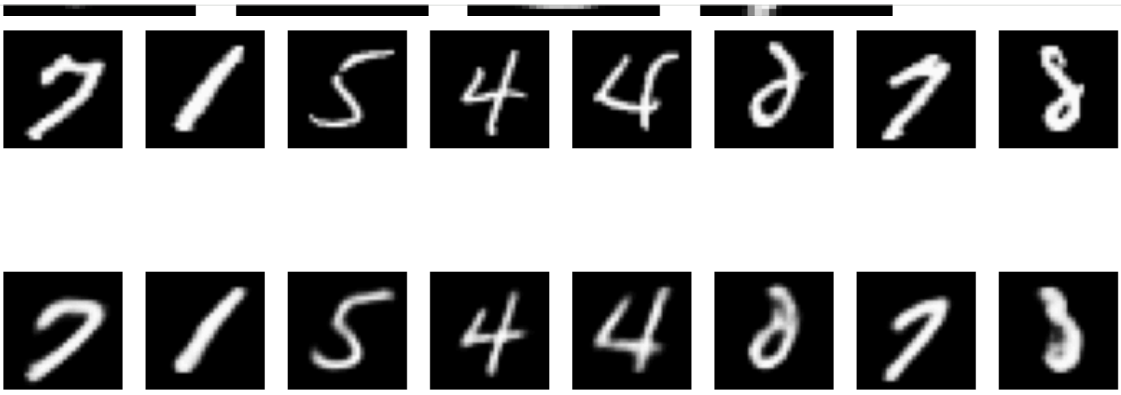
```
more_epochs = 50
start_epoch = 100   # last completed epoch

for epoch in range(more_epochs):
    model.train()
    total_loss = 0

    for imgs, _ in train_loader:
        imgs = imgs.to(device)

        recon, mu, logvar = model(imgs)
        loss = vae_loss(recon, imgs, mu, logvar)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    avg_loss = total_loss / len(train_dataset)
    current_epoch = start_epoch + epoch + 1

    print(f"Epoch {current_epoch}/150 | Loss: {avg_loss:.2f}")

    # Save checkpoint
    torch.save(model.state_dict(),
               f"{SAVE_DIR}/checkpoints/vae_epoch_{current_epoch}.pth")

    # Save reconstruction images (original vs reconstructed)
    with torch.no_grad():
        sample_recon = recon[:8].detach().cpu()
        sample_orig = imgs[:8].detach().cpu()

    import matplotlib.pyplot as plt
    plt.figure(figsize=(8,4))
    for i in range(8):
        # original
        plt.subplot(2,8,i+1)
        plt.imshow(sample_orig[i][0], cmap="gray")
        plt.axis("off")

        # reconstructed
        plt.subplot(2,8,i+9)
        plt.imshow(sample_recon[i][0], cmap="gray")
        plt.axis("off")

    plt.savefig(f"{SAVE_DIR}/recon/recon_epoch_{current_epoch}.png")
    plt.close()
```

```
Epoch 101/150 | Loss: 93.89
Epoch 102/150 | Loss: 93.91
Epoch 103/150 | Loss: 93.90
Epoch 104/150 | Loss: 93.89
Epoch 105/150 | Loss: 93.87
Epoch 106/150 | Loss: 93.87
Epoch 107/150 | Loss: 93.86
Epoch 108/150 | Loss: 93.85
Epoch 109/150 | Loss: 93.84
Epoch 110/150 | Loss: 93.83
Epoch 111/150 | Loss: 93.79
Epoch 112/150 | Loss: 93.82
Epoch 113/150 | Loss: 93.83
Epoch 114/150 | Loss: 93.80
Epoch 115/150 | Loss: 93.80
Epoch 116/150 | Loss: 93.79
Epoch 117/150 | Loss: 93.77
Epoch 118/150 | Loss: 93.79
Epoch 119/150 | Loss: 93.77
Epoch 120/150 | Loss: 93.79
Epoch 121/150 | Loss: 93.74
```

```
Epoch 122/150 | Loss: 93.74
Epoch 123/150 | Loss: 93.75
Epoch 124/150 | Loss: 93.74
Epoch 125/150 | Loss: 93.73
Epoch 126/150 | Loss: 93.72
Epoch 127/150 | Loss: 93.72
Epoch 128/150 | Loss: 93.71
Epoch 129/150 | Loss: 93.71
Epoch 130/150 | Loss: 93.71
Epoch 131/150 | Loss: 93.68
Epoch 132/150 | Loss: 93.69
Epoch 133/150 | Loss: 93.68
Epoch 134/150 | Loss: 93.67
Epoch 135/150 | Loss: 93.65
Epoch 136/150 | Loss: 93.69
Epoch 137/150 | Loss: 93.64
Epoch 138/150 | Loss: 93.65
Epoch 139/150 | Loss: 93.64
Epoch 140/150 | Loss: 93.62
Epoch 141/150 | Loss: 93.64
Epoch 142/150 | Loss: 93.62
Epoch 143/150 | Loss: 93.60
Epoch 144/150 | Loss: 93.59
Epoch 145/150 | Loss: 93.62
Epoch 146/150 | Loss: 93.60
Epoch 147/150 | Loss: 93.61
Epoch 148/150 | Loss: 93.62
Epoch 149/150 | Loss: 93.59
Epoch 150/150 | Loss: 93.58
```

```
model.eval()

with torch.no_grad():
    z = torch.randn(16, latent_dim).to(device)
    generated = model.decoder_fc(z).view(-1,1,28,28)

plt.figure(figsize=(8,8))
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.imshow(generated[i][0].cpu(), cmap="gray")
    plt.axis("off")

plt.savefig(f"{SAVE_DIR}/generated/generated_digits_epoch150.png")
plt.show()
```

```
imgs, _ = next(iter(train_loader))
imgs = imgs.to(device)

with torch.no_grad():
    recon, mu, logvar = model(imgs)

orig = imgs[:8].cpu()
recon_img = recon[:8].detach().cpu()

plt.figure(figsize=(10,4))
for i in range(8):
    plt.subplot(2,8,i+1)
    plt.imshow(orig[i][0], cmap="gray")
    plt.axis("off")

    plt.subplot(2,8,i+9)
    plt.imshow(recon_img[i][0], cmap="gray")
    plt.axis("off")

plt.show()
```



Start coding or generate with AI.