Predict code   
  
  
import cv2

import numpy as np

import tensorflow as tf

import os

# Define the labels for classification

class\_labels = ["Organic", "Recyclable"]

# Define the image size for resizing

IMG\_SIZE = (224, 224)

# Load the trained model from the directory

model\_path = os.path.join("model", "waste\_classifier\_model.h5")

model = tf.keras.models.load\_model(model\_path)

# Load the Haar cascade for object detection

cascade\_path = os.path.join("model", "haarcascade\_frontalface\_default.xml")

cascade = cv2.CascadeClassifier(cascade\_path)

# Open the camera

cap = cv2.VideoCapture(0, cv2.CAP\_DSHOW)

# Set the camera window size

cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 1280)

cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 720)

while True:

    # Capture frame-by-frame

    ret, frame = cap.read()

    # Convert frame to grayscale for object detection

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

    # Perform object detection using Haar cascades

    faces = cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:

        # Crop the detected object from the frame

        object\_img = frame[y:y+h, x:x+w]

        # Preprocess the object image

        object\_resized = cv2.resize(object\_img, IMG\_SIZE)

        object\_normalized = object\_resized / 255.0

        object\_expanded = np.expand\_dims(object\_normalized, axis=0)

        # Make prediction

        predictions = model.predict(object\_expanded)

        predicted\_class = np.argmax(predictions)

        prediction\_percentage = np.max(predictions) \* 100

        # Get the predicted class label and percentage

        class\_label = class\_labels[predicted\_class]

        prediction\_text = f"{class\_label}: {prediction\_percentage:.2f}%"

        # Display the predicted class label and percentage on the frame

        cv2.putText(frame, prediction\_text, (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

        # Draw a rectangle around the detected object

        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # Display the frame

    cv2.imshow("Camera", frame)

    # Break the loop if 'q' is pressed

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Release the camera and close all windows

cap.release()

cv2.destroyAllWindows()