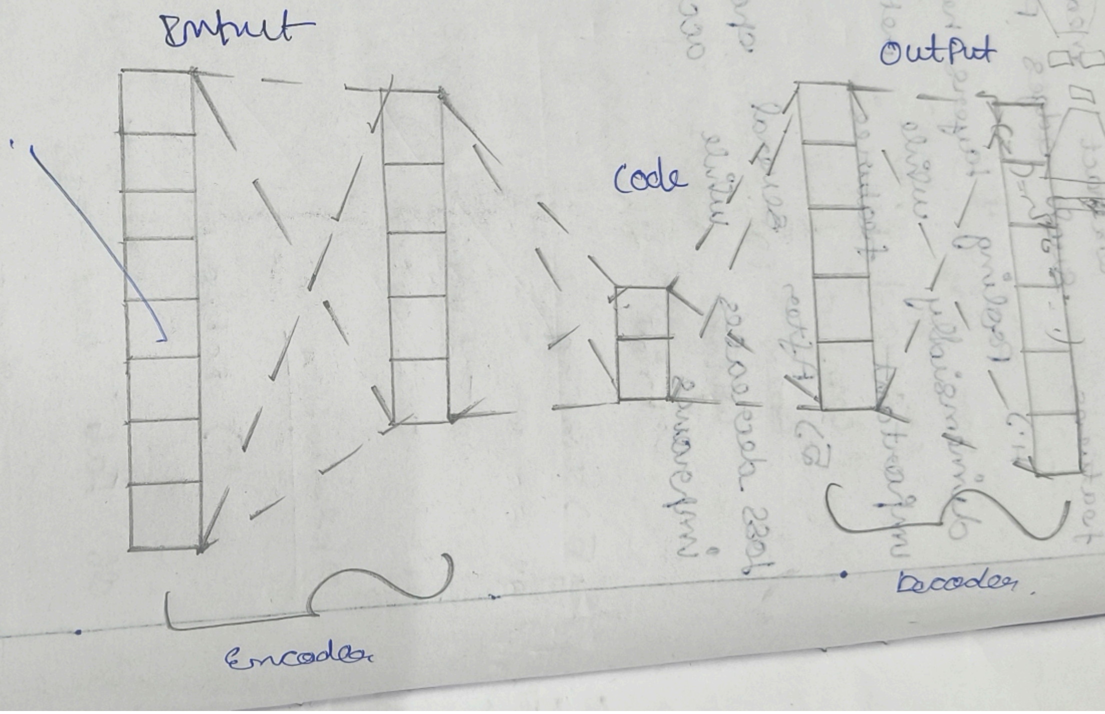
Architecture:-



10. Perform compression on mnist dataset using auto encoder.

Aim

TO Perform compression on minst dataset using auto encoder

objective

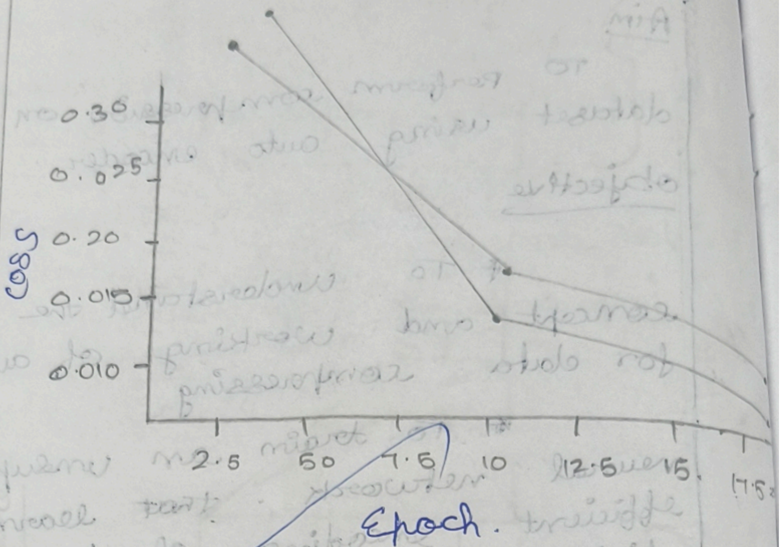
* To understand the concept and working of autoencoder for data compressing

* To train an unsupervised neural network. that learns efficient encoding of handwritten digit images from the MNIST dataset

* To compare input and reconstructed image to evalute compression performance

PSEUDOCODE

```
BEGIN
  1. Input required libraries (torch, torch.nn)
  2. Load MNIST dataset and normali image to [0,1] Flatten images to 784- dimensional
```


0.30
0.025
0.20
0.015
0.010
Loss
2.5
50
7.5
10
12.5
15
17.5
Epoch.

3. Define Autoencoder class:

Encoder: Linear (784->128) -> ReLU -> Linear (128->64) -> ReLU -> Linear (64->32)

Decoder: Linear (32->64) -> ReLU -> Linear (128->784) -> Sigmoid

4. Initialize model, define function (MSE Loss) and optimize (Adam)

5. After training, Pass test images through encoder and decoder

Obtain compressed (latent) and reconstructed images

6. Display few original and reconstructed images using matplotlib

7 observe quality of reconstruction and compression efficiency

END.

OBSERVATION

* The Autoencoder Successfully compressed 784- dimensional MNIST images into a 32- dimensional latent vector, achieving significant dimensionality reduction

* minor loss of fine detail observed typical of compression models

* Training loss gradually decreased sta blising after around 15-20 epochs

Output:-

```
Epoch (1/10) loss :0.060911
Epoch (2/10) loss :0.032390
Epoch (3/10) loss :0.026281
Epoch (4/10) loss :0.022895
Epoch (5/10) loss :0.20843
Epoch (6/10) loss :0.019380
Epoch (7/10) loss :0.017938
Epoch (8/10) loss :0.016113
Epoch (9/10) loss :0.015509
Epoch (10/10) loss :0.015010
```

MENT

Name : Kuberan

Sub : ELA

03110470100005

Result

Therefore the compression is successfully completed.