```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Normalize and reshape the data
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

# Add a channel dimension to the images (required for CNN)
train_images = np.expand_dims(train_images, axis=-1)
test_images = np.expand_dims(test_images, axis=-1)

# One-hot encode the labels
num_classes = 10
train_labels = to_categorical(train_labels, num_classes)
test_labels = to_categorical(test_labels, num_classes)

def create_model(input_shape, num_classes):
    model = Sequential([
        Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape),
        MaxPooling2D(pool_size=(2, 2)),
        Conv2D(64, kernel_size=(3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])
    return model

input_shape = (28, 28, 1)
model = create_model(input_shape, num_classes)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
epochs = 10
batch_size = 128

history = model.fit(train_images, train_labels,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(test_images, test_labels))

test_loss, test_accuracy = model.evaluate(test_images, test_labels, verbose=0)
print(f'Test accuracy: {test_accuracy:.4f}')

# Plot training and validation accuracy over epochs
plt.plot(history.history['accuracy'], label="Training Accuracy")
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
Epoch 1/10
469/469 [==============================] - 57s 117ms/step - loss: 0.2868 - accuracy: 0.9125 - val_loss: 0.0576 - val_accuracy: 0.98
Epoch 2/10
469/469 [==============================] - 46s 98ms/step - loss: 0.0973 - accuracy: 0.9710 - val_loss: 0.0404 - val_accuracy: 0.986
Epoch 3/10
469/469 [==============================] - 48s 103ms/step - loss: 0.0713 - accuracy: 0.9784 - val_loss: 0.0341 - val_accuracy: 0.98
Epoch 4/10
469/469 [==============================] - 47s 100ms/step - loss: 0.0572 - accuracy: 0.9827 - val_loss: 0.0310 - val_accuracy: 0.98
Epoch 5/10
469/469 [==============================] - 56s 120ms/step - loss: 0.0486 - accuracy: 0.9853 - val_loss: 0.0343 - val_accuracy: 0.98
Epoch 6/10
469/469 [==============================] - 46s 97ms/step - loss: 0.0430 - accuracy: 0.9869 - val_loss: 0.0291 - val_accuracy: 0.989
Epoch 7/10
469/469 [==============================] - 48s 102ms/step - loss: 0.0385 - accuracy: 0.9887 - val_loss: 0.0264 - val_accuracy: 0.99
Epoch 8/10
469/469 [==============================] - 47s 99ms/step - loss: 0.0335 - accuracy: 0.9900 - val_loss: 0.0257 - val_accuracy: 0.991
Epoch 9/10
469/469 [==============================] - 45s 97ms/step - loss: 0.0322 - accuracy: 0.9899 - val_loss: 0.0261 - val_accuracy: 0.992
Epoch 10/10
469/469 [==============================] - 47s 100ms/step - loss: 0.0266 - accuracy: 0.9918 - val_loss: 0.0251 - val_accuracy: 0.99
Test accuracy: 0.9922
```