**Name:- Jagdev Hake**

**Roll No:- 381026**

**PRN:- 22310711**

**Assignment 7:**

**YOLO Object Detection**

**1. Problem Statement**

Implement object detection using a pretrained YOLO model to detect objects in images or video streams.

**2. Objective**

* Understand YOLO architecture.
* Load and use pretrained YOLO model.
* Perform real-time object detection.
* Evaluate detection performance.

**3. Software & Hardware Requirements**

* **Software**:  
   Python 3.10+, Jupyter Notebook / Google Colab, Anaconda, YOLOv8 pretrained weights.
* **Hardware**:  
   GPU (NVIDIA CUDA), 8 GB RAM, Webcam.

**4. Libraries Used**

* ultralytics (YOLO implementation)
* OpenCV, PIL, NumPy, Matplotlib, torch, torchvision.

**5. Theory**

* YOLO (You Only Look Once): Detects multiple objects in one forward pass by dividing the image into grids and predicting bounding boxes and class probabilities.
* Pretrained Models: Trained on datasets like COCO for detecting various object classes.
* Real-Time Detection: High speed and good accuracy for video and image streams.

**6. Methodology**

1. **Data Preparation**:  
    Resize images to 640x640; normalize pixel values.
2. **Load Model**:  
    Load YOLOv8 pretrained model.
3. **Inference**:  
    Model predicts bounding boxes, object confidence scores, and class probabilities.
4. **Post-Processing**:  
    Apply Non-Maximum Suppression (NMS) to remove redundant boxes.  
    Filter by confidence threshold (e.g., 0.5).
5. **Visualization**:  
    Draw bounding boxes and labels using OpenCV.  
    Display annotated images or video.
6. **Evaluation**:  
    Calculate accuracy, precision, recall, F1-score.

**7. Advantages**

* Real-time detection.
* High accuracy with a single forward pass.
* Uses pretrained models (e.g., on COCO).

**8. Limitations**

* May miss small or overlapping objects.
* Trade-off between speed and precision.
* Complex object detection can be less reliable.

**9. Applications**

* Autonomous vehicles
* Surveillance systems
* Medical imaging
* Retail product detection
* Gaming and AR/VR

**Working / Algorithm**

1. Load YOLOv8 model and set input size (640x640).
2. Preprocess input image → normalize and resize.
3. Pass image through YOLO → predict bounding boxes and classes.
4. Apply NMS → remove overlapping detections.
5. Filter by confidence threshold (e.g., 0.5).
6. Draw bounding boxes and class labels using OpenCV.
7. Display final annotated result.

**Conclusion**

YOLO provides efficient real-time object detection using pretrained models. Despite challenges with small or overlapping objects, it is widely used in applications like autonomous driving, surveillance, and healthcare, offering a good balance of speed and accuracy.