from google.colab import drive

drive.mount('/content/drive')

import os

import numpy as np

from skimage import io, transform

from sklearn import svm

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make\_pipeline

from sklearn.feature\_extraction import image

# Path to your train, test, and validate folders

train\_folder = '/content/drive/MyDrive/waste project/data/train'

test\_folder = '/content/drive/MyDrive/waste project/data/test'

validate\_folder = '/content/drive/MyDrive/waste project/data/valid'

# Function to extract features from images

def extract\_features\_from\_folder(folder):

features = []

labels = []

for label in os.listdir(folder):

label\_folder = os.path.join(folder, label)

for filename in os.listdir(label\_folder):

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

img = io.imread(img\_path)

# You may need to resize or preprocess images

img = transform.resize(img, (100, 100)) # Resize image to a common size

feature = np.ravel(img) # Flatten image as feature vector

features.append(feature)

labels.append(label)

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

# Load train, test, and validate data

X\_train, y\_train = extract\_features\_from\_folder(train\_folder)

X\_test, y\_test = extract\_features\_from\_folder(test\_folder)

X\_validate, y\_validate = extract\_features\_from\_folder(validate\_folder)

# Create an SVM classifier

svm\_model = make\_pipeline(StandardScaler(), svm.SVC(kernel='linear', C=1.0))

# Train the SVM model

svm\_model.fit(X\_train, y\_train)

# Predict on test set

y\_pred = svm\_model.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy on test set: {accuracy}")

# Optionally, perform hyperparameter tuning using GridSearchCV

param\_grid = {'svc\_\_C': [0.1, 1, 10], 'svc\_\_kernel': ['linear', 'rbf']}

grid\_search = GridSearchCV(svm\_model, param\_grid, cv=5)

grid\_search.fit(X\_train, y\_train)

best\_params = grid\_search.best\_params\_

print("Best parameters:", best\_params)

# Validate the model

y\_validate\_pred = grid\_search.predict(X\_validate)

validate\_accuracy = accuracy\_score(y\_validate, y\_validate\_pred)

print(f"Accuracy of SVM Model: {validate\_accuracy\* 100:.2f}%")

# Generate confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_validate\_pred)

# Generate classification report

class\_report = classification\_report(y\_test, y\_validate\_pred)

# 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(y\_train))

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

plt.show()

# Display classification report

print("Classification Report:")

print(class\_report)