RECYCLABLE WASTE CLASSIFICATION USING DEEP LEARNING

A DESIGN PROJECT REPORT

Submitted by

AGNES MARY LAVANYA A

BRINDHA G

SAHANA SRI D

VINODHA R

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENGE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM–621112

NOVEMBER 2024

SAMPLE CODE:

```
# importing libraries

import numpy as np

#import pandas as pd

import matplotlib.pyplot as plt

import os

import random

import glob  # to find files

# Seaborn library for bar chart

#import seaborn as sns

# Libraries for TensorFlow

from tensorflow.keras.utils import to_categorical

from tensorflow.keras.preprocessing import image

from tensorflow.keras import models, layers

# Library for Transfer Learning

from tensorflow.keras.applications import VGG16

from keras.applications.vgg16 import preprocess_input

print("Importing libraries completed.")

path = 'Dataset/'

train_folder = path + "Data/"

test_folder = path + "Data/"

# variables for image size

img_width = 200

img_height = 200
```

```
# variable for model

batch_size = 32

epochs = 10

print("Variable declaration completed.")

# listing the folders containing images

# Train Dataset

train_class_names = os.listdir(train_folder)

print("Train class names: %s" % (train_class_names))

# print("\n")

# Test Dataset

test_class_names = os.listdir(test_folder)

print("Test class names: %s" % (test_class_names))

# print("\n")

print("\nDataset class name listing completed.")

# declaration of functions

# Declaring variables

x = []  # to store array value of the images

y = []  # to store the labels of the images

for folder in os.listdir(train_folder):

image_list = os.listdir(train_folder + "/" + folder)

    for img_name in image_list:

        # Loading images

img = image.load_img(train_folder + "/" + folder + "/" + img_name, target_size=(img_width,
img_height))
```

```
        # Converting to arrary
img = image.img_to_array(img)
# Transfer Learning: this is to apply preprocess of VGG16 model to our images before passing it to VGG16
img = preprocess_input(img)  # Optional step
        # Appending the arrarys
x.append(img)  # appending image array
y.append(train_class_names.index(folder))  # appending class index to the array
print("Preparing Training Dataset Completed.")
# Preparing validation images data (image array and class name) for processing
test_images = []
test_images_Original = []
test_image_label = []  # to store the labels of the images
for folder in os.listdir(test_folder):
image_list = os.listdir(test_folder + "/" + folder)
    for img_name in image_list:
model_vgg16 = VGG16(weights='imagenet')
model_vgg16.summary()
print("Summary of Custom VGG16 model.\n")
input_layer = layers.Input(shape=(img_width, img_height, 3))
model_vgg16 = VGG16(weights='imagenet', input_tensor=input_layer, include_top=False)
model_vgg16.summary()
last_layer = model_vgg16.output
flatten = layers.Flatten()(last_layer)
output_layer = layers.Dense(6, activation='softmax')(flatten)
```

```
model = models.Model(inputs=input_layer, outputs=output_layer)

model.summary()

for layer in model.layers[:-1]:

layer.trainable = False

model.summary()

from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=5)

print("Splitting data for train and test completed.")

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

print("Model compilation completed.")

history2 = model.fit(xtrain, ytrain, epochs=epochs, batch_size=batch_size, verbose=True,
validation_data=(xtest, ytest))

print("Fitting the model completed.")
```