

```
In [5]: import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import pandas as pd
```

```
In [9]: # Load MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Normalize values between 0 and 1
x_train = x_train / 255.0
x_test = x_test / 255.0

# Reshape for CNN (samples, height, width, channels)
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

print("Training shape:", x_train.shape)
print("Testing shape:", x_test.shape)
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>  
11490434/11490434 [=====] - 3s 0us/step  
Training shape: (60000, 28, 28, 1)  
Testing shape: (10000, 28, 28, 1)

```
In [11]: model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 64)	102464
dense_1 (Dense)	(None, 10)	650
=====		
Total params: 121,930		
Trainable params: 121,930		
Non-trainable params: 0		

```
In [13]: model.compile(
          optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy']
        )
```

```
In [15]: history = model.fit(
          x_train, y_train,
          epochs=5,
          batch_size=64,
          validation_split=0.2
        )
```

Epoch 1/5

750/750 [=====] - 7s 8ms/step - loss: 0.2030 - accuracy: 0.9386 - val\_loss: 0.0763 - val\_accuracy: 0.9763

Epoch 2/5

750/750 [=====] - 6s 8ms/step - loss: 0.0601 - accuracy: 0.9810 - val\_loss: 0.0544 - val\_accuracy: 0.9839

Epoch 3/5

750/750 [=====] - 6s 8ms/step - loss: 0.0411 - accuracy: 0.9872 - val\_loss: 0.0422 - val\_accuracy: 0.9876

Epoch 4/5

750/750 [=====] - 6s 8ms/step - loss: 0.0317 - accuracy: 0.9899 - val\_loss: 0.0398 - val\_accuracy: 0.9882

Epoch 5/5

750/750 [=====] - 6s 8ms/step - loss: 0.0249 - accuracy: 0.9917 - val\_loss: 0.0454 - val\_accuracy: 0.9865

```
In [17]: test_loss, test_accuracy = model.evaluate(x_test, y_test)
          print("Test Accuracy:", test_accuracy)
```

313/313 [=====] - 0s 1ms/step - loss: 0.0361 - accuracy: 0.9886

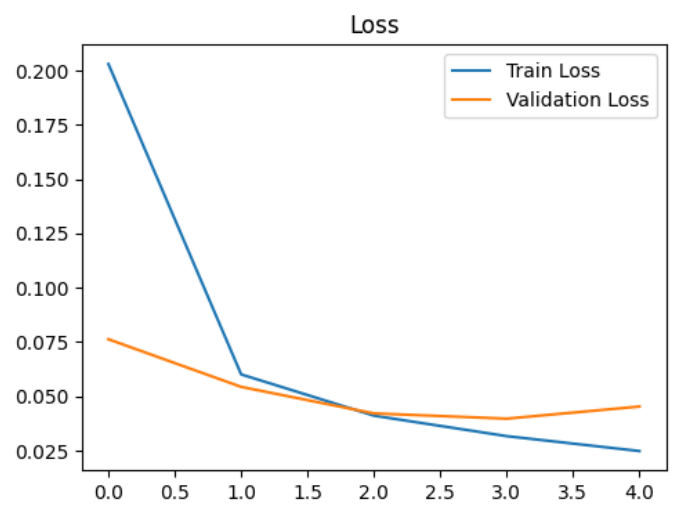
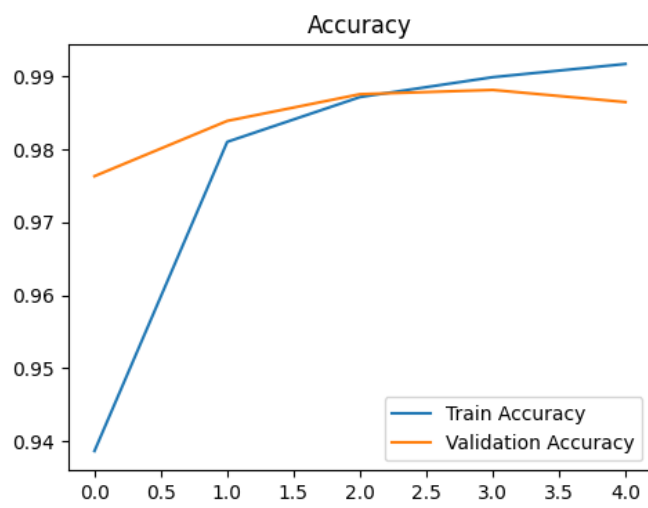
Test Accuracy: 0.9886000156402588

```
In [19]: plt.figure(figsize=(12,4))

          plt.subplot(1,2,1)
          plt.plot(history.history['accuracy'], label='Train Accuracy')
          plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
          plt.legend()
          plt.title("Accuracy")

          plt.subplot(1,2,2)
          plt.plot(history.history['loss'], label='Train Loss')
          plt.plot(history.history['val_loss'], label='Validation Loss')
          plt.legend()
          plt.title("Loss")

          plt.show()
```



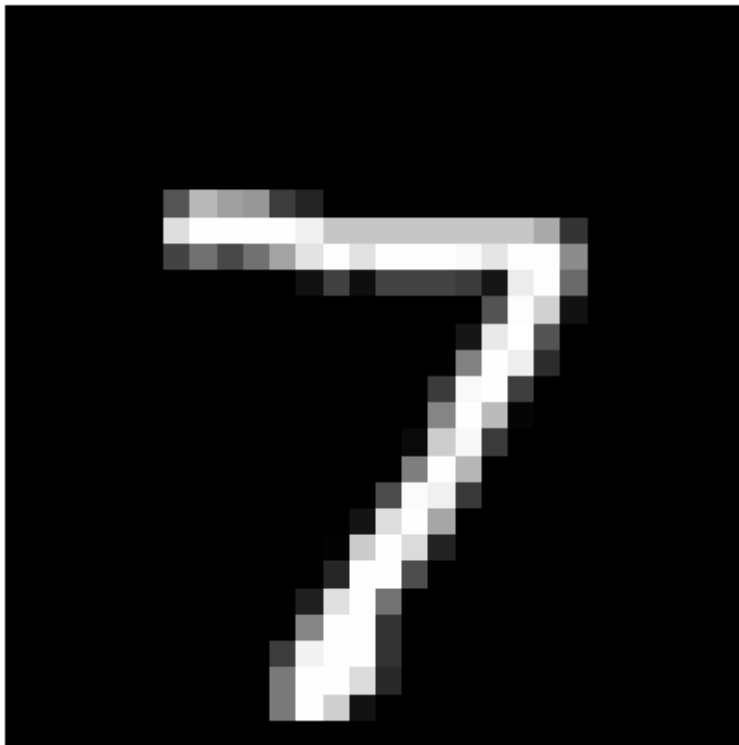
```
In [21]: predictions = model.predict(x_test)

plt.imshow(x_test[0].reshape(28,28), cmap='gray')
plt.title("Predicted Digit: " + str(predictions[0].argmax()))
plt.axis('off')
```

313/313 [=====] - 0s 1ms/step

Out[21]: (-0.5, 27.5, 27.5, -0.5)

Predicted Digit: 7



In [ ]: