**Name: Gaurang Vaghela**

**Rollno: TEAD22561**

**Practical - 13**

**Problem Statement:** MNIST Handwritten Character Detection using PyTorch, Keras,   
 and Tensorflow

**CODE:**

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

# Load the MNIST dataset, a collection of handwritten digits

mnist = keras.datasets.mnist

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Normalize pixel values to be in the range [0, 1]

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

# Build the neural network model

model = keras.models.Sequential([

layers.Flatten(input\_shape=(28, 28)),

layers.Dense(128, activation='relu'),

layers.Dropout(0.2),

layers.Dense(10, activation='softmax')

])

# Compile the model

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

model.fit(x\_train, y\_train, epochs=5)

# Evaluate the model on the test data

test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=2)

print("\nTest accuracy:", test\_acc)

# Save the model

model.save("handwritten\_digit\_recognition\_model.h5")

import tensorflow as tf

from tensorflow.keras.models import load\_model

import numpy as np

import matplotlib.pyplot as plt

# Load the saved model

model = load\_model("handwritten\_digit\_recognition\_model.h5")

# Load an example image (you can replace this with your own image)

image\_path = "D:/number.jpg"

image = tf.keras.preprocessing.image.load\_img(image\_path,

color\_mode="grayscale",

target\_size=(28, 28))

input\_data = tf.keras.preprocessing.image.img\_to\_array(image)

input\_data = input\_data / 255.0 # Normalize the pixel values

# Make a prediction on the input data

prediction = model.predict(np.array([input\_data]))

# Get the predicted digit (the class with the highest probability)

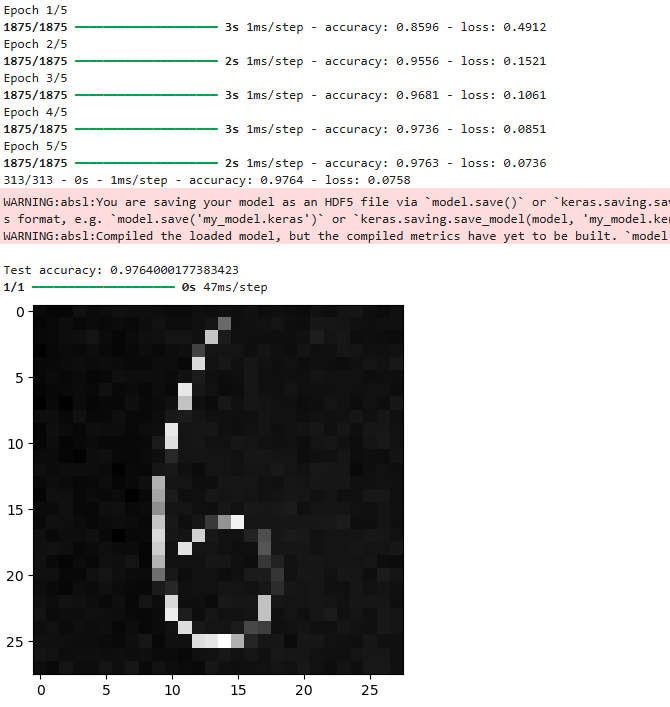
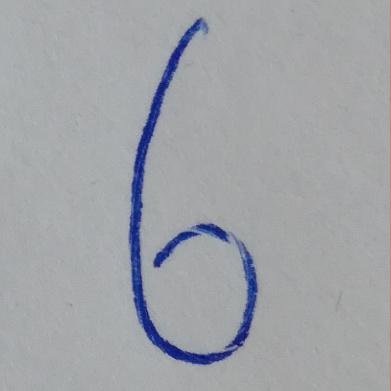
predicted\_digit = np.argmax(prediction)

# Display the image and the predicted digit

plt.imshow(input\_data.reshape(28, 28), cmap="binary")

plt.show()

**OUTPUT:**

****