```
# Import the modules
import numpy as np
import tensorflow as tf  # or import torch for PyTorch
from sklearn.model_selection import train_test_split


# Load the MNIST dataset,
# Which consists of 28x28 pixel grayscale images of handwritten digits.
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
```

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

```
# Flatten the images and normalize the pixel values to the range [0, 1].
X_train = X_train.reshape((X_train.shape[0], -1)) / 255.0
X_test = X_test.reshape((X_test.shape[0], -1)) / 255.0


# Split the data into training and testing sets.
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)


# Create a simple neural network using a framework like TensorFlow or PyTorch.
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(784,)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])



# Compile the model with an appropriate loss function, optimizer, and metrics.
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])


# Train the model using the training data.
model.fit(X_train, y_train, epochs=5, validation_data=(X_val, y_val))
```

```
Epoch 1/5
1500/1500 [==============================] - 8s 5ms/step - loss: 0.3219 - accuracy: 0.9073 - val_loss: 0.1651 - val_accu
Epoch 2/5
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1548 - accuracy: 0.9533 - val_loss: 0.1234 - val_accu
Epoch 3/5
1500/1500 [==============================] - 7s 5ms/step - loss: 0.1171 - accuracy: 0.9653 - val_loss: 0.1007 - val_accu
Epoch 4/5
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0944 - accuracy: 0.9706 - val_loss: 0.0960 - val_accu
Epoch 5/5
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0785 - accuracy: 0.9754 - val_loss: 0.0842 - val_accu
<keras.src.callbacks.History at 0x7dd267f25de0>
```

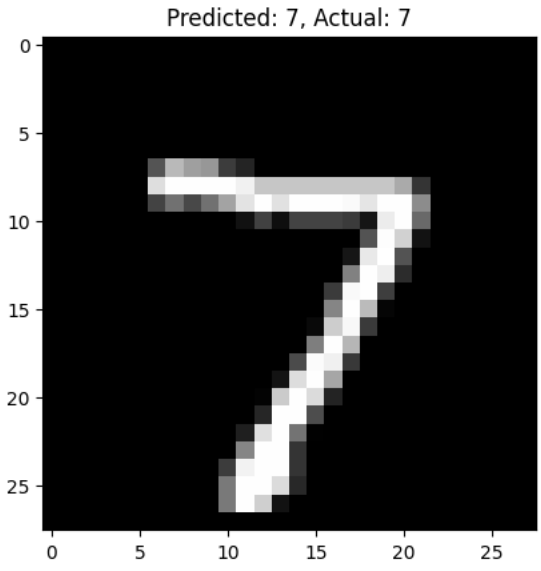
```
# Evaluate the model on the test set to see how well it generalizes.
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_acc}')
```

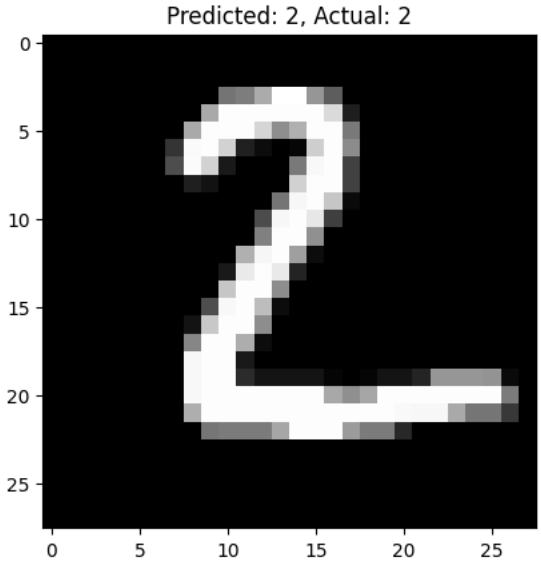
```
313/313 [==============================] - 1s 2ms/step - loss: 0.0793 - accuracy: 0.9753
Test Accuracy: 0.9753000140190125
```

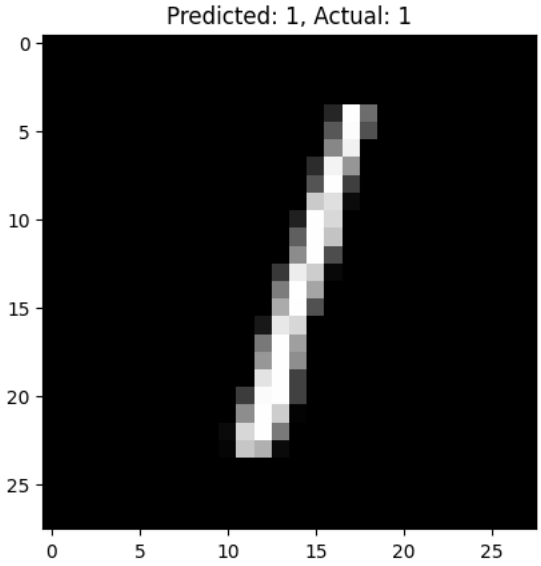
```
# Use the trained model to make predictions on new data.
predictions = model.predict(X_test[:5])
```

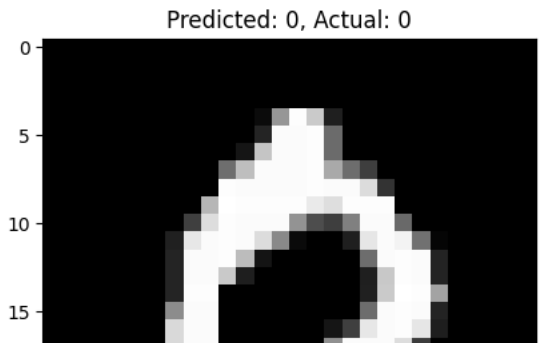
```
1/1 [==============================] - 0s 110ms/step
```

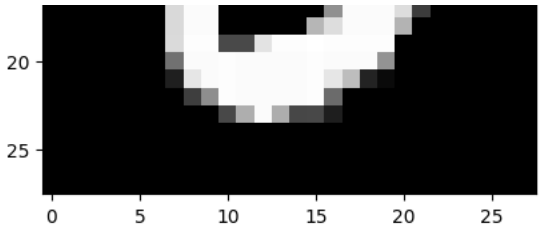
```
# Optionally, visualize the model's predictions and check how well it performs on sample images.

import matplotlib.pyplot as plt

for i in range(5):
    plt.imshow(X_test[i].reshape(28, 28), cmap='gray')
    plt.title(f'Predicted: {np.argmax(predictions[i])}, Actual: {y_test[i]}')
    plt.show()
```


Predicted: 7, Actual: 7


Predicted: 2, Actual: 2


Predicted: 1, Actual: 1


Predicted: 0, Actual: 0



Predicted: 4, Actual: 4