**Assignment 7**

**1. Introduction**

Object detection is one of the most important tasks in computer vision, where the goal is not only to classify objects but also to **localize them within an image** using bounding boxes. In this assignment, we implement **YOLOv8 (You Only Look Once – Version 8)** for real-time and image-based object detection.

YOLO is a **single-stage detector** known for its **speed and accuracy**, making it suitable for real-time applications such as surveillance, autonomous driving, and crowd analysis.

**2. Dataset and Model**

* **Pre-trained Model Used**: yolov8s.pt (YOLOv8 Small, trained on COCO dataset with 80 classes).
* **Classes**: Includes 80 object categories (e.g., person, car, bicycle, dog, etc.).
* **Data Sources**:
  + **Webcam Feed**: Live object detection.
  + **Static Image**: A crowded-place image for offline detection.

**3. Methodology**

**3.1 Real-Time Object Detection**

1. OpenCV VideoCapture(0) was used to access the webcam.
2. Each frame was passed to the YOLO model for inference using yolo.track(frame, stream=True).
3. Objects with **confidence > 40%** were considered valid detections.
4. Bounding boxes were drawn using OpenCV with class names and confidence scores.
5. Custom color generation (getColours) was implemented to assign different colors to each detected class.
6. The annotated frames were displayed live until the user pressed **'q'**.

**3.2 Static Image Detection**

1. An image file (crowded-place.jpeg) was loaded using OpenCV.
2. YOLOv8 was applied directly using results = yolo(frame).
3. Detected objects were annotated with bounding boxes, class labels, and confidence values.
4. The processed image was displayed with detections highlighted.

**4. Model Architecture (YOLOv8 – Summary)**

* **Backbone**: CSPDarknet-based feature extractor.
* **Neck**: Path Aggregation Network (PANet) for feature fusion.
* **Head**: Predicts bounding boxes, confidence scores, and class probabilities in a single step.
* **Detection Approach**: End-to-end, grid-based detection → very fast compared to two-stage methods (like R-CNN).

**5. Implementation Code (Key Snippets)**

**Real-Time Detection (Webcam)**

videoCap = cv2.VideoCapture(0)

while True:

ret, frame = videoCap.read()

results = yolo.track(frame, stream=True)

# Draw bounding boxes + labels on frame

cv2.imshow('frame', frame)

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

break

**Static Image Detection**

frame = cv2.imread("crowded-place.jpeg")

results = yolo(frame)

for result in results:

for box in result.boxes:

if box.conf[0] > 0.4:

cv2.rectangle(frame, (x1, y1), (x2, y2), colour, 2)

cv2.putText(frame, label, (x1, y1-10),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, colour, 2)

**6. Results**

* **Real-Time Detection**: Successfully detected multiple objects (e.g., people, vehicles) in live video with bounding boxes and confidence labels.
* **Static Image Detection**: Correctly identified objects in the crowded image, labeling them with high accuracy.
* **Performance**: The YOLOv8-small model provides a good balance between **speed** (real-time performance on CPU/GPU) and **accuracy**.

**7. Conclusion**

This assignment demonstrated the implementation of **YOLOv8 for object detection** in both real-time (webcam) and static image scenarios. The model successfully identified multiple object categories with bounding boxes and confidence scores.