Experiment No.5
Design the architecture and implement the autoencoder model for Image Compression.
Date of Performance: 29/02/2024
Date of Submission: 07/03/2024

Aim: Design the architecture and implement the autoencoder model for Image Compression.

Objective: To design deep learning models for unsupervised learning.

Theory:

Autoencoder-

An autoencoder is an unsupervised learning technique for neural networks that learns efficient data
representations (encoding) by training the network to ignore signal "noise."

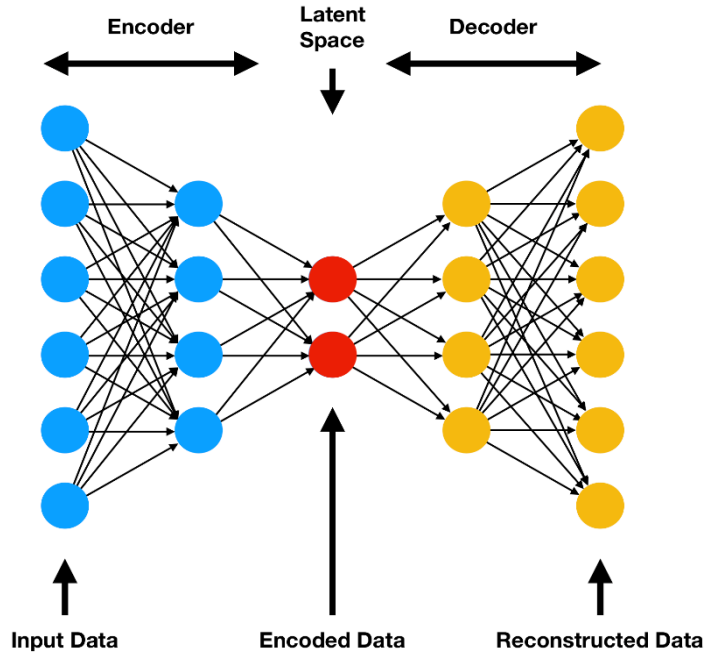
Autoencoders can be used for image denoising, image compression, and, in some cases, even generation of image data.

Autoencoders are a type of neural network that can be used for image compression and reconstruction. The process involves compressing an image into a smaller representation and then reconstructing it back to its original form. Image reconstruction is the process of creating an image from compressed data.

The compressed data can be thought of as a compressed version of the original image. To reconstruct the image, the compressed data is fed through a decoder network, which expands the data back to its original size. The reconstructed image will not be identical to the original, but it will be a close approximation.

Autoencoders use a loss function to determine how well the reconstructed image matches the original. The loss function calculates the difference between the reconstructed image and the original image. The goal of the autoencoder is to minimize the loss function so that the reconstructed image is as close to the original as possible.

Architecture



Code:

```
from keras.datasets import mnist
import numpy as np
(train_images,__),(test_images,_)=mnist.load_data()
print(train_images.shape)
print(test_images.shape)


import matplotlib.pyplot as plt
plt.imshow(test_images[0].reshape(28,28))
plt.gray()

train_images=train_images.astype('float32')/255.
test_images=test_images.astype('float32')/255.


train_images=train_images.reshape((len(train_images),np.prod(train_images.shape[1:])))
test_images=test_images.reshape((len(test_images),np.prod(test_images.shape[1:])))
print(train_images.shape)
print(train_images.shape)


from keras.layers import Input,Dense
from keras.models import Model
encoding_dim=32
input_layer=Input(shape=(784,))
encoder_layer1=Dense(encoding_dim,activation='relu')(input_layer)
decoder_layer1=Dense(784,activation='sigmoid')(encoder_layer1)
autoencoder=Model(input_layer,decoder_layer1)
autoencoder.summary()


encoder=Model(input_layer,encoder_layer1)

encoded_input=Input(shape=(encoding_dim,))
decoder_layer=autoencoder.layers[-1]
decoder=Model(encoded_input,decoder_layer(encoded_input))

autoencoder.compile(optimizer='adam',loss='binary_crossentropy')

autoencoder.fit(train_images,train_images,
epochs=60,
batch_size=256,
shuffle=True,
validation_data=(test_images,test_images))
```

```
encoded_imgs=encoder.predict(test_images)
print(encoded_imgs.shape)

decoded_imgs=decoder.predict(encoded_imgs)
print(decoded_imgs.shape)

decoded_imgs=decoder.predict(encoded_imgs)
print(decoded_imgs.shape)
```

Output:

Image 1:

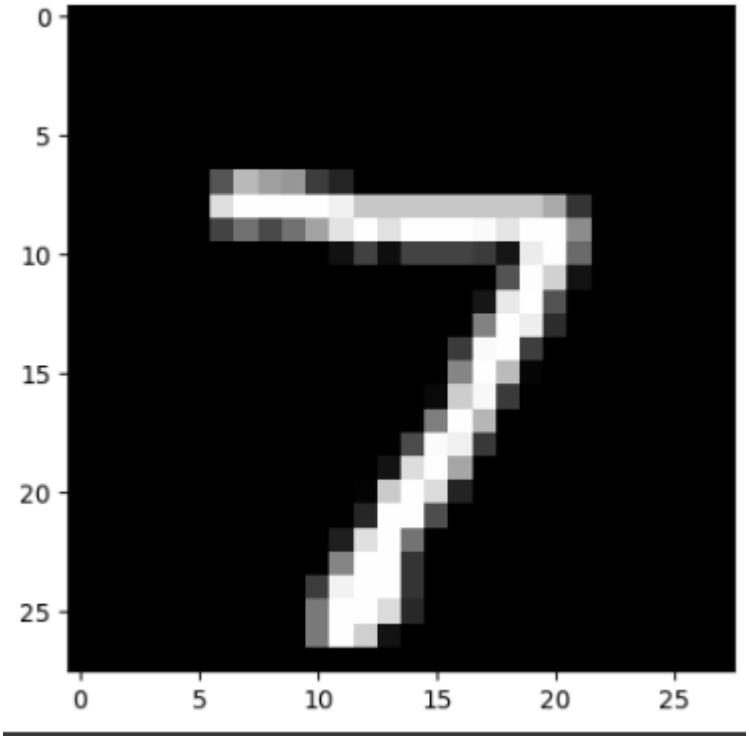
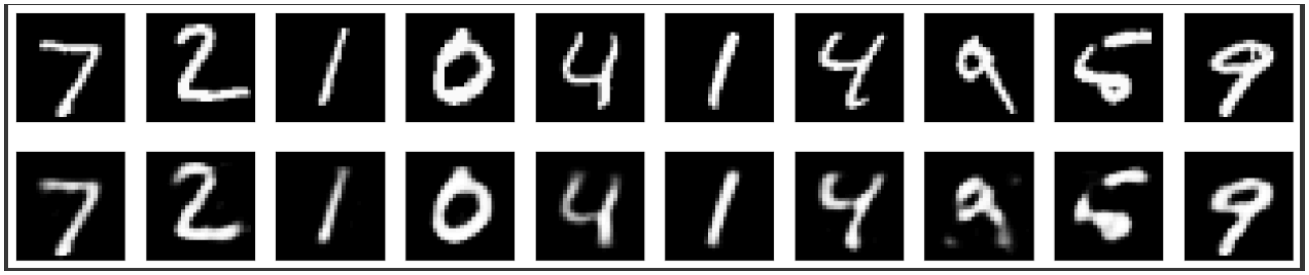


Image 2:



Conclusion: Autoencoders are neural network architectures used for unsupervised learning and dimensionality reduction tasks. By learning to reconstruct input data from a compressed representation, they capture meaningful features and patterns. With applications ranging from data denoising to feature learning, autoencoders are versatile tools in machine learning. However, their performance heavily depends on architecture design, training data quality, and hyperparameter tuning. Despite challenges, autoencoders offer powerful solutions for tasks like anomaly detection, generative modeling, and data visualization.