import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

mnist=tf.keras.datasets.mnist

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

x\_train=tf.keras.utils.normalize(x\_train,axis=1)

x\_test=tf.keras.utils.normalize(x\_test,axis=1)

model=tf.keras.models.Sequential()

model.add(tf.keras.layers.Flatten(input\_shape=(28,28)))

model.add(tf.keras.layers.Dense(128,activation='relu'))

model.add(tf.keras.layers.Dense(128,activation='relu'))

model.add(tf.keras.layers.Dense(10,activation='softmax'))

model.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])

model.fit(x\_train,y\_train,epochs=3)

# Save the model

model.save('handwritten\_model.keras')

# Load the model

model = tf.keras.models.load\_model('handwritten\_model.keras')

# Evaluate the model

loss, accuracy = model.evaluate(x\_test, y\_test)

print(loss)

print(accuracy)

# Predict on custom images

image\_number = 1

while os.path.isfile(f"digits/digit{image\_number}.png"):

    img = cv2.imread(f"digits/digit{image\_number}.png", cv2.IMREAD\_GRAYSCALE)

    img = cv2.resize(img, (28, 28))  # Resize to 28x28

    img = np.invert(img)

    img = img.reshape(1, 28, 28)  # Reshape to match the input shape of the model

    img = tf.keras.utils.normalize(img, axis=1)  # Normalize the image

    prediction = model.predict(img)

    print(f"This digit is probably a {np.argmax(prediction)}")

    plt.imshow(img[0], cmap=plt.cm.binary)

    plt.show()

    image\_number += 1

**OUTPUT:**

C:\Users\supri\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\reshaping\flatten.py:37: 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\_\_(\*\*kwargs)

Epoch 1/3

**1875/1875** ━━━━━━━━━━━━━━━━━━━━ **7s** 3ms/step - accuracy: 0.8638 - loss: 0.4713

Epoch 2/3

**1875/1875** ━━━━━━━━━━━━━━━━━━━━ **5s** 3ms/step - accuracy: 0.9635 - loss: 0.1180

Epoch 3/3

**1875/1875** ━━━━━━━━━━━━━━━━━━━━ **5s** 3ms/step - accuracy: 0.9766 - loss: 0.0749

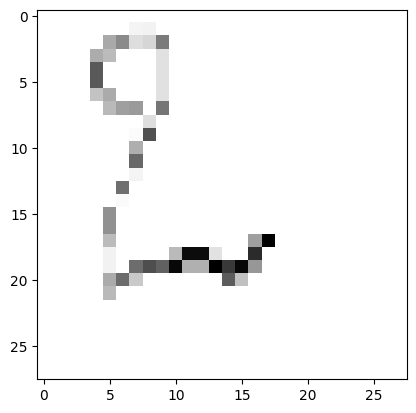
**313/313** ━━━━━━━━━━━━━━━━━━━━ **1s** 2ms/step - accuracy: 0.9634 - loss: 0.1151

0.10237119346857071

0.9678999781608582

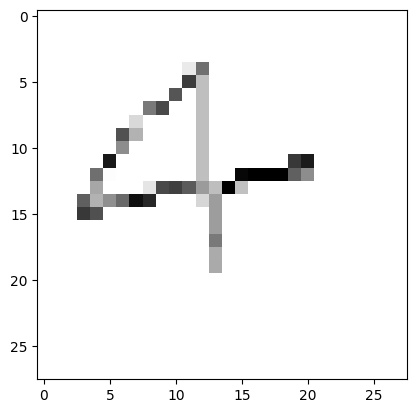
**1/1** ━━━━━━━━━━━━━━━━━━━━ **0s** 83ms/step

**This digit is probably a 2**



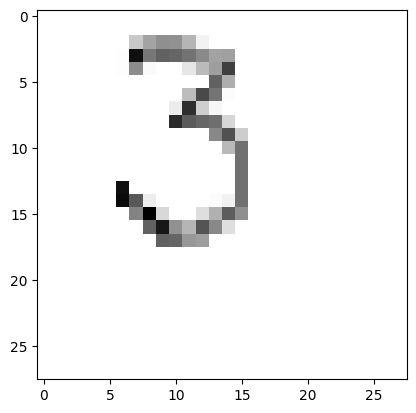
**1/1** ━━━━━━━━━━━━━━━━━━━━ **0s** 44ms/step

**This digit is probably a 4**

****

**1/1** ━━━━━━━━━━━━━━━━━━━━ **0s** 45ms/step

**This digit is probably a 3**

****

**1/1** ━━━━━━━━━━━━━━━━━━━━ **0s** 43ms/step

**This digit is probably a 5**

