```python
import tensorflow as tf
from keras.layers import Input, Dense, Flatten, Reshape, Conv2D, Conv2DTranspose, UpSampling2D
from keras.models import Model
from keras.datasets import mnist

(x_train, _), (x_test, _) = mnist.load_data()
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]


# Encoder
input_img = Input(shape=(28, 28, 1))

x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = Flatten()(x)

# Bottleneck layer
encoded = Dense(16, activation='relu')(x)

# Decoder
x = Dense(7 * 7 * 64, activation='relu')(encoded)
x = Reshape((7, 7, 64))(x)
x = Conv2DTranspose(64, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x) # Upsample to (14, 14)
x = Conv2DTranspose(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x) # Upsample to (28, 28)
decoded = Conv2DTranspose(1, (3, 3), activation='sigmoid', padding='same')(x)

# Create the autoencoder model
autoencoder = Model(input_img, decoded)

# Compile the model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

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

# Evaluate the autoencoder
loss = autoencoder.evaluate(x_test, x_test)
print(f'Test Loss: {loss}')

# Save the model
autoencoder.save('autoencoder_mnist.h5')
```

```
Epoch 39/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - loss: 0.1192 - val_loss: 0.1234
Epoch 40/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 18ms/step - loss: 0.1187 - val_loss: 0.1237
Epoch 41/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1184 - val_loss: 0.1236
Epoch 42/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1184 - val_loss: 0.1236
Epoch 43/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1185 - val_loss: 0.1237
Epoch 44/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - loss: 0.1178 - val_loss: 0.1244
Epoch 45/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1183 - val_loss: 0.1240
Epoch 46/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1177 - val_loss: 0.1238
Epoch 47/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 16ms/step - loss: 0.1176 - val_loss: 0.1240
Epoch 48/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1176 - val_loss: 0.1237
Epoch 49/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step - loss: 0.1172 - val_loss: 0.1244
Epoch 50/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - loss: 0.1173 - val_loss: 0.1239
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1242
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format
Test Loss: 0.1238531768321991
```

```
1 autoencoder.save('autoencoder.keras')
```

```
1 import numpy as np
2
3 decoded_imgs = autoencoder.predict(x_test)
4
```

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

```
1 import matplotlib.pyplot as plt
2
3 def plot_images(original_images, reconstructed_images, num_images=10):
4     plt.figure(figsize=(20, 4))
5     for i in range(num_images):
6
7         ax = plt.subplot(2, num_images, i + 1)
8         plt.imshow(original_images[i].reshape(28, 28), cmap='gray')
9         plt.title("Original")
10        plt.gray()
11        plt.axis('off')
12
13        ax = plt.subplot(2, num_images, i + 1 + num_images)
14        plt.imshow(reconstructed_images[i].reshape(28, 28), cmap='gray')
15        plt.title("Reconstructed")
16        plt.gray()
17        plt.axis('off')
18
19    plt.show()
20
21 plot_images(x_test, decoded_imgs)
22
```