import os

import numpy as np

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, classification\_report

from sklearn import preprocessing

from PIL import Image

def load\_data\_from\_folder(folder):

    data = []

    labels = []

    for class\_folder in os.listdir(folder):

        class\_path = os.path.join(folder, class\_folder)

        for filename in os.listdir(class\_path):

            img\_path = os.path.join(class\_path, filename)

            img = Image.open(img\_path).convert('L')  # Open image and convert to grayscale

            img = img.resize((64, 64))  # Resize the image to a fixed size

            img\_array = np.array(img)

            data.append(img\_array.flatten())  # Flatten the image into a 1D array

            labels.append(class\_folder)

    return np.array(data), np.array(labels)

# Load training and testing data

train\_data, train\_labels = load\_data\_from\_folder('/content/drive/MyDrive/mini project/Indian Currency Dataset/train')

test\_data, test\_labels = load\_data\_from\_folder('/content/drive/MyDrive/mini project/Indian Currency Dataset/test')

# Normalize the data

scaler = preprocessing.MinMaxScaler()

train\_data = scaler.fit\_transform(train\_data)

test\_data = scaler.transform(test\_data)

# Train the KNN model

knn = KNeighborsClassifier(n\_neighbors=12)

knn.fit(train\_data, train\_labels)

# Predict using the trained model

predictions = knn.predict(test\_data)

# Generate confusion matrix

conf\_matrix = confusion\_matrix(test\_labels, predictions)

# Generate classification report

class\_report = classification\_report(test\_labels, predictions)

# Generate confusion matrix

conf\_matrix = confusion\_matrix(test\_labels, predictions)

# Display confusion matrix with TP and TN counts

plt.figure(figsize=(8, 6))

plt.imshow(conf\_matrix, cmap='Blues', interpolation='nearest')

for i in range(conf\_matrix.shape[0]):

    for j in range(conf\_matrix.shape[1]):

        plt.text(j, i, str(conf\_matrix[i, j]), ha='center', va='center', color='black')

plt.colorbar()

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.xticks(np.arange(len(np.unique(train\_labels))), np.unique(train\_labels))

plt.yticks(np.arange(len(np.unique(test\_labels))), np.unique(test\_labels))

plt.show()

# Display classification report

print("Classification Report:")

print(class\_report)