In [1]:

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
```

In [2]:

```
# Load the MNIST dataset
transform = transforms.ToTensor()

train_dataset = torchvision.datasets.FashionMNIST(
    root='./data', train=True, transform=transform, download=True
)

train_loader = torch.utils.data.DataLoader(
    train_dataset, batch_size=128, shuffle=True, num_workers=2
)
```

In [3]:

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

        self.encoder = nn.Sequential(
            nn.Linear(784, 256),
            nn.ReLU(),
            nn.Linear(256, 64),
            nn.ReLU(),
            nn.Linear(64, latent_dim * 2)
        )

        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, 64),
            nn.ReLU(),
            nn.Linear(64, 256),
            nn.ReLU(),
            nn.Linear(256, 784),
            nn.Sigmoid()
        )

    def reparameterize(self, mu, log_var):
```

```
        std = torch.exp(0.5 * log_var)
        eps = torch.randn_like(std)
        return mu + eps * std

    def forward(self, x):
        x = x.view(-1, 784)
        latent = self.encoder(x)
        mu, log_var = latent[:, :latent_dim], latent[:, latent_dim:]
        z = self.reparameterize(mu, log_var)
        reconstructed = self.decoder(z)
        return reconstructed, mu, log_var
```

In [4]:

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
latent_dim = 10  # Dimensionality of the latent space
epochs = 20
batch_size = 128




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

def loss_function(reconstructed, x, mu, log_var):
    reconstruction_loss =
nn.functional.binary_cross_entropy(reconstructed, x.view(-1, 784),
reduction='sum')
    kl_divergence = -0.5 * torch.sum(1 + log_var - mu.pow(2) -
log_var.exp())
    return reconstruction_loss + kl_divergence

def train_epoch(epoch):
    model.train()
    train_loss = 0

    for batch_idx, (data, _) in enumerate(train_loader):
        data = data.to(device)

        optimizer.zero_grad()
        reconstructed, mu, log_var = model(data)
        loss = loss_function(reconstructed, data, mu, log_var)
        loss.backward()
```

```
        train_loss += loss.item()
        optimizer.step()

        if batch_idx % 100 == 0:
            print(f"Epoch {epoch}, Batch {batch_idx}/{len(train_loader)}, Loss: {loss.item() / len(data):.4f}")

    print(f"Epoch {epoch}, Average Loss: {train_loss / len(train_loader.dataset):.4f}")

# Train the model
for epoch in range(epochs):
    train_epoch(epoch)
```

```
Epoch 0, Batch 0/469, Loss: 543.7628
Epoch 0, Batch 100/469, Loss: 305.0846
Epoch 0, Batch 200/469, Loss: 283.6670
Epoch 0, Batch 300/469, Loss: 287.1628
Epoch 0, Batch 400/469, Loss: 267.8576
Epoch 0, Average Loss: 298.8270
Epoch 1, Batch 0/469, Loss: 281.9788
Epoch 1, Batch 100/469, Loss: 260.8710
Epoch 1, Batch 200/469, Loss: 261.6942
Epoch 1, Batch 300/469, Loss: 261.2574
Epoch 1, Batch 400/469, Loss: 257.7576
Epoch 1, Average Loss: 260.4260
Epoch 2, Batch 0/469, Loss: 254.8357
Epoch 2, Batch 100/469, Loss: 246.4852
Epoch 2, Batch 200/469, Loss: 261.1157
Epoch 2, Batch 300/469, Loss: 248.5186
Epoch 2, Batch 400/469, Loss: 249.1737
Epoch 2, Average Loss: 253.6424
Epoch 3, Batch 0/469, Loss: 248.3493
Epoch 3, Batch 100/469, Loss: 262.6652
Epoch 3, Batch 200/469, Loss: 259.0901
Epoch 3, Batch 300/469, Loss: 254.1499
Epoch 3, Batch 400/469, Loss: 256.1468
Epoch 3, Average Loss: 249.6442
Epoch 4, Batch 0/469, Loss: 255.3473
Epoch 4, Batch 100/469, Loss: 263.5406
Epoch 4, Batch 200/469, Loss: 244.2292
Epoch 4, Batch 300/469, Loss: 248.2120
Epoch 4, Batch 400/469, Loss: 230.5767
Epoch 4, Average Loss: 247.3209
Epoch 5, Batch 0/469, Loss: 239.7042
Epoch 5, Batch 100/469, Loss: 251.8990
Epoch 5, Batch 200/469, Loss: 250.7216
Epoch 5, Batch 300/469, Loss: 236.9218
Epoch 5, Batch 400/469, Loss: 241.6642
Epoch 5, Average Loss: 245.7056
Epoch 6, Batch 0/469, Loss: 232.5364
Epoch 6, Batch 100/469, Loss: 254.6011
Epoch 6, Batch 200/469, Loss: 241.6078
Epoch 6, Batch 300/469, Loss: 248.7771
Epoch 6, Batch 400/469, Loss: 236.6903
Epoch 6, Average Loss: 244.5790
Epoch 7, Batch 0/469, Loss: 247.1589
Epoch 7, Batch 100/469, Loss: 244.6559
Epoch 7, Batch 200/469, Loss: 251.1007
Epoch 7, Batch 300/469, Loss: 241.5705
Epoch 7, Batch 400/469, Loss: 243.7389
Epoch 7, Average Loss: 243.8163
Epoch 8, Batch 0/469, Loss: 260.0096
Epoch 8, Batch 100/469, Loss: 233.2990
Epoch 8, Batch 200/469, Loss: 244.6042
Epoch 8, Batch 300/469, Loss: 237.5286
Epoch 8, Batch 400/469, Loss: 232.1450
Epoch 8, Average Loss: 243.1802
Epoch 9, Batch 0/469, Loss: 255.9304
Epoch 9, Batch 100/469, Loss: 236.2511
```

```
Epoch 9, Batch 200/469, Loss: 246.2390
Epoch 9, Batch 300/469, Loss: 229.0603
Epoch 9, Batch 400/469, Loss: 236.9773
Epoch 9, Average Loss: 242.6769
Epoch 10, Batch 0/469, Loss: 245.8609
Epoch 10, Batch 100/469, Loss: 240.7279
Epoch 10, Batch 200/469, Loss: 245.2117
Epoch 10, Batch 300/469, Loss: 257.6628
Epoch 10, Batch 400/469, Loss: 245.6034
Epoch 10, Average Loss: 242.2416
Epoch 11, Batch 0/469, Loss: 237.5823
Epoch 11, Batch 100/469, Loss: 243.4727
Epoch 11, Batch 200/469, Loss: 240.1369
Epoch 11, Batch 300/469, Loss: 247.3159
Epoch 11, Batch 400/469, Loss: 238.7717
Epoch 11, Average Loss: 241.8055
Epoch 12, Batch 0/469, Loss: 234.1462
Epoch 12, Batch 100/469, Loss: 243.9664
Epoch 12, Batch 200/469, Loss: 251.7397
Epoch 12, Batch 300/469, Loss: 248.6368
Epoch 12, Batch 400/469, Loss: 258.4583
Epoch 12, Average Loss: 241.4957
Epoch 13, Batch 0/469, Loss: 245.5028
Epoch 13, Batch 100/469, Loss: 230.1773
Epoch 13, Batch 200/469, Loss: 244.7818
Epoch 13, Batch 300/469, Loss: 243.6082
Epoch 13, Batch 400/469, Loss: 239.8372
Epoch 13, Average Loss: 241.2280
Epoch 14, Batch 0/469, Loss: 241.3689
Epoch 14, Batch 100/469, Loss: 245.7228
Epoch 14, Batch 200/469, Loss: 238.7797
Epoch 14, Batch 300/469, Loss: 239.5878
Epoch 14, Batch 400/469, Loss: 242.2291
Epoch 14, Average Loss: 240.9801
Epoch 15, Batch 0/469, Loss: 248.0406
Epoch 15, Batch 100/469, Loss: 233.0863
Epoch 15, Batch 200/469, Loss: 253.5703
Epoch 15, Batch 300/469, Loss: 230.8655
Epoch 15, Batch 400/469, Loss: 237.6715
Epoch 15, Average Loss: 240.7469
Epoch 16, Batch 0/469, Loss: 246.0285
Epoch 16, Batch 100/469, Loss: 254.6825
Epoch 16, Batch 200/469, Loss: 236.5495
Epoch 16, Batch 300/469, Loss: 244.3291
Epoch 16, Batch 400/469, Loss: 231.9512
Epoch 16, Average Loss: 240.4991
Epoch 17, Batch 0/469, Loss: 246.9929
Epoch 17, Batch 100/469, Loss: 234.1529
Epoch 17, Batch 200/469, Loss: 247.7659
Epoch 17, Batch 300/469, Loss: 233.0761
Epoch 17, Batch 400/469, Loss: 245.3297
Epoch 17, Average Loss: 240.2998
Epoch 18, Batch 0/469, Loss: 247.9945
Epoch 18, Batch 100/469, Loss: 233.9246
Epoch 18, Batch 200/469, Loss: 231.5633
Epoch 18, Batch 300/469, Loss: 240.9778
```

```
Epoch 18, Batch 400/469, Loss: 238.5697
Epoch 18, Average Loss: 240.1955
Epoch 19, Batch 0/469, Loss: 239.7804
Epoch 19, Batch 100/469, Loss: 233.5929
Epoch 19, Batch 200/469, Loss: 236.9984
Epoch 19, Batch 300/469, Loss: 235.9163
Epoch 19, Batch 400/469, Loss: 234.6351
Epoch 19, Average Loss: 240.0012
```

In [5]:

```
def generate_images():
    model.eval()

    with torch.no_grad():
        latent_samples = torch.randn(64, latent_dim).to(device)
        generated = model.decoder(latent_samples).cpu()

    # Plot the generated images
    fig, axes = plt.subplots(8, 8, figsize=(10, 10))
    for i, ax in enumerate(axes.flat):
        ax.imshow(generated[i].view(28, 28), cmap='gray')
        ax.axis('off')

    plt.show()

# Generate and plot images
generate_images()
```

In [ ]: