```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras import layers
import numpy as np
```

```
(X_train,_),(X_test,_) = keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
```

```
X_train[0]
```

```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
         90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
        190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
        253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
        241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
```

```
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:])))
```

```
X_train = X_train/255
X_test = X_test/255
```

```
print(X_train.shape)

     (60000, 784)
```

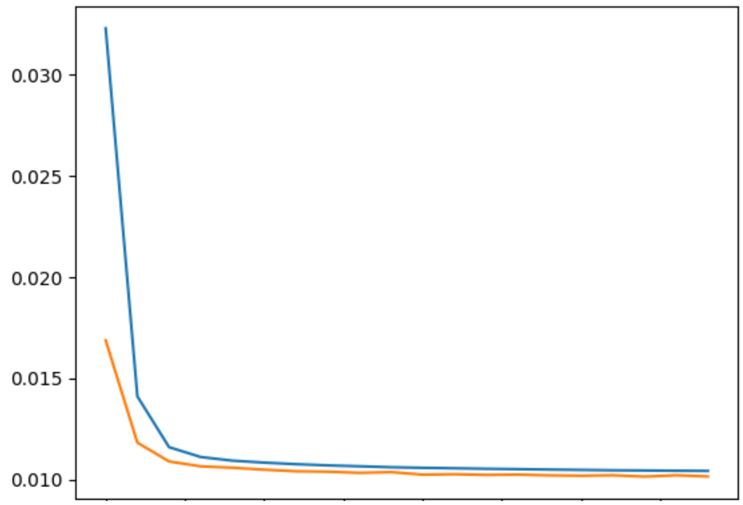
```
model = Sequential()
# Add the hidden layer with the specified input
model.add(Dense(32, activation='relu', input_shape=(784,)))

# Add the output layer
model.add(Dense(784, activation='sigmoid'))
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
model.summary()

     Model: "sequential"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #   
     =================================================================
      dense (Dense)               (None, 32)                25120     
                                                                      
      dense_1 (Dense)             (None, 784)               25872     
                                                                      
     =================================================================
     Total params: 50,992
     Trainable params: 50,992
     Non-trainable params: 0
     _________________________________________________________________
```

```
history = model.fit(X_train,X_train,epochs=20,validation_data=(X_test,X_test))

     Epoch 1/20
     1875/1875 [==============================] - 7s 4ms/step - loss: 0.0323 - val_loss: 0.0169
     Epoch 2/20
     1875/1875 [==============================] - 11s 6ms/step - loss: 0.0141 - val_loss: 0.0118
     Epoch 3/20
     1875/1875 [==============================] - 11s 6ms/step - loss: 0.0116 - val_loss: 0.0109
     Epoch 4/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0111 - val_loss: 0.0106
     Epoch 5/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0109 - val_loss: 0.0106
     Epoch 6/20
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.0108 - val_loss: 0.0105
     Epoch 7/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0107 - val_loss: 0.0104
     Epoch 8/20
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.0107 - val_loss: 0.0104
     Epoch 9/20
     1875/1875 [==============================] - 7s 3ms/step - loss: 0.0106 - val_loss: 0.0103
     Epoch 10/20
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.0106 - val_loss: 0.0104
     Epoch 11/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0106 - val_loss: 0.0102
     Epoch 12/20
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.0105 - val_loss: 0.0103
     Epoch 13/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0105 - val_loss: 0.0102
     Epoch 14/20
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.0105 - val_loss: 0.0102
     Epoch 15/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0105 - val_loss: 0.0102
     Epoch 16/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0105 - val_loss: 0.0102
     Epoch 17/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0104 - val_loss: 0.0102
     Epoch 18/20
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.0104 - val_loss: 0.0101
     Epoch 19/20
     1875/1875 [==============================] - 5s 3ms/step - loss: 0.0104 - val_loss: 0.0102
     Epoch 20/20
     1875/1875 [==============================] - 8s 4ms/step - loss: 0.0104 - val_loss: 0.0101
```

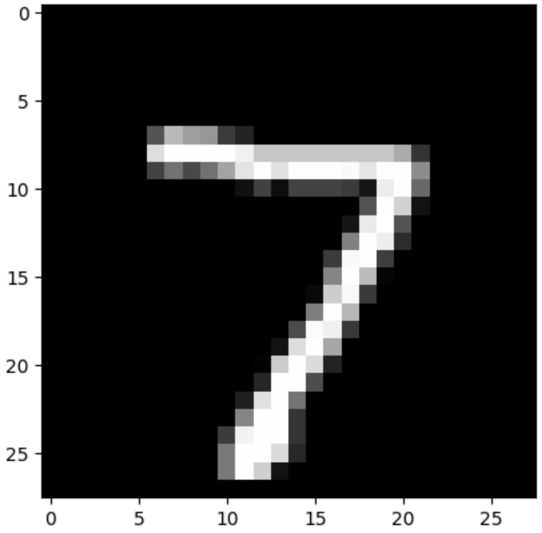
```
import matplotlib.pyplot as plt
# plotting loss vs val_loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x7b1e947176d0>]
```



```
orgImg = X_test.reshape((10000, 28, 28))
print(orgImg.shape)
plt.imshow(orgImg[0]*255)
plt.gray()
```

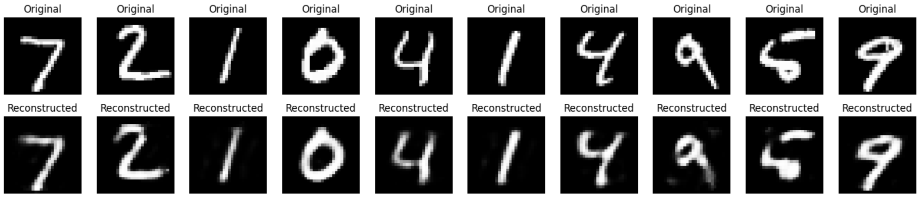
```
(10000, 28, 28)
```



```
encoded_imgs = model.predict(X_test)
```

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

```
n = 10
plt.figure(figsize=(20,4))

for i in range(n):
  # Original
  ax = plt.subplot(2,n,i+1)
  plt.imshow(X_test[i].reshape(28, 28), cmap='gray')
  plt.gray()
  plt.title("Original")
  plt.axis('off')

  # reconstruction
  ax = plt.subplot(2,n,i+1+n)
  plt.imshow(encoded_imgs[i].reshape(28,28))
  plt.gray()
  plt.title("Reconstructed")
  plt.axis('off')
plt.show()
```


Original
Original
Original
Original
Original
Original
Original
Original
Original
Original
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed

stochastic