```python
# 📦 Importing required libraries
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist

# 📥 Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# 🔄 Normalize the images
x_train = x_train / 255.0
x_test = x_test / 255.0

# 🔄 Reshape to fit CNN input
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# 🏗 Build the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28,
1)),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')  # 10 classes for digits 0-
9
])

# ⚙ Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 🏋 Train the model
model.fit(x_train, y_train, epochs=5, validation_data=(x_test,
y_test))

# 📊 Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f"\n🎯 Test Accuracy: {test_accuracy:.4f}")

# 💾 Save the model
model.save("digit_recognition_model.h5")

# 🔍 Predict on test data
predictions = model.predict(x_test)
```

```python
# 🔍 Display one sample of each digit (0-9)
shown_digits = set()
plt.figure(figsize=(15, 5))
i = 0
count = 0
while len(shown_digits) < 10 and i < len(x_test):
    label = y_test[i]
    if label not in shown_digits:
        plt.subplot(2, 5, count + 1)
        plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
        plt.title(f"Digit: {label} | Predicted:
{np.argmax(predictions[i])}")
        plt.axis('off')
        shown_digits.add(label)
        count += 1
    i += 1

plt.suptitle("Sample of each digit from 0 to 9")
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 58s 30ms/step - accuracy: 0.9001 -
loss: 0.3227 - val_accuracy: 0.9845 - val_loss: 0.0486
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 79s 28ms/step - accuracy: 0.9838 -
loss: 0.0517 - val_accuracy: 0.9864 - val_loss: 0.0405
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 28ms/step - accuracy: 0.9902 -
loss: 0.0299 - val_accuracy: 0.9874 - val_loss: 0.0375
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 56s 30ms/step - accuracy: 0.9927 -
loss: 0.0213 - val_accuracy: 0.9888 - val_loss: 0.0353
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 83s 31ms/step - accuracy: 0.9948 -
loss: 0.0178 - val_accuracy: 0.9907 - val_loss: 0.0366
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9878 - loss:
0.0496

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
```

```
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.


 Test Accuracy: 0.9907
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step
```