Exp 11:

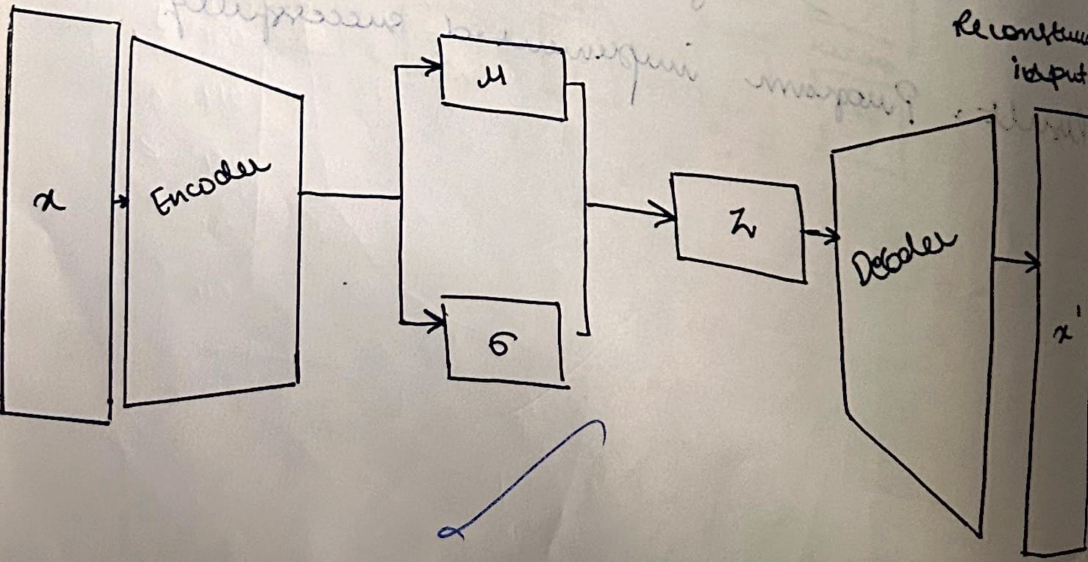
Experiment using Variational Autoencoder.

Aim: To implement a Variational Autoencoder (VAE) in deep learning for the compression & generation of MNIST handwritten digits images.

Pseudo code:

1. Import libraries & load MNIST dataset.
2. Normalize & flatten the images
3. Define VAE model:
   - Encoder: outputs mean (μ) and log-variance
   - Reparameterization Trick: $z = \mu + \sigma * \varepsilon$
   - Decoder: Reconstructs image from z.
4. Define loss:
   - Reconstruction loss (MSE)
   - KL divergence loss
   - Total loss = Reconstruction + KL Divergence
5. Train model:
   - forward pass
   - compute loss
   - Backpropagate & update
6. Evaluate on test data
7. Visualize:
   - Original v/s Reconstructed images.
   - Randomly generated images from latent space

Input.



$$z = \mu + \sigma \varepsilon$$
$$\varepsilon \sim N(0, 1)$$

Justification:

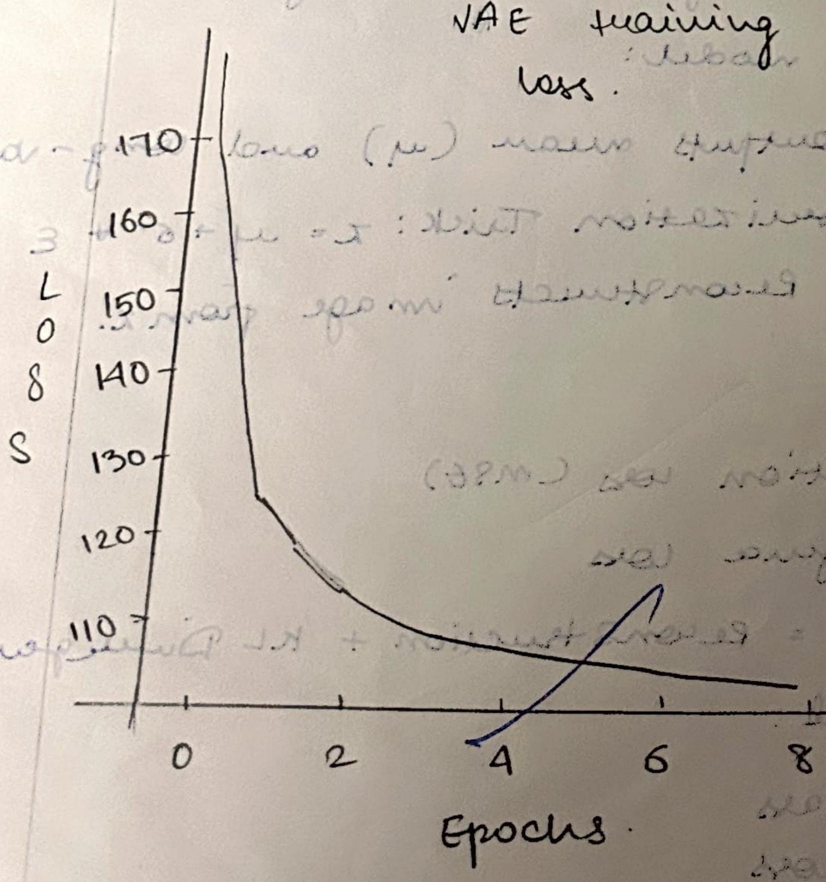
Unlike a simple Autoencoder, a VAE doesn't just learn to reconstruct input data.
It learns the underlying probability distribution of the data & allows us to generate new, similar samples.

Result: Program implemented successfully.

Output:

epoch [1/7], loss : 165.5449
epoch [2/7], loss : 121.9764
epoch [3/7], loss : 114.8512
epoch [4/7], loss : 111.6771
epoch[5/7], loss : 109.8832
epoch [6/7], loss : 108.6224
epoch [7/7], loss : 107.7510

Evaluation metrics:
Reconstruction Accuracy : 0.9639
F1 score : 0.8660



Untitled8.ipynb ☆ Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share

Connecting

[ ]

```
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
from sklearn.metrics import f1_score, accuracy_score
import numpy as np
transform = transforms.Compose([transforms.ToTensor()])
train_data = datasets.MNIST(root="./data", train=True, transform=transform, download=True)
test_data = datasets.MNIST(root="./data", train=False, transform=transform, download=True)

train_loader = DataLoader(train_data, batch_size=128, shuffle=True)
test_loader = DataLoader(test_data, batch_size=128, shuffle=False)
class VAE(nn.Module):
    def __init__(self, latent_dim=20):
        super(VAE, self).__init__()
        self.encoder = nn.Sequential(
            nn.Flatten(),
            nn.Linear(28*28, 400),
            nn.ReLU()
        )
        self.mu = nn.Linear(400, latent_dim)
        self.logvar = nn.Linear(400, latent_dim)
        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, 400),
            nn.ReLU(),
            nn.Linear(400, 28*28),
            nn.Sigmoid()
        )
```

Variables Terminal

4:32 PM Connecting to Python 3

Untitled8.ipynb   Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all   Connecting

```
            nn.Linear(400, 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):
        encoded = self.encoder(x)
        mu = self.mu(encoded)
        logvar = self.logvar(encoded)
        z = self.reparameterize(mu, logvar)
        decoded = self.decoder(z)
        return decoded, mu, logvar
def vae_loss(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
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = VAE().to(device)
optimizer = optim.Adam(model.parameters(), lr=0.001)
train_losses = []
for epoch in range(10):
    model.train()
    total_loss = 0

    for batch_idx, (data, _) in enumerate(train_loader):
        data = data.to(device)
        optimizer.zero_grad()
        recon, mu, logvar = model(data)
```

Variables   Terminal   4:32 PM

Untitled8.ipynb ☆ Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all

Connecting

[ ]

```python
    model.train()
    total_loss = 0

    for batch_idx, (data, _) in enumerate(train_loader):
        data = data.to(device)
        optimizer.zero_grad()
        recon, mu, logvar = model(data)
        loss = vae_loss(recon, data, mu, logvar)
        loss.backward()
        total_loss += loss.item()
        optimizer.step()

    avg_loss = total_loss / len(train_loader.dataset)
    train_losses.append(avg_loss)
    print(f"Epoch [{epoch+1}/10], Loss: {avg_loss:.4f}")
model.eval()
reconstructions, originals = [], []

with torch.no_grad():
    for data, _ in test_loader:
        data = data.to(device)
        recon, _, _ = model(data)
        reconstructions.append(recon.cpu())
        originals.append(data.cpu())

reconstructions = torch.cat(reconstructions)
originals = torch.cat(originals)
orig_bin = (originals.view(-1, 28*28) > 0.5).int().numpy()
recon_bin = (reconstructions.view(-1, 28*28) > 0.5).int().numpy()

acc = accuracy_score(orig_bin.flatten(), recon_bin.flatten())
f1 = f1_score(orig_bin.flatten(), recon_bin.flatten())
```

{ } Variables Terminal 4:32 PM

Untitled8.ipynb ☆ Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all Connecting

Share A

```
acc = accuracy_score(orig_bin.flatten(), recon_bin.flatten())
f1 = f1_score(orig_bin.flatten(), recon_bin.flatten())

print("\n✅ Evaluation Metrics:")
print(f"Reconstruction Accuracy: {acc:.4f}")
print(f"F1 Score: {f1:.4f}")
plt.figure(figsize=(6,4))
plt.plot(train_losses, label='Train Loss')
plt.title("VAE Training Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
n = 8
plt.figure(figsize=(12,4))
for i in range(n):
    plt.subplot(2, n, i+1)
    plt.imshow(originals[i][0], cmap="gray")
    plt.axis("off")
    plt.subplot(2, n, n+i+1)
    plt.imshow(reconstructions[i].view(28,28), cmap="gray")
    plt.axis("off")

plt.suptitle("Top: Original | Bottom: Reconstructed", fontsize=14)
plt.show()
```

```
100%|██████████| 9.91M/9.91M [00:00<00:00, 60.3MB/s]
100%|██████████| 28.9k/28.9k [00:00<00:00, 1.88MB/s]
100%|██████████| 1.65M/1.65M [00:00<00:00, 15.1MB/s]
100%|██████████| 4.54k/4.54k [00:00<00:00, 7.15MB/s]
Epoch [1/10], Loss: 165.5449
```

Variables Terminal 4:32 PM

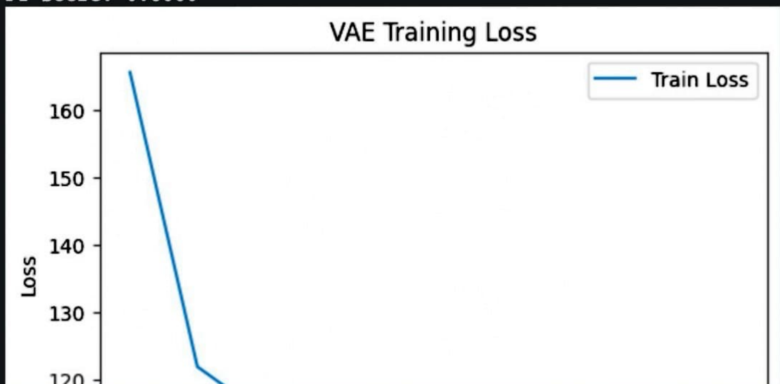
Untitled8.ipynb ☆ Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all ··· Connecting

```
100%|██████████| 9.91M/9.91M [00:00<00:00, 60.3MB/s]
100%|██████████| 28.9k/28.9k [00:00<00:00, 1.88MB/s]
100%|██████████| 1.65M/1.65M [00:00<00:00, 15.1MB/s]
100%|██████████| 4.54k/4.54k [00:00<00:00, 7.15MB/s]
Epoch [1/10], Loss: 165.5449
Epoch [2/10], Loss: 121.9764
Epoch [3/10], Loss: 114.8512
Epoch [4/10], Loss: 111.6771
Epoch [5/10], Loss: 109.8832
Epoch [6/10], Loss: 108.6224
Epoch [7/10], Loss: 107.7510
Epoch [8/10], Loss: 107.0746
Epoch [9/10], Loss: 106.5901
Epoch [10/10], Loss: 106.1244

✅ Evaluation Metrics:
Reconstruction Accuracy: 0.9639
F1 Score: 0.8660
```



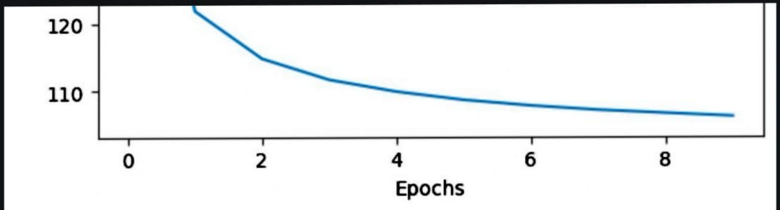
Variables Terminal 4:32 PM

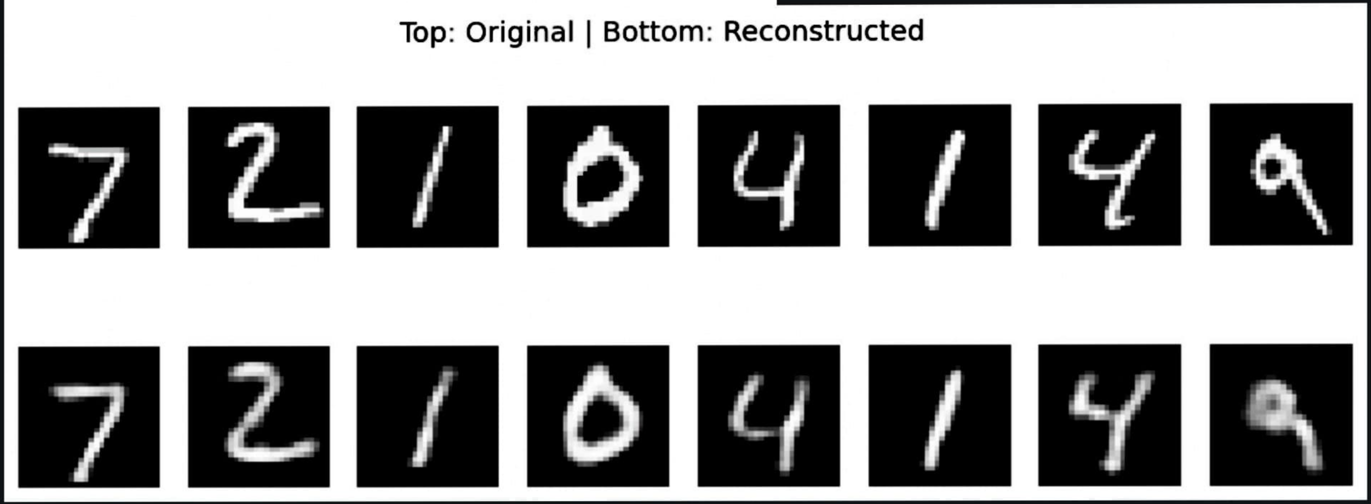
Untitled8.ipynb ☆ Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all

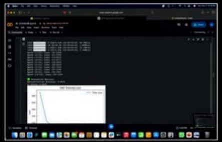
Connecting





[ ]

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
```



{ } Variables   Terminal   4:32 PM   Co