import argparse

import os

import sys

from pathlib import Path

import pandas as pd

import torch

from tqdm import tqdm

import cv2

from pathlib import Path

import pathlib

relay1 = 17

relay2 = 4

# import RPi.GPIO as GPIO

import time

# GPIO.setmode(GPIO.BCM)

# GPIO.setup(relay1, GPIO.OUT)

# GPIO.setup(relay2, GPIO.OUT)

temp = pathlib.PosixPath

pathlib.PosixPath = pathlib.WindowsPath

from models.common import DetectMultiBackend

from utils.dataloaders import IMG\_FORMATS, VID\_FORMATS, LoadImages, LoadScreenshots, LoadStreams

from utils.general import (LOGGER, Profile, check\_file, check\_img\_size, check\_imshow, check\_requirements, colorstr, cv2,

increment\_path, non\_max\_suppression, print\_args, scale\_boxes, scale\_segments,

strip\_optimizer, xyxy2xywh)

from utils.plots import Annotator, colors, save\_one\_box

from utils.segment.general import masks2segments, process\_mask

from utils.torch\_utils import select\_device, smart\_inference\_mode

from utils.augmentations import classify\_transforms

classdct = {

"0": 'Bacterial Spot', "1": 'Black Measles', "2": 'Black Rot', "3": 'Common Rust', "4": 'Early Blight',

"5": 'Gray Leaf Spot', "6": 'Isariopsis Leaf Spot', "7": 'Late Blight', "8": 'Leaf Mold',

"9": 'Leaf Scorch', "10": 'Mosaic Virus', "11": 'Northern Leaf Blight', "12": 'Powdery Mildew',

"13": 'Septoria Leaf Spot', "14": 'Spider Mites', "15": 'Target Spot',

"16": 'Yellow Leaf Curl Virus', "17": 'Fussarium\_Wilt', "18": 'Healthy'

}

pesticide\_dict = {

'Bacterial Spot': 'Copper-based fungicides',

'Black Measles': 'Copper sulfate',

'Black Rot': 'Fungicides containing copper or sulfur',

'Common Rust': 'Systemic fungicides (e.g., propiconazole)',

'Early Blight': 'Fungicides with chlorothalonil or mancozeb',

'Gray Leaf Spot': 'Fungicides containing azoxystrobin or chlorothalonil',

'Isariopsis Leaf Spot': 'Fungicides containing chlorothalonil or mancozeb',

'Late Blight': 'Fungicides containing metalaxyl or copper-based products',

'Leaf Mold': 'Fungicides with copper or systemic fungicides like azoxystrobin',

'Leaf Scorch': 'Use foliar fertilizers and manage water stress',

'Mosaic Virus': 'No specific pesticide; focus on controlling insect vectors',

'Northern Leaf Blight': 'Fungicides with propiconazole or tebuconazole',

'Powdery Mildew': 'Fungicides with sulfur or potassium bicarbonate',

'Septoria Leaf Spot': 'Fungicides with chlorothalonil or mancozeb',

'Spider Mites': 'Acaricides like abamectin or miticides',

'Target Spot': 'Fungicides with chlorothalonil or azoxystrobin',

'Yellow Leaf Curl Virus': 'No specific pesticide; control insect vectors',

'Fussarium\_Wilt': 'Soil treatments with fungicides containing thiabendazole or other systemic fungicides',

'Healthy': 'No pesticide needed'

}

device = torch.device('cpu')

model\_weights = "../model\_file/best.pt"

imgsz = (224,)

workers = 8

verbose = False

half = False

dnn = False

device = select\_device(device)

model = DetectMultiBackend(model\_weights, device=device, dnn=dnn, fp16=half)

stride, pt, jit, engine = model.stride, model.pt, model.jit, model.engine

imgsz = check\_img\_size(imgsz, s=stride) # check image size

half = model.fp16

vid\_stride = 1

cap = cv2.VideoCapture(0)

ret = cap.set(3, 512)

ret = cap.set(4, 256)

cap.set(cv2.CAP\_PROP\_FPS, 5)

fps = cap.get(cv2.CAP\_PROP\_FPS)

while True:

ret, frame = cap.read()

filename = "./realtime\_feed.jpg"

cv2.imwrite(filename, frame)

dataset = LoadImages(filename, img\_size=imgsz, transforms=classify\_transforms(imgsz[0]), vid\_stride=vid\_stride)

for path, im, im0s, vid\_cap, s in dataset:

im = torch.Tensor(im).to(model.device)

im = im.half() if model.fp16 else im.float()

if len(im.shape) == 3:

im = im[None]

results = model(im)

final\_pred = results.argsort(1, descending=True)[:, :5].tolist()[0][0]

final\_pred\_class = classdct[f"{final\_pred}"]

text = f"{final\_pred\_class} :{pesticide\_dict[final\_pred\_class]}"

print("#" \* 75)

print(text)

print("#" \* 75)

'''if text:

GPIO.output(relay1, True)

GPIO.output(relay2, False)

time.sleep(2)

GPIO.output(relay1, False)

GPIO.output(relay2, True)

time.sleep(2)

GPIO.output(relay1, False)

GPIO.output(relay2, False)'''

position = (10, 30)

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_scale = 0.5

color = (0, 0, 255)

thickness = 1

cv2.putText(frame, text, position, font, font\_scale, color, thickness, cv2.LINE\_AA)

cv2.imshow('Video Feed', frame)

# Wait for 1ms for a key press and stop if 'q' is pressed

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

cap.release()

cv2.destroyAllWindows()

break

# Break the outer loop if 'q' is pressed

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

break

cap.release()

cv2.destroyAllWindows()