Kode ini bertujuan untuk men-deploy sebuah model TensorFlow Lite ke dalam perangkat Raspberry Pi untuk melakukan inferensi/klasifikasi secara real-time pada video stream dari kamera.

Pertama, kode mengimpor pustaka Python yang dibutuhkan seperti Interpreter dari TensorFlow Lite, GPIO dan PWM untuk kontrol servo, numpy untuk operasi array, OpenCV untuk manajemen gambar dan video, serta pustaka waktu.

from tensorflow.lite.python.interpreter import Interpreter

import RPi.GPIO as GPIO

import numpy as np

import time

import cv2

Kemudian didefinisikan kelas Servo yang bertugas mengontrol pergerakan servo menggunakan pin GPIO dan sinyal PWM. Metode rotate\_180() digunakan untuk memutar servo ke sudut 180 derajat.

class Servo:

    def \_\_init\_\_(*self*, *servo\_pin*:int, *pwm\_period*:int=50):

*self*.servo\_pin = *servo\_pin*

*self*.pwm\_period= *pwm\_period*

        GPIO.setmode(GPIO.BCM)

        GPIO.setup(*self*.servo\_pin, GPIO.OUT)

        GPIO.setwarnings(False)

*self*.pwm = GPIO.PWM(*self*.servo\_pin, *self*.pwm\_period)

*self*.pwm.start(7)

    def rotate\_180(*self*):

        "rotate servo to 180 degrees"

*self*.pwm.ChangeDutyCycle(2.0)

        time.sleep(0.5)

*self*.pwm.ChangeDutyCycle(12.0)

        time.sleep(0.5)

        GPIO.cleanup()

Selanjutnya didefinisikan kelas utama ObjectDetection yang menerima 3 parameter: path ke model TensorFlow Lite, path ke webcam/video, dan path ke labelmap. Labelmap berisi daftar kelas yang dideteksi model.

class ObjectDetection():

    def \_\_init\_\_(*self*, *model\_path*:str, *webcam\_path*:str, *label\_path*:str):

*self*.model\_path = *model\_path*

*self*.webcam\_path = *webcam\_path*

*self*.label\_path = *label\_path*

Pada method start(), pertama model TFLite dimuat ke memori menggunakan Interpreter. Kemudian detail input dan output model diambil untuk mendapatkan dimensi input yang diharapkan model. Video input dibuka menggunakan OpenCV.

    def start(*self*, *min\_conf*=0.5):

        # Load the label map into memory

        with open(*self*.label\_path, 'r') as f:

            labels = [line.strip() for line in f.readlines()]

        # Load the Tensorflow Lite model into memory

        interpreter = Interpreter(*model\_path*=*self*.model\_path)

        interpreter.allocate\_tensors()

        # Get model details

        input\_details = interpreter.get\_input\_details()[0]

        output\_details = interpreter.get\_output\_details()

        height = input\_details['shape'][1]

        width = input\_details['shape'][2]

        float\_input = (input\_details['dtype'] == np.float32)

        input\_mean = 127.5

        input\_std = 127.5

        # Open video capture

        cap = cv2.VideoCapture(*self*.webcam\_path)

Dalam loop, setiap frame video di-preprocess menjadi input model dengan resize ke dimensi yang diharapkan dan normalisasi nilai pixel jika model bersifat floating point. Input diberikan ke model untuk inferensi, lalu hasil deteksi diambil.

while True:

    # Read frame from video capture

    ret, frame = cap.read()

    if not ret:

        break

    # Resize frame to expected shape [1xHxWx3]

    image\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    imH, imW, \_ = frame.shape

    image\_resized = cv2.resize(image\_rgb, (width, height))

    input\_data = np.expand\_dims(image\_resized, *axis*=0)

    # Normalize pixel values if using a floating model (i.e. if model is non-quantized)

    if float\_input:

        input\_data = (np.float32(input\_data) - input\_mean) / input\_std

    # Perform the actual detection by running the model with the image as input

    interpreter.set\_tensor(input\_details['index'], input\_data)

    interpreter.invoke()

    # Retrieve detection results

    boxes = interpreter.get\_tensor(output\_details[1]['index'])[0]

    classes = interpreter.get\_tensor(output\_details[3]['index'])[0]

    scores = interpreter.get\_tensor(output\_details[0]['index'])[0]

    detections = []

Hasil deteksi berupa kotak pembatas, kelas, dan skor kepercayaan. Kotak pembatas digambar jika skor melebihi threshold minimum. Nama kelas dan persentase skor ditampilkan.

for i in range(len(scores)):

    if ((scores[i] > min\_conf) and (scores[i] <= 1.0)):

        # Get bounding box coordinates and draw box

        ymin = int(max(1,(boxes[i][0] \* imH)))

        xmin = int(max(1,(boxes[i][1] \* imW)))

        ymax = int(min(imH,(boxes[i][2] \* imH)))

        xmax = int(min(imW,(boxes[i][3] \* imW)))

        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

        # Draw label

        object\_name = labels[int(classes[i])]

        label = '%s: %d%%' % (object\_name, int(scores[i]\*100))

        labelSize, baseLine = cv2.getTextSize(label, cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, 2)

        label\_ymin = max(ymin, labelSize[1] + 10)

        cv2.rectangle(frame, (xmin, label\_ymin-labelSize[1]-10), (xmin+labelSize[0], label\_ymin+baseLine-10), (255, 255, 255), cv2.FILLED)

        cv2.putText(frame, label, (xmin, label\_ymin-7), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 0), 2)

        detections.append(object\_name.upper())

Jika kelas tertentu terdeteksi, misal kelas organik, servo dirotasikan. Akhirnya frame hasil deteksi ditampilkan.