```
import numpy as np
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
```

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

x_train = x_train.astype('float32') / 255.0
x_train = x_train.reshape((len(x_train), 784))
x_test = x_test.astype('float32') / 255.0
x_test = x_test.reshape((len(x_test), 784))
```

```
encoding_dim = 32

input_img = Input(shape=(784,))

# Build the encoder
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded_output = Dense(encoding_dim, activation='relu')(encoded)

# Build the decoder
decoded = Dense(64, activation='relu')(encoded_output)
decoded = Dense(128, activation='relu')(decoded)
decoded_output = Dense(784, activation='sigmoid')(decoded)

autoencoder = Model(input_img, decoded_output)
```

```
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

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

```
Epoch 1/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 7s 18ms/step - loss: 0.3352 - val_loss: 0.1638
Epoch 2/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 35ms/step - loss: 0.1549 - val_loss: 0.1342
Epoch 3/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 0.1324 - val_loss: 0.1239
Epoch 4/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 20ms/step - loss: 0.1239 - val_loss: 0.1186
Epoch 5/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - loss: 0.1183 - val_loss: 0.1135
Epoch 6/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - loss: 0.1137 - val_loss: 0.1098
Epoch 7/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 22ms/step - loss: 0.1101 - val_loss: 0.1066
Epoch 8/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - loss: 0.1071 - val_loss: 0.1039
Epoch 9/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - loss: 0.1046 - val_loss: 0.1018
Epoch 10/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 22ms/step - loss: 0.1025 - val_loss: 0.1005
Epoch 11/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 19ms/step - loss: 0.1010 - val_loss: 0.0986
Epoch 12/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - loss: 0.0992 - val_loss: 0.0980
Epoch 13/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 24ms/step - loss: 0.0981 - val_loss: 0.0962
Epoch 14/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - loss: 0.0965 - val_loss: 0.0949
Epoch 15/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - loss: 0.0957 - val_loss: 0.0941
Epoch 16/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 23ms/step - loss: 0.0945 - val_loss: 0.0933
Epoch 17/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - loss: 0.0937 - val_loss: 0.0923
Epoch 18/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 18ms/step - loss: 0.0930 - val_loss: 0.0917
Epoch 19/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - loss: 0.0922 - val_loss: 0.0909
Epoch 20/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - loss: 0.0914 - val_loss: 0.0909
```

```
Epoch 21/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 22ms/step - loss: 0.0911 - val_loss: 0.0900
Epoch 22/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 16ms/step - loss: 0.0908 - val_loss: 0.0894
Epoch 23/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 7s 23ms/step - loss: 0.0901 - val_loss: 0.0891
Epoch 24/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 17ms/step - loss: 0.0897 - val_loss: 0.0889
Epoch 25/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - loss: 0.0892 - val_loss: 0.0883
Epoch 26/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - loss: 0.0887 - val_loss: 0.0880
Epoch 27/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - loss: 0.0887 - val_loss: 0.0878
Epoch 28/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - loss: 0.0883 - val_loss: 0.0875
Epoch 29/50
```

```
# separate encoder model
encoder = Model(input_img, encoded_output)

# Encode and decode digits
encoded_imgs = encoder.predict(x_test)
decoded_imgs = autoencoder.predict(x_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

```
import matplotlib.pyplot as plt

n = 10  # Number of digits to display
plt.figure(figsize=(20, 4))

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

    #  reconstructed images
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28), cmap='gray')
    plt.title("Reconstructed")
    plt.axis('off')

plt.show()
```