09/10/25 LAB: 11

Experiments using Variational Autoencoder

*Aim :-

To implement a variational autoencoder for compressing and generating MNIST digit images using a probabilistic approach

*objective :-

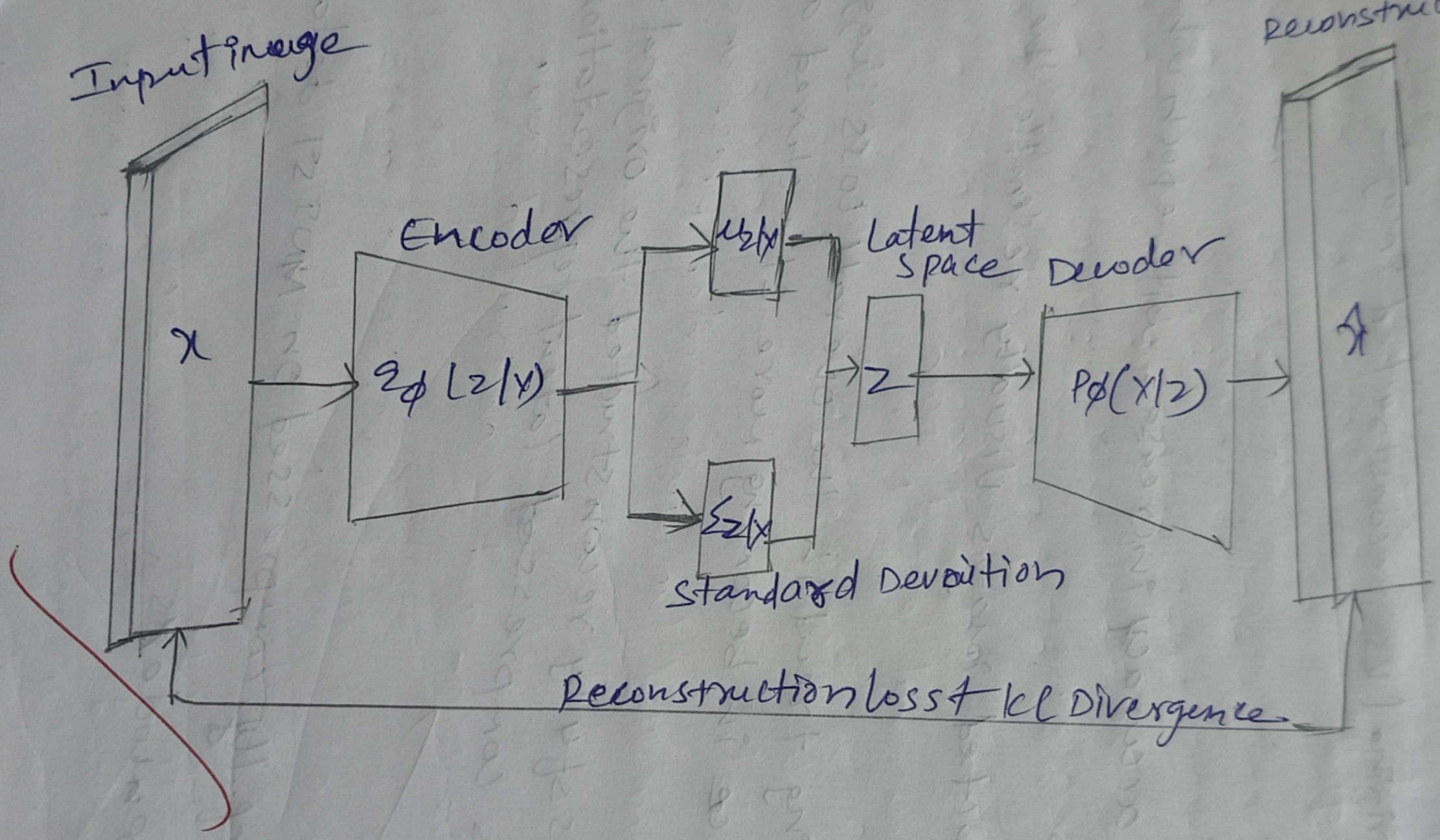
1. To learn a latent space representation of MNIST digits.
2. To generate new images by sampling from the learned latent distribution.
3. To compare the reconstructed and generated images with the original dataset.

*pseudocode :-

Begin
    Load MNIST dataset
    Normalize and flatten images (28X28 → 784)
    Define encoder:
        Input layer → Dense (256, Relu)
        Mean layer(μ) → Dense (latent_dim)
        Log variance layer(σ^2) → Dense (latent_dim
        Sample latent vector z = μ + exp(σ^2/2) * ε
        where ε ~ N(0,1)

    Define decoder:
        Input z → Dense (256, Relu)
        Output → Dense (784, Sigmoid) →
                re constructed i

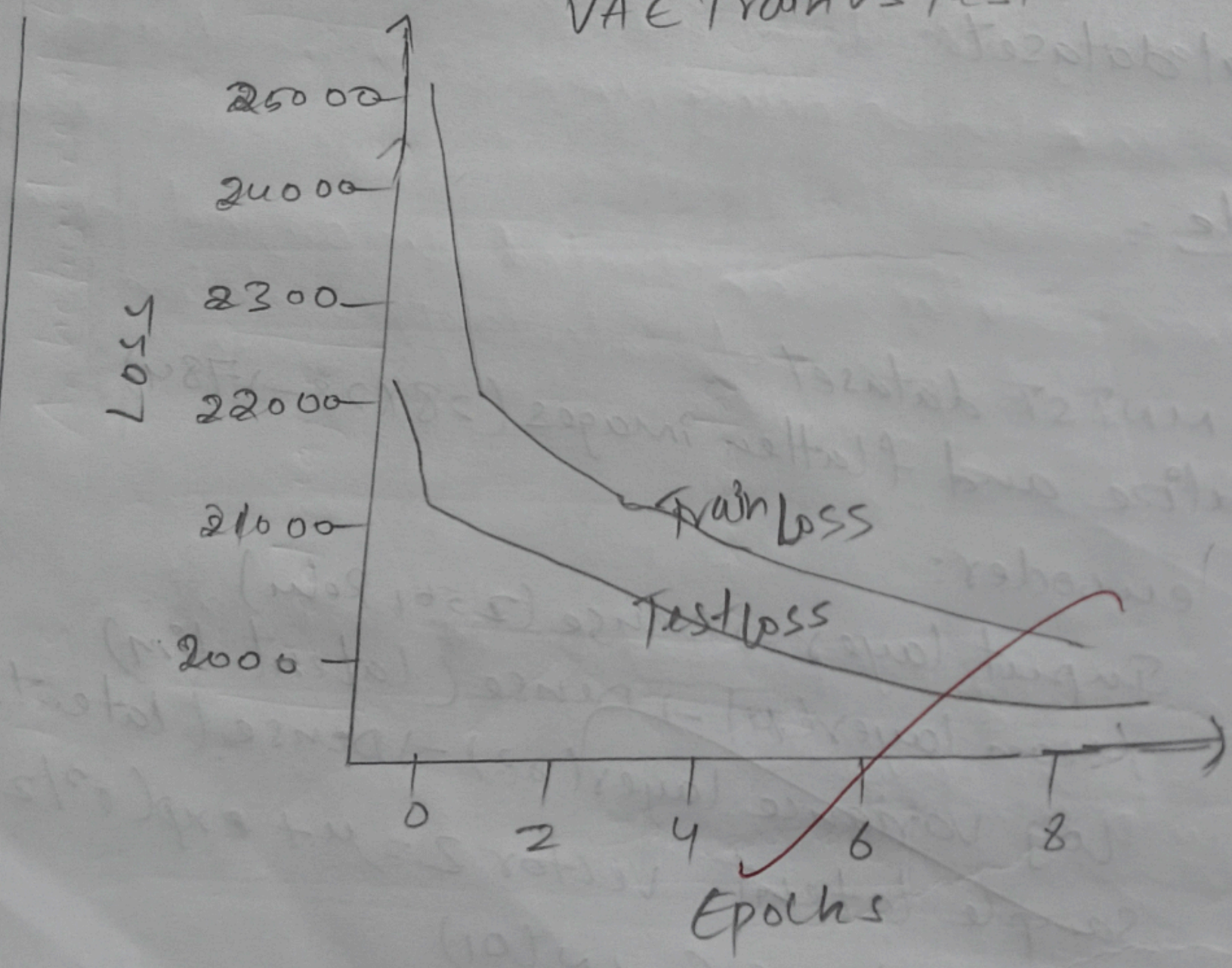
Architecture of VAE



Output:-

Epoch 1/10, loss: 25003.6881, Test loss: 22058.46
Epoch 2/10, loss: 21878.4932, Test loss: 21122.08
Epoch 3/10, loss: 21232.1173, Test loss: 20715.54
Epoch 4/10, loss: 20888.5306, Test loss: 20482.5
Epoch 5/10, loss: 20659.487 Test loss: 20302.64
Epoch 6/10, loss: 20486.1709, Test loss: 20138.84
Epoch 7/10, loss: 20346.4544, Test loss: 2003 9.99
Epoch 8/10, loss: 20235.4947, Test loss: 1970.30
Epoch 9/10, loss: 20137.8976, Test loss: 1922.53
Epoch 10/10, loss: 2005 5.5366, Test loss: 19820.25

VAE Train vs Test loss



Define loss:
Total loss = Reconstruction loss + kl divergence loss

Train model using Adam optimizer for multiple epochs

Encode test images → get latent vectors

Decode latent vectors → reconstruct images

Sample random z values → generate new images

End

*Observation =

→ Instead of mapping input to a fixed point, VAE encode input into a probability distribution in latent space

→ VAEs use the reparametesization trick to allow for back propagation through the sampling process

→ The loss function of VAE includes a reconstruction loss & KL divergen to regularize space & encourage to follow a prior distribution

*Result =

Successfully implemented VAE (variational autoencoder) using MINST dataset

Eg.

Lab10.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[ ]

```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
import matplotlib.cm as cm
import numpy as np

# ------------------------------
# Hyperparameters
# ------------------------------
batch_size = 128
epochs = 10
learning_rate = 1e-3
latent_dim = 2  # 2D latent space for visualization

# ------------------------------
# Dataset
# ------------------------------
transform = transforms.ToTensor()
train_dataset = datasets.MNIST(root='data', train=True, download=True, transform=transform)
test_dataset = datasets.MNIST(root='data', train=False, download=True, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

# ------------------------------
# VAE Model
# ------------------------------
class VAE(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28, 256)
        self.fc_mu = nn.Linear(256, latent_dim)
        self.fc_logvar = nn.Linear(256, latent_dim)
        self.fc2 = nn.Linear(latent_dim, 256)
        self.fc3 = nn.Linear(256, 28*28)

    def encode(self, x):
        h = torch.relu(self.fc1(x))
        return self.fc_mu(h), self.fc_logvar(h)

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

Lab10.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
        def forward(self, x):
            mu, logvar = self.encode(x)
            z = self.reparameterize(mu, logvar)
            return self.decode(z), mu, logvar

    # ----------------------------
    # Device and Model
    # ----------------------------
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = VAE().to(device)
    optimizer = optim.Adam(model.parameters(), lr=learning_rate)

    # ----------------------------
    # Loss Function
    # ----------------------------
    def loss_fn(recon_x, x, mu, logvar):
        BCE = nn.functional.binary_cross_entropy(recon_x, x.view(-1, 28*28), reduction='sum')
        KLD = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
        return BCE + KLD

    # ----------------------------
    # Training Loop with Train & Test Loss
    # ----------------------------
    train_losses = []
    test_losses = []

    for epoch in range(epochs):
        model.train()
        running_loss = 0
        for x, _ in train_loader:
            x = x.to(device).view(x.size(0), -1)  # Flatten
            optimizer.zero_grad()
            recon, mu, logvar = model(x)
            loss = loss_fn(recon, x, mu, logvar)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        avg_train_loss = running_loss / len(train_loader)
        train_losses.append(avg_train_loss)

        # Compute test loss
        model.eval()
        test_loss = 0
        with torch.no_grad():
            for x, _ in test_loader:
                x = x.to(device).view(x.size(0), -1)
                recon, mu, logvar = model(x)
                test_loss += loss_fn(recon, x, mu, logvar).item()
    avg_test_loss = test_loss / len(test_loader)
```

Lab10.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[ ]
# ------------------------------
# Visualize 2D Latent Space
# ------------------------------
x_sample, y_sample = next(iter(test_loader))
x_sample = x_sample.to(device).view(x_sample.size(0), -1)
with torch.no_grad():
    mu, logvar = model.encode(x_sample)
mu = mu.cpu().numpy()

plt.figure(figsize=(8,6))
scatter = plt.scatter(mu[:,0], mu[:,1], c=y_sample, cmap=cm.tab10)
plt.colorbar(scatter)
plt.title('VAE 2D Latent Space (Test Set)')
plt.show()
```

```
Epoch 1/10 | Train Loss: 25003.6881 | Test Loss: 22056.4637
Epoch 2/10 | Train Loss: 21878.4032 | Test Loss: 21122.0819
Epoch 3/10 | Train Loss: 21232.1173 | Test Loss: 20715.5480
Epoch 4/10 | Train Loss: 20888.5306 | Test Loss: 20482.5152
Epoch 5/10 | Train Loss: 20658.4827 | Test Loss: 20302.1048
Epoch 6/10 | Train Loss: 20486.1709 | Test Loss: 20138.8614
Epoch 7/10 | Train Loss: 20346.4544 | Test Loss: 20039.9918
Epoch 8/10 | Train Loss: 20235.4947 | Test Loss: 19970.3085
Epoch 9/10 | Train Loss: 20137.8976 | Test Loss: 19922.5389
Epoch 10/10 | Train Loss: 20055.5366 | Test Loss: 19820.2528
```