Code

import matplotlib.pyplot as plt

import pandas as pd

import cv2

import numpy as np

import collections

import tensorflow as tf

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

# Load the digits dataset

digits = datasets.load\_digits()

# Get the input images from the dataset

data = digits.images

# Calculate the moments of the input images

moments = []

for img in data:

# Convert the image to grayscale

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Apply thresholding to get a binary image

ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)

# Calculate the moments

M = cv2.moments(thresh)

moments.append(M)

# Convert the moments to a pandas DataFrame

df\_data = pd.DataFrame(moments)

# Convert the DataFrame to float32

df\_data = df\_data.astype('float32')

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df\_data, digits.target, test\_size=0.2, shuffle=False)

# Define the neural network model

model = tf.keras.Sequential([

tf.keras.layers.Flatten(input\_shape=(df\_data.shape[1],)),

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

tf.keras.layers.Dense(10)

])

# Compile the model

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=['accuracy'])

# Train the model

model.fit(X\_train, y\_train, epochs=100)

# Evaluate the model on the testing data

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

# Print the test accuracy

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