Commands + Code + Text ▷ Run all

RAM
Disk

[1]
2m

```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 and Prepare MNIST Data
# ===============================
transform = transforms.Compose([
    transforms.ToTensor(),
])

train_data = datasets.MNIST(root='./data', train=True, download=True, transform=transform)
test_data = datasets.MNIST(root='./data', train=False, download=True, transform=transform)

train_loader = DataLoader(train_data, batch_size=128, shuffle=True)
test_loader = DataLoader(test_data, batch_size=128, shuffle=False)

# ===============================
# 2. Define Autoencoder Model
# ===============================
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()

        # Encoder: compress input (28x28 → 64 features)
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 256),
            nn.ReLU(True),
            nn.Linear(256, 64),
            nn.ReLU(True)
        )
```

Commands + Code + Text Run all

RAM
Disk

[1]
2m

```
    'features': compressed_features,
    'labels': labels
}, 'mnist_compressed.pt')

print("Compressed features saved to mnist_compressed.pt")
```

```
100%|██████████| 9.91M/9.91M [00:00<00:00, 62.2MB/s]
100%|██████████| 28.9k/28.9k [00:00<00:00, 1.65MB/s]
100%|██████████| 1.65M/1.65M [00:00<00:00, 14.3MB/s]
100%|██████████| 4.54k/4.54k [00:00<00:00, 6.73MB/s]
Epoch [1/10], Loss: 0.0425
Epoch [2/10], Loss: 0.0165
Epoch [3/10], Loss: 0.0120
Epoch [4/10], Loss: 0.0100
Epoch [5/10], Loss: 0.0088
Epoch [6/10], Loss: 0.0079
Epoch [7/10], Loss: 0.0072
Epoch [8/10], Loss: 0.0067
Epoch [9/10], Loss: 0.0063
Epoch [10/10], Loss: 0.0060
```


```
Compressed features saved to mnist_compressed.pt
```

File Edit View Insert Runtime Tools Help

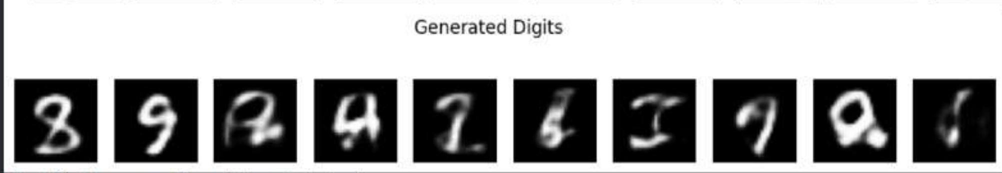
Commands + Code + Text ▷ Run all RAM Disk

[2]
✓ 2m

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

# =====================================
# 1. Load and Prepare the MNIST Dataset
# =====================================
transform = transforms.Compose([
    transforms.ToTensor(),
])

train_data = datasets.MNIST(root='./data', train=True, download=True, transform=transform)
test_data = datasets.MNIST(root='./data', train=False, download=True, transform=transform)

train_loader = DataLoader(train_data, batch_size=128, shuffle=True)
test_loader = DataLoader(test_data, batch_size=128, shuffle=False)

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

# =====================================
# 2. Define Variational Autoencoder (VAE)
# =====================================
class VAE(nn.Module):
    def __init__(self, latent_dim=20):
        super(VAE, self).__init__()
        self.latent_dim = latent_dim

        # Encoder: 784 -> 400 -> μ, log(σ²)
        self.fc1 = nn.Linear(28*28, 400)
        self.fc_mu = nn.Linear(400, latent_dim)
        self.fc_logvar = nn.Linear(400, latent_dim)
```

```
}, 'mnist_vae_latent.pt')

print("Latent features saved to mnist_vae_latent.pt")
```

```
Epoch [1/10]  Loss: 162.8189
Epoch [2/10]  Loss: 120.8224
Epoch [3/10]  Loss: 114.1165
Epoch [4/10]  Loss: 111.2614
Epoch [5/10]  Loss: 109.5371
Epoch [6/10]  Loss: 108.3486
Epoch [7/10]  Loss: 107.5457
Epoch [8/10]  Loss: 106.9098
Epoch [9/10]  Loss: 106.4470
Epoch [10/10]  Loss: 106.0076
```




```
Latent features saved to mnist_vae_latent.pt
```

Output:-

Epoch [1/10]; loss : 164.4406

Epoch [2/10]: Loss : 121.4260

Epoch [3/10]; Loss : 111.4229

Epoch [4/10]; Loss : 109.7214

Epoch [5/10]; Loss : 108.5538

0	9	9	5
8	8	3	3

Reparametrization:

Decoder:

Forward:

Define loss function:

Reconstruct loss

Initialize model, optimize

For each epoch:

For epoch

recon, M, logvar = model(img)

loss - BCE + KLD

print average epoch loss

Test nodes

Display generated Samples.

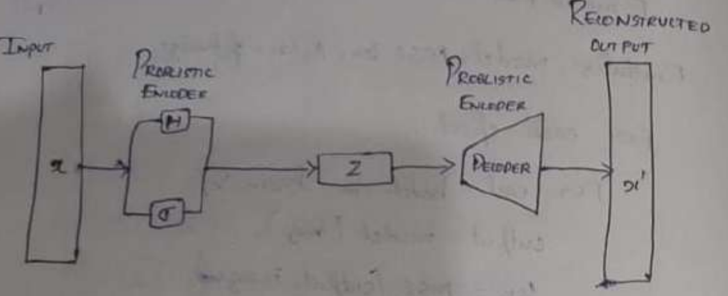
END

Result:-

Successfully experimented using variation

Autoencoder.

Architecture of VAE :-



Exp 11 Experiment using variational autoencoder (VAE)

Aim:-

To perform a Experiment using variational autoencoder.

Objectives:-

1.) To understand the concept of VAE and how they diffs from standard autoencoder

2.) To implement VAE model using pytorch that learn a latern distribution.

3.) To reconstruct and generate new image using the learned latent space.

4.) To evaluate the generations capability of the trained VAE.

Pseudocode:

BEGIN

Import torch, torch.nn, torch.optim

Load MNIST data

DEFINE VAE class:

Encode :

Output:

Epoch [1/10] ; Loss : 0.0608

Epoch [2/10] ; Loss : 0.0322

Epoch [3/10] ; Loss : 0.0262

Epoch [4/10] ; Loss : 0.0212

Epoch [5/10] ; Loss : 0.0176

⋮

Epoch [10/10] ; Loss : 0.0157

$\boxed{7}$ $\boxed{2}$ $\boxed{1}$ $\boxed{0}$ $\boxed{4}$

Decoder:

Forward pass:

Initialize model, MSE loss, Adam optimizer.

For each epoch :

  For each batch in train set.

    output = model (img)

    loss = MSE (output, image)

    print epoch loss

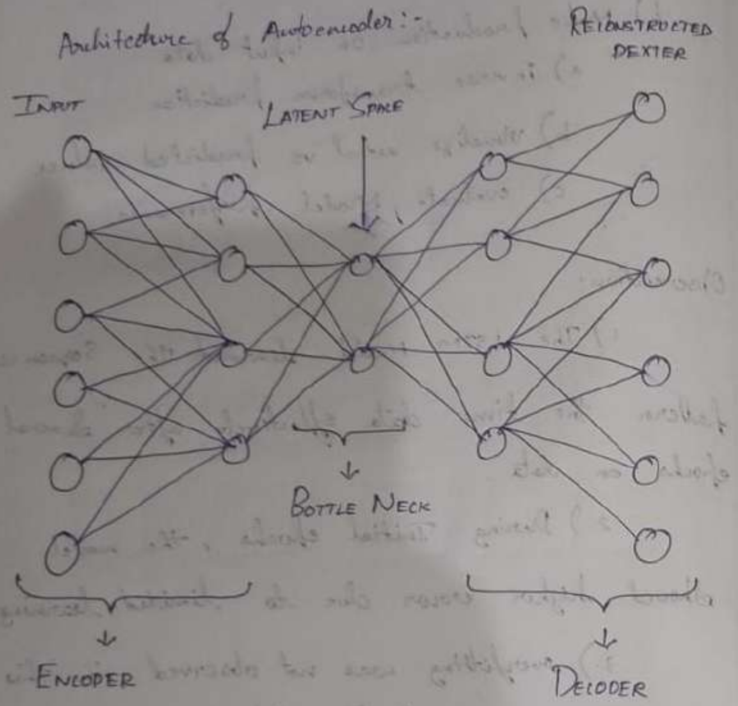
  Test model on Sample img

END

RESULT :-

Successfully Performed Compression on MNIST using Autoencoders.

Architecture of Autoencoder:-



EX:10 Perform Compression on mnist Dataset using Autoencoder.

Aim:-

To perform compression on mnist Dataset using Autoencoder.

Objectives:-

1.) To understand the working principle of an autoencoder and its encoder-decoder Structure

2.) To build and train a fully connected Autoencoder using Pytorch on mnist dataset.

3.) To evaluate the models ability to Compress and reconstruct images.

4.) To visualize the original and reconstructed image to analyze reconstruction quality.

Pseudocode :-

Begin

Import torch, torch.n.n, torch.opt,

Load MNIST dataset will transfer

create data Load for train & test

Define Autoencoder class:-

Encoder: