Experiment No. 5
Design the architecture and implement the autoencoder model for Image Compression
Date of Performance: 04/09/23
Date of Submission: 11/09/23

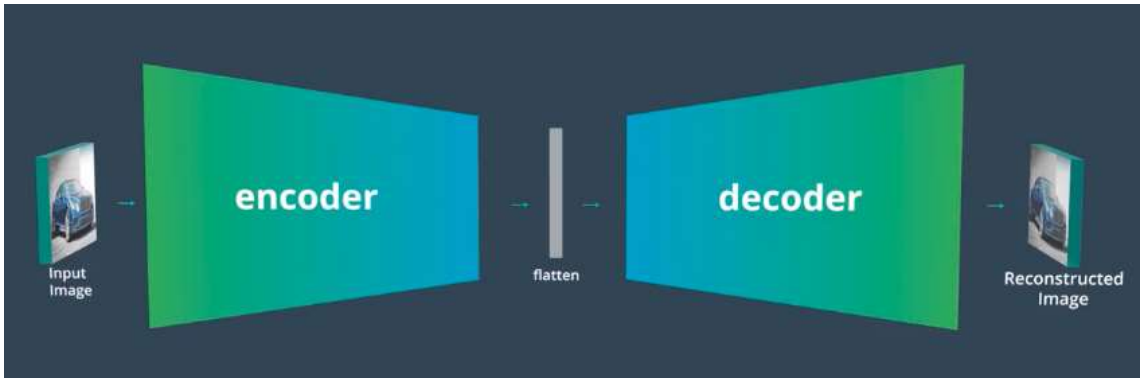
Aim: Design the architecture and implement the autoencoder model for Image Compression.

Objective: Ability to design autoencoder model to solve the given problem.

Theory:

Autoencoders are self-supervised machine learning models which are used to reduce the size of input data by recreating it. These models are trained as supervised machine learning models and during inference, they work as unsupervised models that's why they are called self-supervised models. Autoencoder is made up of two components:

1. Encoder: It works as a compression unit that compresses the input data.
2. Decoder: It decompresses the compressed input by reconstructing it.



n an Autoencoder both Encoder and Decoder are made up of a combination of NN (Neural Networks) layers, which helps to reduce the size of the input image by recreating it. In the case of CNN Autoencoder, these layers are CNN layers (Convolutional, Max Pool, Flattening, etc.) while in the case of RNN/LSTM their respective layers are used.

Applications of Autoencoders

Three are several applications of Autoencoders some of the important ones are:

1. File Compression: Primary use of Autoencoders is that they can reduce the dimensionality of input data which we in common refer to as file compression. Autoencoders works with all kinds of data like Images, Videos, and Audio, this helps in sharing and viewing data faster than we could do with its original file size.

2. Image De-noising: Autoencoders are also used as noise removal techniques (Image De-noising), what makes it the best choice for De-noising is that it does not require any human interaction, once trained on any kind of data it can reproduce that data with less noise than the original image.

3. Image Transformation: Autoencoders are also used for image transformations, which is typically classified under GAN(Generative Adversarial Networks) models. Using these we can transform B/W images to colored one and vice versa, we can up-sample and down-sample the input data, etc.

Code:

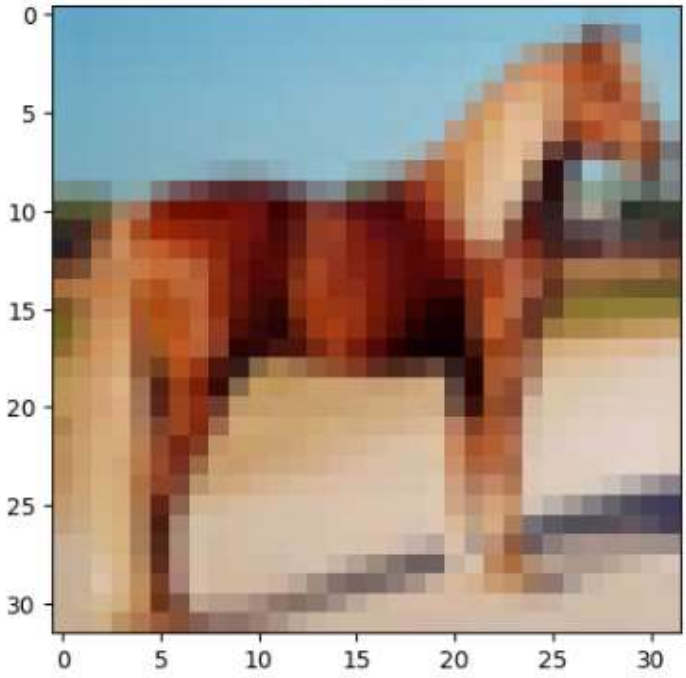
```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers


# Load and preprocess the dataset
(x_train, _), (x_test, _) = keras.datasets.cifar10.load_data()
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0


# Define the autoencoder model
latent_dim = 32
```

```
encoder_inputs = keras.Input(shape=(32, 32, 3))
x = layers.Conv2D(16, kernel_size=3, activation="relu", padding="same")(encoder_inputs)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(8, kernel_size=3, activation="relu", padding="same")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
encoder_outputs = layers.Dense(latent_dim, activation="relu")(x)

encoder = keras.Model(encoder_inputs, encoder_outputs, name="encoder")

latent_inputs = keras.Input(shape=(latent_dim,))
x = layers.Dense(8 * 8 * 8, activation="relu")(latent_inputs)
x = layers.Reshape((8, 8, 8))(x)
x = layers.Conv2DTranspose(8, kernel_size=3, activation="relu", padding="same")(x)
x = layers.UpSampling2D((2, 2))(x)
x = layers.Conv2DTranspose(16, kernel_size=3, activation="relu", padding="same")(x)
x = layers.UpSampling2D((2, 2))(x)
decoder_outputs = layers.Conv2DTranspose(3, kernel_size=3, activation="sigmoid", padding="same")(x)
decoder = keras.Model(latent_inputs, decoder_outputs, name="decoder")
autoencoder = keras.Model(encoder_inputs, decoder(encoder_outputs), name="autoencoder")
autoencoder.summary()
# Compile and train the model
autoencoder.compile(optimizer="adam", loss="binary_crossentropy")
autoencoder.fit(x_train, x_train, epochs=10, batch_size=128, validation_data=(x_test, x_test))
```

Output:

```
autoencoder.imshow("horse")
```

```
<matplotlib.image.AxesImage at 0x7bdab8efb730>
```



Conclusion:

The autoencoder architecture is a type of neural network that is commonly used for unsupervised learning tasks, particularly in the field of image compression. It consists of two main components: an encoder and a decoder.

The encoder takes in the input image and compresses it into a lower-dimensional representation, often referred to as a latent space or code. The encoder's goal is to capture the most important information of the input image in a compact representation.

The decoder, on the other hand, takes the compressed representation produced by the encoder and reconstructs the original image. The decoder's task is to faithfully recreate the original image using the information stored in the compressed representation.