Lab : 10
09-10-25

PERFORM COMPRESSION ON MNIST DATASET USING AUTOENCODER.

AIM

To implement an autoencoder for compressing and reconstructing images from the MNIST dataset.

Objectives :

1. To understand the concept and working of autoencoders
2. To perform image compression using the encoder-decoder.
3. To visualize and analyze reconstructed images and the quality
4. To explore dimensionality reduction through neural networks

pseudocode

start

Import necessary libraries (tensorflow/keras, numpy, matplotlib)

Load MNIST dataset

Normalise pixel values (0-1)

flatten images into 784-dimensional vectors

Define autoencoder architecture

Encoder

Input layer (784)

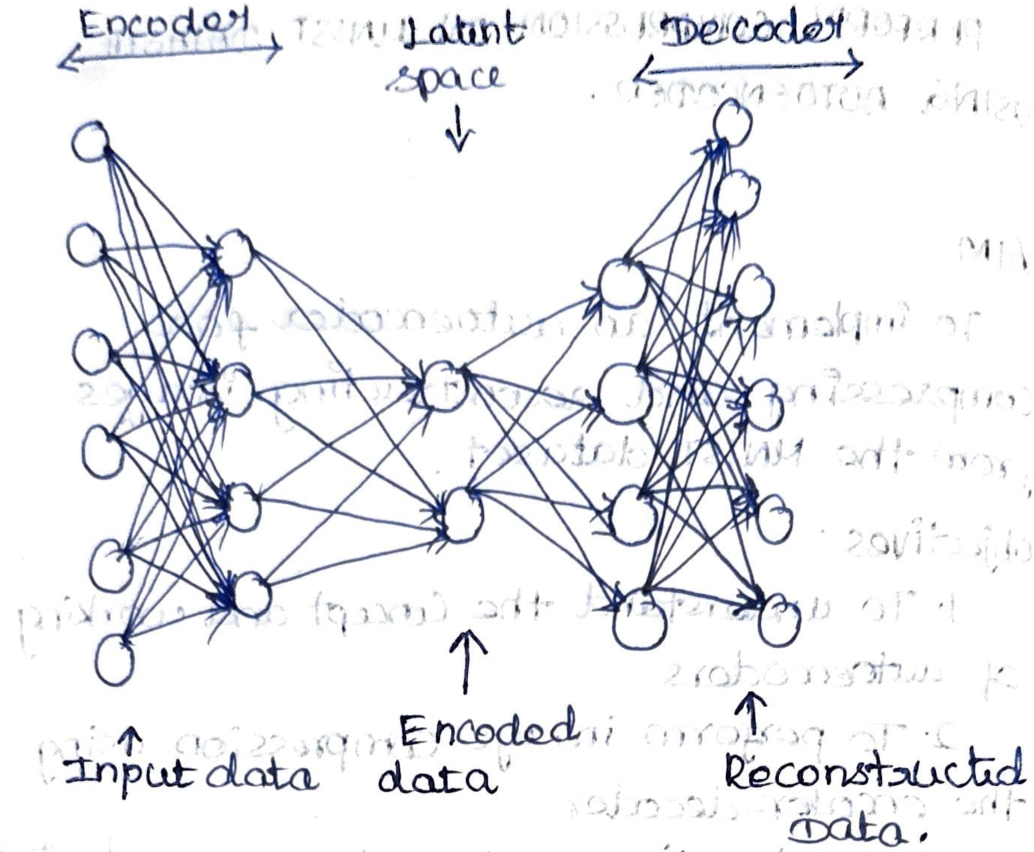
Dense (128, activation = 'relu')

Dense (64, activation = 'relu')

Dense (32, activation = 'relu')

Decoder

Dense (64, activation = 'relu')



Output

Epoch [1/5] , Loss : 0.0621

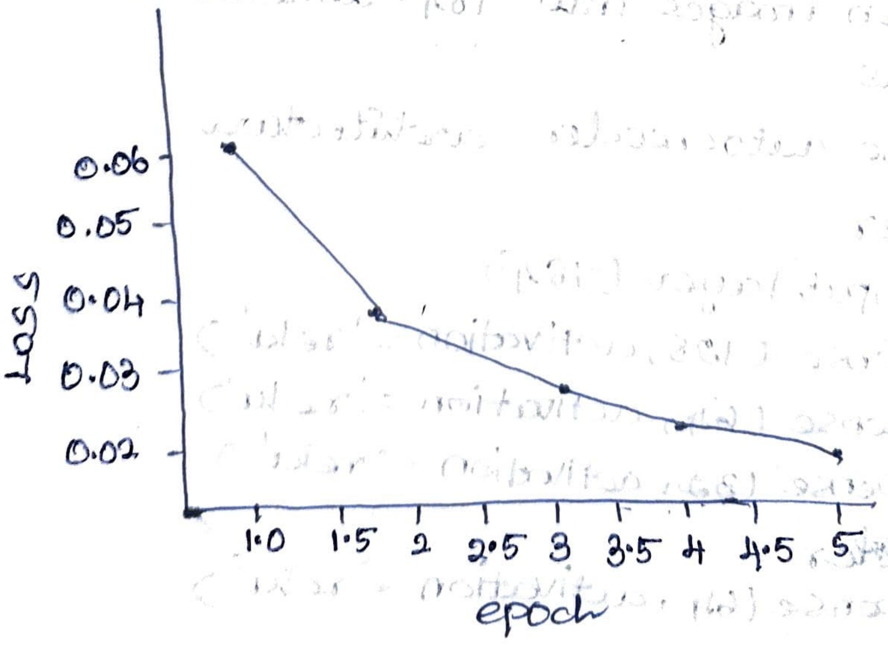
Epoch [2/5] , Loss : 0.0321

Epoch [3,5] , Loss : 0.0249

Epoch [4,5] , Loss : 0.0219

Epoch [5,5] , Loss : 0.0197

Accuracy : 98.17 %



Dense(784, activation ='sigmoid')

compile the model

Loss : MSE

optimizer : Adam

Evaluate model performance on test data.

plot training loss curve

END

Observation

Training Behavior: Autoencoder learned to reconstruct digits over epochs. Training loss decreased steadily.

Compression effect : 784D inputs Compressed to 32D latent space. reconstructed images kept key digit details, with slight loss of sharpness.

Visualization: Orginal vs reconstructed images showed strong similarity. latent space captured distinct digit patterns.

Interpretation: performed non-linear dimensionality reduction. learned features useful for clustering or classification

Result:

Implement perform compression on MNIST dataset using autoencoder.

EX. NO: 11
17.10.25

EXPERIMENTS USING VARIATIONAL AUTO-ENCODERS

Aim

To implement a variational autoencoder and study its generative ability to reconstruct and generate new data samples using the MNIST dataset.

Objectives

To understand the concept and working of a VAE.

To generate new data points by sampling from a latent distribution

To visualize reconstructed and newly generated images.

To compare the performance of VAE with a basic autoencoder.

pseudocode

Start

Import Libraries (tensorflow / keras / numpy, matplotlib).

Load Mnist dataset and normalize images (0-1)

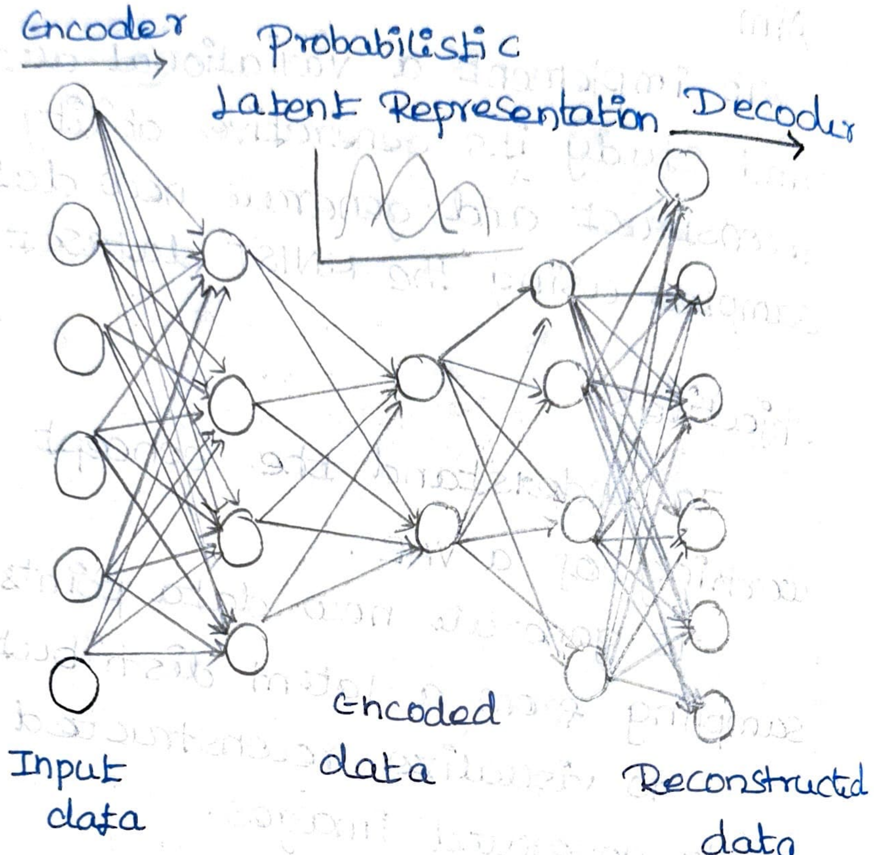
Define encoder

Input layer (784)

Dense (256, activation = relu)

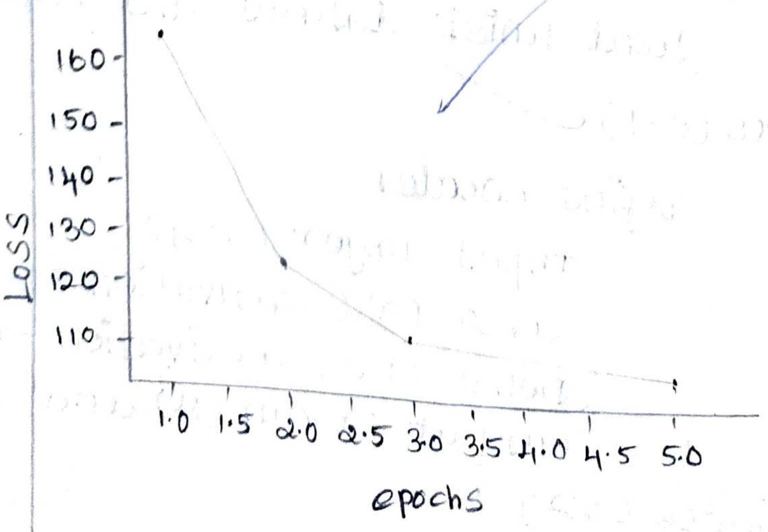
Dense (128, activation = relu)

output : mean (μ) and log variance (σ^2)



Epoch [1/5], Loss: 164.0216
Epoch [2/5], Loss: 121.5716
Epoch [3/5], Loss: 114.6072
Epoch [4,5], Loss: 111.6099
Epoch [5,5], Loss: 109.8843

Epoch Loss



Sample latent vector z using reparameterization.

$z = \mu + \sigma * \epsilon$, where $\epsilon \sim N(0,1)$

Define Decoder

Input : latent vector z

Dense (128, activation = relu)

Dense (256, activation = relu)

Dense (784, activation = sigmoid)

reconstructed output

Define total loss

Total loss = Reconstruction loss + KL divergence loss

compile and train model using MNIST dataset

visualize

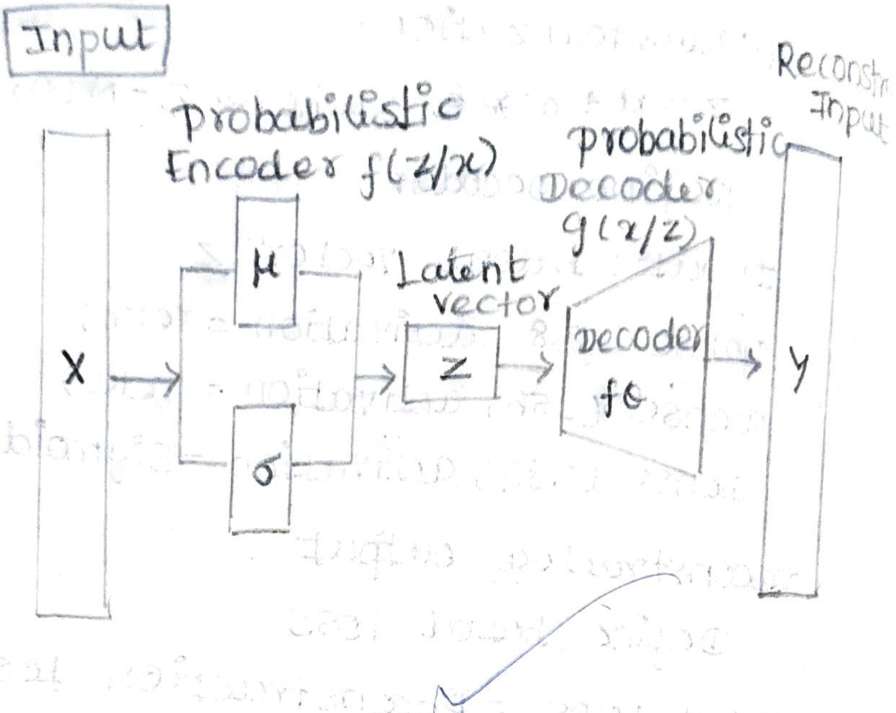
Original vs Reconstructed images

Observation

Training Dynamics

The VAE balanced accurate reconstructed with maintaining a continuous latent space. reconstruction loss decreased gradually while KL divergence regularized the model.

VAE ARCHITECTURE



Latent Space Representation

The Latent space followed a gaussian distribution, where similar digits clustered together, allowing smooth transitions between types

Reconstruction Quality

Reconstructd images were slightly blurred due to sampling but still recognizable, showing that key digit structures were learned.

Generative Ability

Random samples from the latent space produced realistic new digits, proving the VAE's generative capability.

Comparison with autoencoder

unlike basic autoencoder, the VAE learned a probabilistic model and could generate new samples, not just reconstruct inputs.

Result

TO Successfully implement using VAE