Pritesh Dhumal Roll No-11 Deep Learning(CSE(DS)) Exp_05

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

```
(x_train, _), (x_test, _) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

```
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)
```

```
(60000, 784)
(10000, 784)
```

```
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 = 20,
                batch_size = 64,
                shuffle = True,
                validation_data = (x_test, x_test))
```

```
Epoch 1/20
938/938 [==============================] - 7s 6ms/step - loss: 0.1909 - val_loss: 0.1325
Epoch 2/20
938/938 [==============================] - 7s 7ms/step - loss: 0.1191 - val_loss: 0.1075
Epoch 3/20
938/938 [==============================] - 6s 7ms/step - loss: 0.1031 - val_loss: 0.0978
Epoch 4/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0972 - val_loss: 0.0946
Epoch 5/20
938/938 [==============================] - 6s 6ms/step - loss: 0.0953 - val_loss: 0.0937
Epoch 6/20
938/938 [==============================] - 7s 8ms/step - loss: 0.0945 - val_loss: 0.0931
Epoch 7/20
938/938 [==============================] - 5s 6ms/step - loss: 0.0941 - val_loss: 0.0928
Epoch 8/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0939 - val_loss: 0.0926
Epoch 9/20
938/938 [==============================] - 6s 6ms/step - loss: 0.0937 - val_loss: 0.0923
Epoch 10/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0936 - val_loss: 0.0923
Epoch 11/20
938/938 [==============================] - 6s 6ms/step - loss: 0.0935 - val_loss: 0.0922
Epoch 12/20
```

```
938/938 [==============================] - 7s 8ms/step - loss: 0.0935 - val_loss: 0.0923
Epoch 13/20
938/938 [==============================] - 5s 6ms/step - loss: 0.0934 - val_loss: 0.0922
Epoch 14/20
938/938 [==============================] - 7s 8ms/step - loss: 0.0934 - val_loss: 0.0922
Epoch 15/20
938/938 [==============================] - 5s 6ms/step - loss: 0.0933 - val_loss: 0.0921
Epoch 16/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0933 - val_loss: 0.0921
Epoch 17/20
938/938 [==============================] - 5s 6ms/step - loss: 0.0933 - val_loss: 0.0921
Epoch 18/20
938/938 [==============================] - 7s 8ms/step - loss: 0.0932 - val_loss: 0.0921
Epoch 19/20
938/938 [==============================] - 6s 6ms/step - loss: 0.0932 - val_loss: 0.0920
Epoch 20/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0932 - val_loss: 0.0921
<keras.callbacks.History at 0x7b4a3b247520>
```

```
encoded_imgs = encoder.predict(x_test)
decoded_imgs = decoder.predict(encoded_imgs)
```

```
313/313 [==============================] - 1s 1ms/step
313/313 [==============================] - 1s 2ms/step
```

```
# Use Matplotlib (don't ask)
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()
```