import tensorflow as tf

from tensorflow.keras.preprocessing import image

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras import layers, models

import matplotlib.pyplot as plt

import numpy as np

# Path to your dataset (adjust accordingly)

train\_dir = r'C:\Users\kound\OneDrive\Desktop\Harshini minor1\train'

validation\_dir = r'C:\Users\kound\OneDrive\Desktop\Harshini minor1\validation'

# Image size for resizing

IMG\_SIZE = 150

# Image data generator to augment and preprocess images

train\_datagen = ImageDataGenerator(rescale=1./255, horizontal\_flip=True, rotation\_range=40, zoom\_range=0.2, shear\_range=0.2)

validation\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

    train\_dir,

    target\_size=(IMG\_SIZE, IMG\_SIZE),

    batch\_size=32,

    class\_mode='binary',  # Binary classification (biodegradable or not)

)

validation\_generator = validation\_datagen.flow\_from\_directory(

    validation\_dir,

    target\_size=(IMG\_SIZE, IMG\_SIZE),

    batch\_size=32,

    class\_mode='binary',

)

# Load a pre-trained model (MobileNetV2 for example) with ImageNet weights, excluding the top layers

base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(IMG\_SIZE, IMG\_SIZE, 3))

# Freeze the layers of the base model to prevent them from being trained again

base\_model.trainable = False

# Add custom layers on top of the pre-trained model

model = models.Sequential([

    base\_model,

    layers.GlobalAveragePooling2D(),

    layers.Dense(512, activation='relu'),

    layers.Dense(1, activation='sigmoid')  # Output 1 unit for binary classification

])

# Compile the model

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

# Train the model

model.fit(

    train\_generator,

    steps\_per\_epoch=train\_generator.samples // train\_generator.batch\_size,

    epochs=10,

    validation\_data=validation\_generator,

    validation\_steps=validation\_generator.samples // validation\_generator.batch\_size,

)

# Save the trained model

model.save('biodegradable\_classifier\_mobilenetv2.h5')

# Function to classify and display the image

def classify\_and\_display\_image(img\_path):

    # Load and preprocess the image

    img = image.load\_img(img\_path, target\_size=(IMG\_SIZE, IMG\_SIZE))

    img\_array = image.img\_to\_array(img)

    img\_array = np.expand\_dims(img\_array, axis=0)

    img\_array /= 255.0  # Normalize the image

    # Predict the class using the trained model

    prediction = model.predict(img\_array)

    # Display the image

    img = plt.imread(img\_path)

    plt.imshow(img)

    plt.axis('off')

    # Show the prediction (flip the label)

    label = 'Non-Biodegradable' if prediction[0] > 0.5 else 'Biodegradable'

    plt.title(f'Prediction: {label}')

    plt.show()

# Example usage

img\_path = r'C:\Users\kound\OneDrive\Desktop\Harshini minor1\train\non biodegradable\image 5.jpeg'  # Path to the image you want to classify

classify\_and\_display\_image(img\_path)