**SOURCE CODE**

import argparse

import io

import os

from PIL import Image

import cv2

import numpy as np

from torchvision.models import detection

import torch

from torchvision import models

from flask import Flask, render\_template, request, redirect, Response

app = Flask(\_name\_)

model = torch.hub.load(

"ultralytics/yolov5", "custom", path="model/last.pt", force\_reload=True

)

model.eval()

model.conf = 0.5

model.iou = 0.45

from io import BytesIO

def gen():24

"""

The function takes in a video stream from the webcam, runs it through the model, and

returns the

output of the model as a video stream

"""

cap = cv2.VideoCapture(0)

while cap.isOpened():

success, frame = cap.read()

if success == True:

ret, buffer = cv2.imencode(".jpg", frame)

frame = buffer.tobytes()

img = Image.open(io.BytesIO(frame))

results = model(img, size=640)

results.print()

img = np.squeeze(results.render())

img\_BGR = cv2.cvtColor(img, cv2.COLOR\_RGB2BGR)

else:

break

frame = cv2.imencode(".jpg", img\_BGR)[1].tobytes()

yield (b"--frame\r\n" b"Content-Type: image/jpeg\r\n\r\n" + frame + b"\r\n")

@app.route("/video")

def video():

"""

It returns a response object that contains a generator function that yields a sequence of

images

:return: A response object with the gen() function as the body.

"""

return Response(gen(), mimetype="multipart/x-mixed-replace; boundary=frame")25

@app.route("/", methods=["GET", "POST"])

def predict():

"""

The function takes in an image, runs it through the model, and then saves the output

image to a

static folder

:return: The image is being returned.

"""

if request.method == "POST":

if "file" not in request.files:

return redirect(request.url)

file = request.files["file"]

if not file:

return

img\_bytes = file.read()

img = Image.open(io.BytesIO(img\_bytes))

results = model(img, size=640)

results.render()

for img in results.imgs:

img\_base64 = Image.fromarray(img)

img\_base64.save("static/image0.jpg", format="JPEG")

return redirect("static/image0.jpg")

return render\_template("index.html")

if \_name\_ == "\_main\_":

app.run(host="0.0.0.0", port=5000)

from ai.ai\_model import load\_yolov5\_model

from ai.ai\_model import detection26

from helper.params import Parameters

from helper.general\_utils import filter\_text

from helper.general\_utils import save\_results

from ai.ocr\_model import easyocr\_model\_load

from ai.ocr\_model import easyocr\_model\_works

from utils.visual\_utils import \*

import cv2

from datetime import datetime

# Loading the parameters from the params.py file.

params = Parameters()

if \_name\_ == "\_main\_":

# Loading the model and labels from the ai\_model.py file.

model, labels = load\_yolov5\_model()

# Capturing the video from the webcam.

camera = cv2.VideoCapture(0)

# Loading the model for the OCR.

text\_reader = easyocr\_model\_load()

while 1:

# Reading the video from the webcam.

ret, frame = camera.read()

if ret:27

# Detecting the text from the image.

detected, \_ = detection(frame, model, labels)

# Reading the text from the image.

resulteasyocr = text\_reader.readtext(

detected

) # text\_read.recognize() , you can use cropped plate image or whole image

# Filtering the text from the image.

text = filter\_text(params.rect\_size, resulteasyocr, params.region\_threshold)

# Saving the results of the OCR in a csv file.

save\_results(text[-1], "ocr\_results.csv", "Detection\_Images")

print(text)

cv2.imshow("detected", detected)

if cv2.waitKey(1) & 0xFF == 27:

cv2.destroyAllWindows()

break28

import cv2

from easyocr import Reader

from yolov5 import YOLOv5

# Initialize YOLOv8

yolo = YOLOv8(weights='path/to/weights.pt', device='cuda')

# Initialize EasyOCR

reader = Reader(['en'])

image = cv2.imread('input\_image.jpg')

results = yolo.detect(image)

for box in results['boxes']:

x1, y1, x2, y2, conf, cls = box

plate\_region = image[y1:y2, x1:x2]

plate\_text = reader.readtext(plate\_region)

print("Number Plate Text:", plate\_text)

cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.putText(image, plate\_text, (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

cv2.imshow('Result', image)

cv2.waitKey(0)

cv2.destroyAllWindows()