```
import random

import cv2
import numpy as np
from ultralytics import YOLO

# Load a COCO-pretrained YOLOv8n model
#model = YOLO('yolov8n.pt')

# Display model information (optional)
#model.info()

# Train the model on the COCO8 example dataset for 100 epochs
#results = model.train(data='coco8.yaml', epochs=100, imgsz=640)

# opening the file in read mode
my_file = open(r'C:\Users\Hanshu\Desktop\VS code\DL\OPENCV\YOLO\coco.txt', "r")
# reading the file
data = my_file.read()
# replacing end splitting the text | when newline ('\n') is seen.
class_list = data.split("\n")
my_file.close()

# print(class_list)

# Generate random colors for class list
detection_colors = []
for i in range(len(class_list)):
    r = random.randint(0, 255)
    g = random.randint(0, 255)
    b = random.randint(0, 255)
```

```
    detection_colors.append((b, g, r))

# load a pretrained YOLOv8n model
model = YOLO("weights/yolov8n.pt", "v8")

# Vals to resize video frames | small frame optimise the run
frame_wid = 640
frame_hyt = 480

# cap = cv2.VideoCapture(1)
cap = cv2.VideoCapture(r"C:\Users\Hanshu\Downloads\background video _ people _ walking _.mp4")

if not cap.isOpened():
    print("Cannot open camera")
    exit()

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    # if frame is read correctly ret is True

    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    #  resize the frame | small frame optimise the run
    # frame = cv2.resize(frame, (frame_wid, frame_hyt))

    # Predict on image
    detect_params = model.predict(source=[frame], conf=0.45, save=True)
```

```
# Convert tensor array to numpy
DP = detect_params[0].numpy()
print(DP)

if len(DP) != 0:
    for i in range(len(detect_params[0])):
        print(i)

        boxes = detect_params[0].boxes
        box = boxes[i]  # returns one box
        clsID = box.cls.numpy()[0]
        conf = box.conf.numpy()[0]
        bb = box.xyxy.numpy()[0]

        cv2.rectangle(
            frame,
            (int(bb[0]), int(bb[1])),
            (int(bb[2]), int(bb[3])),
            detection_colors[int(clsID)],
            3,
        )

        # Display class name and confidence
        font = cv2.FONT_HERSHEY_COMPLEX
        cv2.putText(
            frame,
            class_list[int(clsID)] + " " + str(round(conf, 3)) + "%",
            (int(bb[0]), int(bb[1]) - 10),
            font,
            1,
```

```
            (255, 255, 255),
            2,
        )

    # Display the resulting frame
    cv2.imshow("ObjectDetection", frame)

    # Terminate run when "Q" pressed
    if cv2.waitKey(1) == ord("q"):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```