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
(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)
```

OUTPUT:-


```
encoding_dim = 32
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))
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
autoencoder.fit(x_train, x_train, epochs=10, batch_size=64, shuffle=True,
validation_data=(x_test, x_test))
```

```
Epoch 1/10
938/938 [==============================] - 5s 4ms/step - loss: 0.1903 - val_loss: 0.1319
Epoch 2/10
938/938 [==============================] - 3s 3ms/step - loss: 0.1179 - val_loss: 0.1060
Epoch 3/10
938/938 [==============================] - 3s 3ms/step - loss: 0.1022 - val_loss: 0.0974
Epoch 4/10
938/938 [==============================] - 4s 4ms/step - loss: 0.0968 - val_loss: 0.0941
Epoch 5/10
938/938 [==============================] - 3s 3ms/step - loss: 0.0950 - val_loss: 0.0931
Epoch 6/10
938/938 [==============================] - 3s 3ms/step - loss: 0.0943 - val_loss: 0.0929
Epoch 7/10
938/938 [==============================] - 3s 3ms/step - loss: 0.0940 - val_loss: 0.0927
Epoch 8/10
938/938 [==============================] - 4s 4ms/step - loss: 0.0938 - val_loss: 0.0924
Epoch 9/10
938/938 [==============================] - 3s 3ms/step - loss: 0.0936 - val_loss: 0.0924
Epoch 10/10
```

```python
encoded_imgs = encoder.predict(x_test)
decoded_imgs = decoder.predict(encoded_imgs)
import matplotlib.pyplot as plt

n = 10  # How many digits we will 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()
```