Name: Prajjwal Vishwakarma

Roll No.: 63

Sub.: Deep Learning

Experiment No.:5

```
import keras
from keras import layers
from keras.datasets import
mnistimport 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 [==============================] - 6s 5ms/step - loss: 0.1910
```

```
- val_loss:
0.1338Epoch
2/20
938/938 [==============================] - 6s 6ms/step - loss: 0.1201
- val_loss:
0.1083Epoch
3/20
938/938 [==============================] - 7s 7ms/step - loss: 0.1038
- val_loss:
0.0983Epoch
4/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0974
- val_loss: 0.0948
```

```
Epoch 5/20
938/938 [==============================] - 7s 8ms/step - loss: 0.0954
- val_loss:
0.0936Epoch
6/20
938/938 [==============================] - 6s 6ms/step - loss: 0.0947
- val_loss:
0.0931Epoch
7/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0943
- val_loss:
0.0930Epoch
8/20
938/938 [==============================] - 9s 9ms/step - loss: 0.0941
- val_loss:
0.0928Epoch
9/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0939
- val_loss:
0.0926Epoch
10/20
938/938 [==============================] - 6s 6ms/step - loss: 0.0938
- val_loss:
0.0925Epoch
11/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0937
- val_loss:
0.0925Epoch
12/20
938/938 [==============================] - 5s 6ms/step - loss: 0.0936
- val_loss:
0.0923Epoch
13/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0936
```

```
- val_loss:
0.0922Epoch
14/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0935
- val_loss:
0.0923Epoch
15/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0935
- val_loss:
0.0924Epoch
16/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0934
- val_loss:
0.0921Epoch
17/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0934
- val_loss:
0.0923Epoch
18/20
938/938 [==============================] - 7s 7ms/step - loss: 0.0933
- val_loss:
0.0922Epoch
19/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0933
- val_loss:
0.0922Epoch
20/20
938/938 [==============================] - 6s 6ms/step - loss: 0.0933
- val_loss: 0.0921
```

```
<keras.callbacks.History at 0x7da7bd3e6140>
```

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

```
313/313 [==============================] - 0s 1ms/step
313/313 [==============================] - 0s 1ms/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()
```