Experiment No.5

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

Code:

```
import keras
from keras import layers
from keras.datasets import mnist
import numpy as np

# Load and preprocess data
(x_train, _), (x_test, _) = mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))

print(x_train.shape)
print(x_test.shape)

# Define the size of the encoded representation
encoding_dim = 32

# Define the model
input_img = keras.Input(shape=(784,))
encoded = layers.Dense(encoding_dim, activation='relu')(input_img)
decoded = layers.Dense(784, activation='sigmoid')(encoded)

autoencoder = keras.Model(input_img, decoded)
encoder = keras.Model(input_img, encoded)

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

# Compile the model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

```
# Train the model
autoencoder.fit(x_train, x_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))
```

Output:

Downloading data from

https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

(60000, 784)

(10000, 784)

Epoch 1/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 0.3849 -

val_loss: 0.1924

Epoch 2/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - loss: 0.1829 -

val_loss: 0.1554

Epoch 3/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.1508 -

val_loss: 0.1343

Epoch 4/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 0.1317 -

val_loss: 0.1198

Epoch 5/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step - loss: 0.1190 -

val_loss: 0.1114

Epoch 6/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 0.1112 -

val_loss: 0.1059
Epoch 7/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - loss: 0.1058 -
val_loss: 0.1017
Epoch 8/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.1021 -
val_loss: 0.0986
Epoch 9/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 0.0993 -
val_loss: 0.0965
Epoch 10/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.0975 -
val_loss: 0.0950
Epoch 11/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 0.0964 -
val_loss: 0.0942
Epoch 12/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - loss: 0.0951 -
val_loss: 0.0934
Epoch 13/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - loss: 0.0948 -
val_loss: 0.0930
Epoch 14/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0944 -
val_loss: 0.0928
Epoch 15/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.0940 -
val_loss: 0.0926
Epoch 16/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - loss: 0.0938 - val_loss: 0.0924
Epoch 17/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - loss: 0.0936 - val_loss: 0.0923
Epoch 18/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0933 - val_loss: 0.0922
Epoch 19/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.0933 - val_loss: 0.0920
Epoch 20/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0932 - val_loss: 0.0920
Epoch 21/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 0.0932 - val_loss: 0.0920
Epoch 22/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step - loss: 0.0931 - val_loss: 0.0920
Epoch 23/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0932 - val_loss: 0.0919
Epoch 24/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 0.0932 - val_loss: 0.0918
Epoch 25/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 0.0932 -

val_loss: 0.0918

Epoch 26/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 0.0930 - val_loss: 0.0918

Epoch 27/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - loss: 0.0929 - val_loss: 0.0917

Epoch 28/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.0928 - val_loss: 0.0917

Epoch 29/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0929 - val_loss: 0.0917

Epoch 30/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - loss: 0.0927 - val_loss: 0.0916

Epoch 31/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - loss: 0.0927 - val_loss: 0.0916

Epoch 32/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.0927 - val_loss: 0.0916

Epoch 33/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0928 - val_loss: 0.0916

Epoch 34/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - loss: 0.0926 - val_loss: 0.0916

Epoch 35/50

235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - loss: 0.0926 - val_loss: 0.0915
Epoch 36/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0928 - val_loss: 0.0916
Epoch 37/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 0.0929 - val_loss: 0.0915
Epoch 38/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0929 - val_loss: 0.0915
Epoch 39/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 0.0925 - val_loss: 0.0915
Epoch 40/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - loss: 0.0928 - val_loss: 0.0915
Epoch 41/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.0927 - val_loss: 0.0915
Epoch 42/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0929 - val_loss: 0.0915
Epoch 43/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - loss: 0.0926 - val_loss: 0.0915
Epoch 44/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 0.0926 -

val_loss: 0.0915
Epoch 45/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 0.0925 -
val_loss: 0.0915
Epoch 46/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 0.0927 -
val_loss: 0.0915
Epoch 47/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - loss: 0.0925 -
val_loss: 0.0915
Epoch 48/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 0.0924 -
val_loss: 0.0914
Epoch 49/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.0928 -
val_loss: 0.0915
Epoch 50/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - loss: 0.0925 -
val_loss: 0.0915
<keras.src.callbacks.history.History at 0x7c59d720fca0>

Code:

```
# Encode some digits from the test set
encoded_imgs = encoder.predict(x_test)

# Decode the encoded images
decoded_imgs = decoder.predict(encoded_imgs)
```

Output:

313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step

313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step

Code:

```
import matplotlib.pyplot as plt

n = 10  # Number of digits to display

plt.figure(figsize=(20, 4))

for i in range(n):
    # Display original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display reconstruction
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

plt.show()
```

Output: