**Assignment 7 – Object Detection using YOLO and Pretrained Model**

**Name: Shubham Gautam Dahane**

**Roll No: 382015**

**PRN: 22310312**

**Problem Statement**

Implement an object detection system using the **YOLOv8 pretrained model**. The system should detect multiple objects in both static images and real-time webcam input, and display bounding boxes with labels.

**Objectives**

* To understand the working of **YOLO (You Only Look Once)** object detection models.
* To apply a **pretrained YOLOv8 model** for real-time and image-based object detection.
* To visualize bounding boxes and labels on detected objects.
* To implement real-time detection using a webcam feed.

**Requirements**

* **Operating System:** Windows/Linux/MacOS
* **Python Version:** 3.x
* **Tools:** Jupyter Notebook / Anaconda / Google Colab
* **Hardware:** CPU (GPU recommended for faster inference)
* **Libraries Used:**
  + ultralytics (YOLOv8)
  + OpenCV (cv2)
  + Matplotlib

**Theory**

**YOLO (You Only Look Once)** is a state-of-the-art deep learning algorithm for real-time object detection. Unlike traditional methods, YOLO divides an image into grids and predicts bounding boxes and class probabilities in a **single forward pass**, making it extremely fast.

* **YOLOv8** is the latest version, supporting pretrained weights for a wide variety of objects.
* **Pretrained Model (yolov8s.pt):** Already trained on the COCO dataset, capable of detecting 80 object categories.
* **OpenCV:** Used for capturing video streams and displaying annotated frames.

**Methodology**

1. **Setup**
   * Install required libraries: ultralytics, opencv-python, matplotlib.
   * Import YOLO model and load pretrained weights (yolov8s.pt).
2. **Image Detection**
   * Load a test image.
   * Pass image to the YOLO model.
   * Generate detection results (bounding boxes, labels, confidence).
   * Display annotated image using Matplotlib.
3. **Real-Time Detection**
   * Capture video stream using OpenCV (cv2.VideoCapture).
   * Pass each frame to YOLO model for detection.
   * Annotate frame with bounding boxes and labels.
   * Display annotated frames in real-time.
   * Exit loop when user presses q.

**Graphs and Visualizations**

1. **Annotated Test Image**
   * Bounding boxes and class labels drawn on static test images.
2. **Real-Time Detection Output**
   * Live camera feed showing detected objects with YOLO annotations.

**Advantages**

* **Real-time performance** due to single-shot detection.
* Can detect **multiple objects simultaneously**.
* Works with **pretrained models**, saving training time.
* Easily extendable to **custom datasets** for specific object detection tasks.

**Limitations**

* Performance depends on **lighting, camera quality, and environment**.
* Pretrained model limited to **80 COCO classes** unless retrained.
* Requires **GPU acceleration** for smooth real-time detection with larger models.
* Bounding boxes may overlap or miss small/occluded objects.

**Applications**

* **Surveillance Systems** – detecting people, vehicles, or suspicious objects.
* **Autonomous Vehicles** – real-time detection of pedestrians, traffic signs, and vehicles.
* **Retail Analytics** – customer behavior and product recognition.
* **Robotics** – navigation and object manipulation.
* **Smart Cities** – monitoring traffic flow and safety.

**Working / Algorithm**

**Step 1:** Import required libraries (ultralytics, cv2, matplotlib).  
**Step 2:** Load pretrained YOLOv8 model (yolov8s.pt).  
**Step 3:** Perform detection on static images and display annotated results.  
**Step 4:** Start webcam capture using OpenCV.  
**Step 5:** Pass each frame through YOLO model for object detection.  
**Step 6:** Display annotated frames with bounding boxes and labels.  
**Step 7:** Exit loop when user presses q.

**Conclusion**

The YOLOv8 pretrained model was successfully applied to perform **object detection on both images and real-time video**. The system demonstrated fast and accurate predictions across multiple object categories.

By integrating **YOLO with OpenCV**, the project highlights the effectiveness of pretrained deep learning models for real-world object detection tasks in fields such as security, autonomous systems, and robotics.