EXP No: 10 :- Perform Compression on MNIST

Aim :-

To implement an Autoencoder for Compressing and reconstructing MNIST images, thereby understanding the concept of dimensionality reduction using neural network.

Objectives :-

1. To build an encoder that Compresses high dimensional image data into a lower - dimensional latent space.

2. To reconstruct the input image using a decoder work

3. To visualize and evaluate the performance of Compression and reconstruction

4. To understand unsupervised feature learning using neural networks

Algorithm :-

1. Import MNIST dataset.

2. Normalize image data (scale pixels b/w 0 & 1)

3. Build an Autoencoder Consisting of:
   - Encoder: Compresses image into latent vector
   - Decoder: Reconstructs image from latent representation

4. Compile the model with loss = 'binary cross entropy' & optimizer = 'adam'.

5. Train the model on MNIST images.

6. Evaluate the reconstruction quality using test images.

7. Visualize input vs reconstructed images.

Pseudo Code:-

    Load MNIST dataset

    Normalize images to range $[0,1]$

    Define encoder:

        Flatten input

        Dense (128, relu)

        Dense (64, relu)

        Dense (latent_dim, relu)

    Define decoder:

        Dense (64, relu)

        Dense (128, relu)

        Dense (784, sigmoid)

        Reshape (28, 28)

    Combine encoder + decoder -> autoencoder

    Compile autoencoder (adam, binary - crossentropy)

    Train on MNIST data (epochs = 50, batch-size = 256)

    Generate reconstructed images

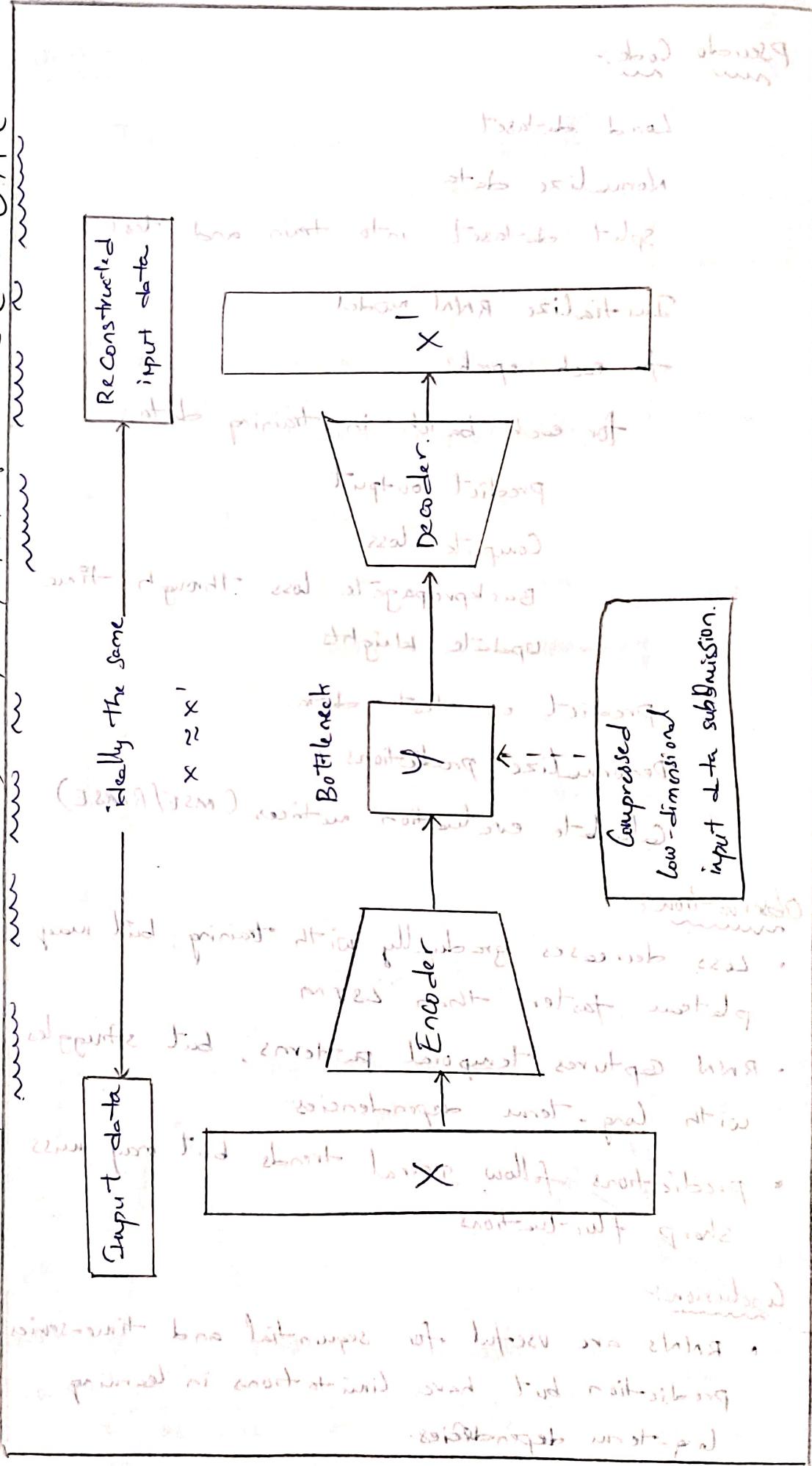   Display original & reconstructed samples

Observation:-

- The Autoencoder successfully Compresses 784 feature into a 32-dimensional talent space

- The reconstructed images closely resemble the original MNIST digits, though with slight bluring due to compression loss.

- Training loss decreases steadily with epochs, indicating good learning

Conclusion:-

Autoencoders efficiently learn compressed feature representations of images without supervision The MNIST Autoencoder achieves dimensionality reduction while maintaining the essential structure of the digits, demonstrating effective unsupervised Compression

AUTO ENCODER ARCHITECTURE

Input data

Encoder

Bottleneck

y

Compressed low-dimensional input data subdivision.

Decoder

$\bar{X}$

Reconstructed input data

ideally the same
$X \approx X'$
(MSE/BCE)

X

Output:

Simple AE | Epoch [1/5], Loss : 0.0501

Simple AE | Epoch [2/5], Loss : 0.0211

Simple AE | Epoch [3/5], Loss : 0.0154

Simple AE | Epoch [4/5], Loss : 0.0128

Simple AE | Epoch [5/5], Loss : 0.0114

Deep AE | Epoch [1/5], Loss: 0.0445

Deep AE | Epoch [2/5], Loss: 0.0179

Deep AE | Epoch [3/5], Loss : 0.0150

Deep AE | Epoch [4/5], Loss : 0.0124

Deep AE | Epoch [5/5], Loss : 0.0108

Training Loss Comparision

— Simple AE
--- Deep AE