Lab 10 : Perform compression on MNIST dataset using autoencoder

Aim : To design and implement an Autoencoder model using Pytorch to perform data compression on the MNIST dataset, by encoding input images into a low-dimensional latent representation and reconstructing them back.

Description :

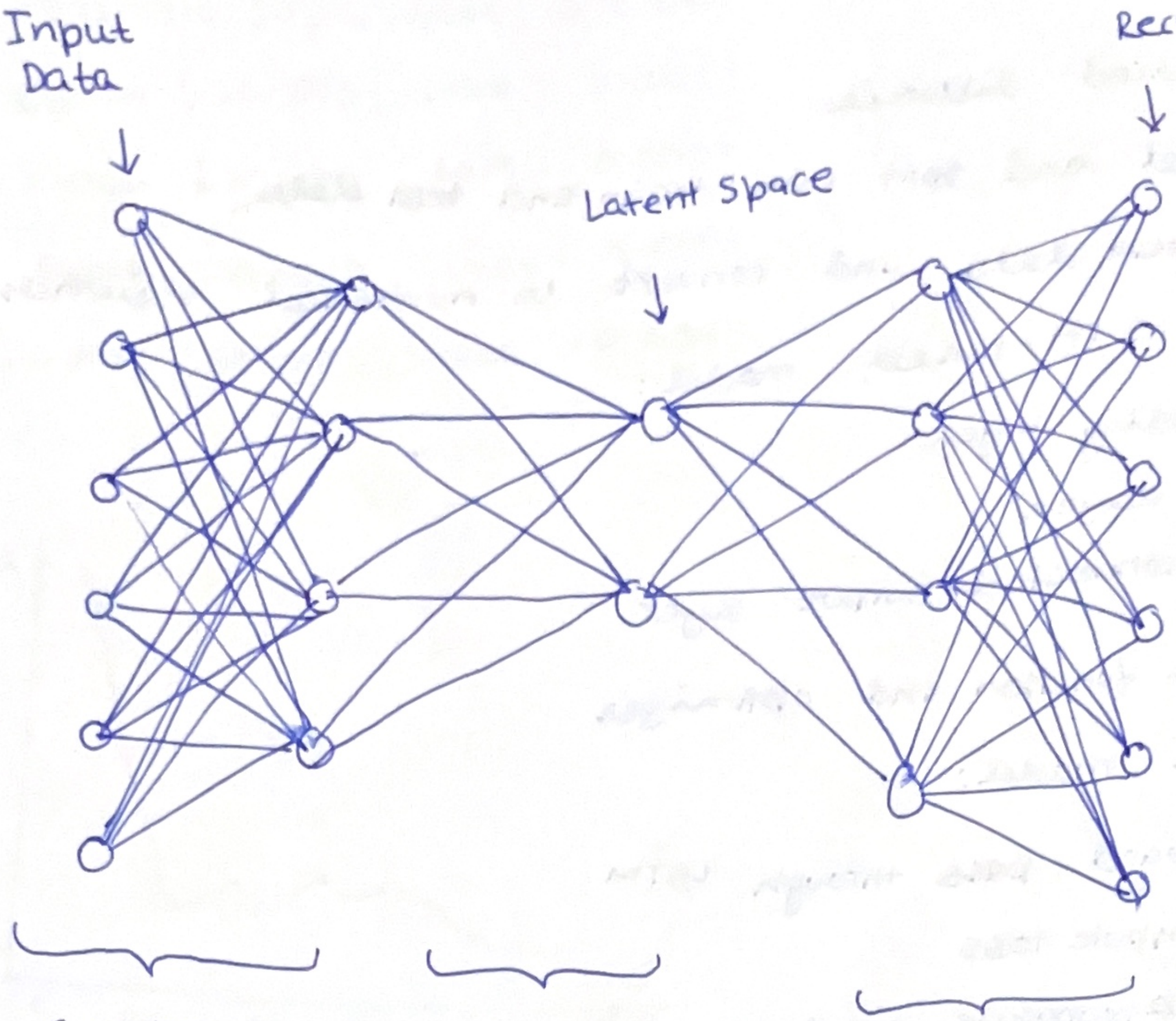
An autoencoder is a type of neural network used to learn efficient codings of unlabeled data. It aims to compress (encode) the input data into a smaller latent space and then reconstruct (decode) it back as accurately as possible.

1. Encoder
   - Maps the input data x to a lower-dimensional representation z.
   - $z = f(x)$

2. Decoder
   - Attempts to reconstruct the input from z.
   - $\hat{x} = g(z)$

3. Loss function
   - The difference between the input and reconstructed output is minimized using mean square error.

$$\text{Loss} = \frac{1}{N} \sum_i (x_i - \hat{x}_i)^2$$



Autoencoder Architecture

Pseudocode:

1. Import necessary libraries (torch, torchvision, matplotlib, etc.)
2. Load MNIST dataset and normalize it to [0,1]
3. Define Autoencoder class with:

    Encoder: Flatten → Linear (784 → 128 → 64 → 32)

    Decoder: Linear (32 → 64 → 128 → 784) → reshape to 28X28
4. Define loss function (MSELoss) and optimizer
5. For each epoch:

    a. Forward pass: encode and decode the input

    b. Compute reconstruction loss

    c. Backpropogate and update weights

    d. Store loss for visualization
6. Plot training loss curve
7. Display original and reconstructed MNIST images

Result:

The autoencoder successfully compressed MNIST images into vectors and reconstructed them with minimal loss.

The training loss decreased consistently.

The model captured important features of handwritten digits while reducing dimensionality

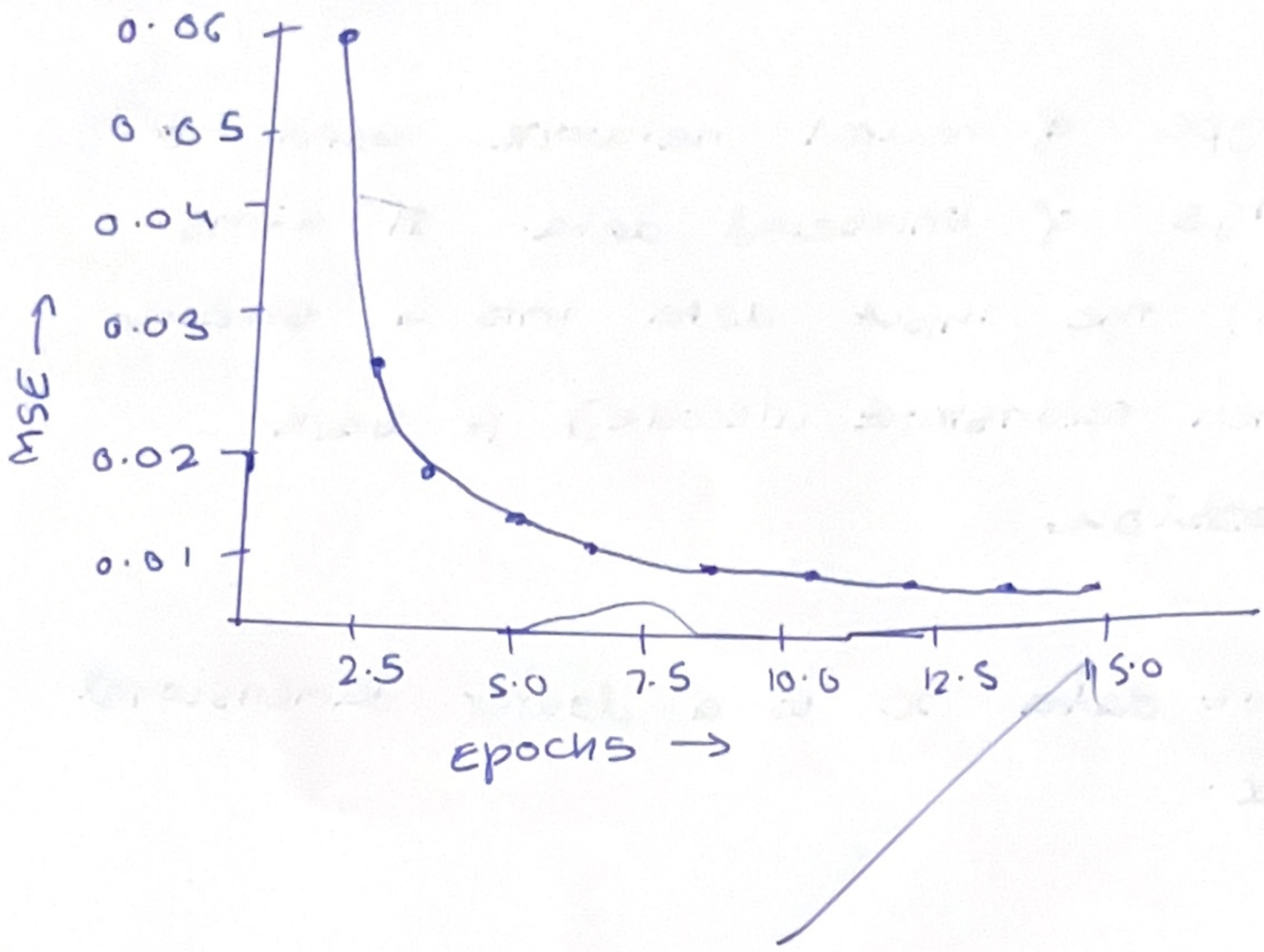
Epoch [1/20], Loss: 0.0683

Epoch [5/20], Loss : 0.0182

Epoch [10/20], Loss: 0.0129

Epoch [15/20], Loss: 0.0103

Epoch [20/20], Loss: 0.0090

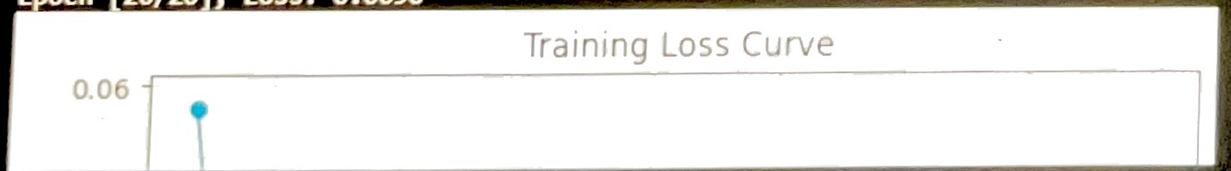


```python
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt

# 1. Load MNIST dataset
transform = transforms.Compose([transforms.ToTensor()])
train_data = datasets.MNIST(root='./data', train=True, transform=transform, download=True)
train_loader = DataLoader(train_data, batch_size=128, shuffle=True)

# 2. Define Autoencoder
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28*28, 128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 32)
        )
        self.decoder = nn.Sequential(
            nn.Linear(32, 64),
            nn.ReLU(),
            nn.Linear(64, 128),
            nn.ReLU(),
            nn.Linear(128, 28*28),
            nn.Sigmoid()
```

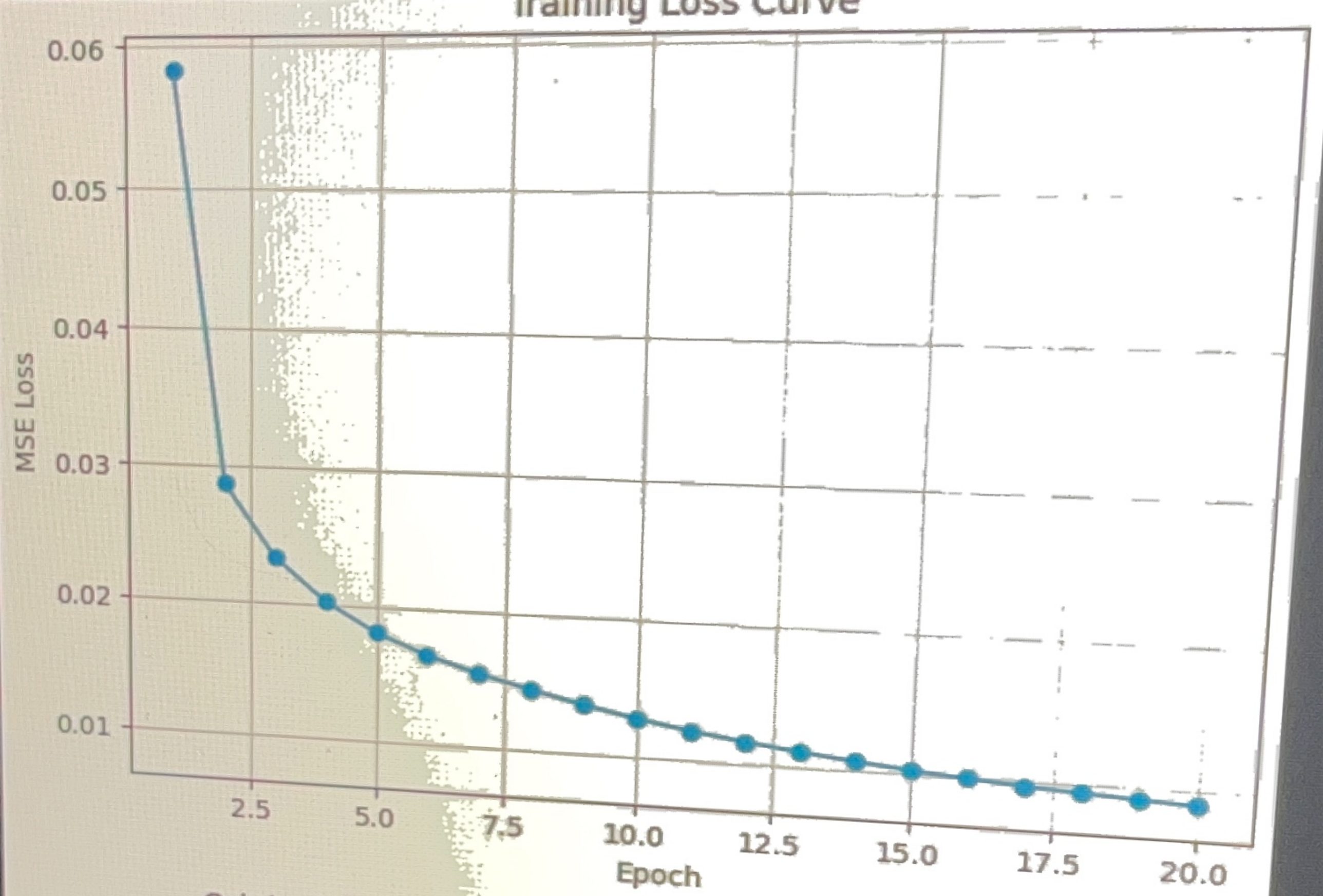
```
        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}")

# 5. Plot training loss
plt.figure(figsize=(7,5))
plt.plot(range(1, num_epochs+1), losses, marker='o')
plt.title("Training Loss Curve")
plt.xlabel("Epoch")
plt.ylabel("MSE Loss")
plt.grid(True)
plt.show()

# 6. Display original vs reconstructed images
dataiter = iter(train_loader)
images, _ = next(dataiter)
with torch.no_grad():
    reconstructed = model(images)

# Plot
plt.figure(figsize=(8, 3))
for i in range(6):
    plt.subplot(2, 6, i+1)
    plt.imshow(images[i].squeeze(), cmap='gray')
    plt.axis('off')
    plt.subplot(2, 6, i+7)
    plt.imshow(reconstructed[i].squeeze(), cmap='gray')
    plt.axis('off')
plt.suptitle("Original (Top) vs Reconstructed (Bottom) Images")
plt.show()
```

```
plt.show()
```

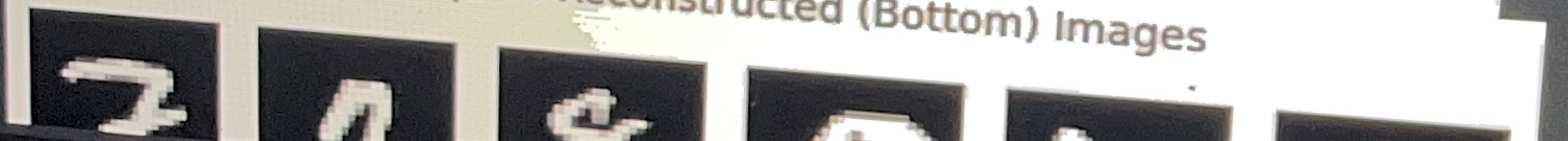
```
100%|| 9.91M/9.91M [00:00<00:00, 18.7MB/s]
100%|| 28.9k/28.9k [00:00<00:00, 498kB/s]
100%|| 1.65M/1.65M [00:00<00:00, 4.52MB/s]
100%|| 4.54k/4.54k [00:00<00:00, 8.81MB/s]
Epoch [1/20], Loss: 0.0583
Epoch [2/20], Loss: 0.0287
Epoch [3/20], Loss: 0.0234
Epoch [4/20], Loss: 0.0203
Epoch [5/20], Loss: 0.0182
Epoch [6/20], Loss: 0.0167
Epoch [7/20], Loss: 0.0155
Epoch [8/20], Loss: 0.0146
Epoch [9/20], Loss: 0.0137
Epoch [10/20], Loss: 0.0129
Epoch [11/20], Loss: 0.0122
Epoch [12/20], Loss: 0.0117
Epoch [13/20], Loss: 0.0112
Epoch [14/20], Loss: 0.0107
Epoch [15/20], Loss: 0.0103
Epoch [16/20], Loss: 0.0100
Epoch [17/20], Loss: 0.0097
Epoch [18/20], Loss: 0.0095
Epoch [19/20], Loss: 0.0092
Epoch [20/20], Loss: 0.0090
```

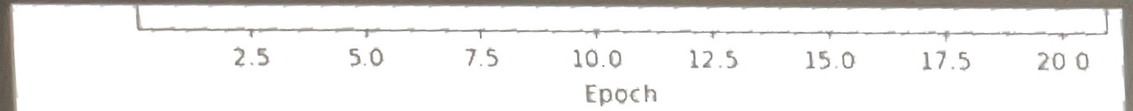


Training Loss Curve



Original (Top) vs Reconstructed (Bottom) Images





Original (Top) vs Reconstructed (Bottom) Images