**Assignment No. 7**

**YOLO Object Detection**

**1. Problem Statement:**

Object detection using YOLO and Pretrained Model.

**2. Objective:**

The objective of this assignment is to implement object detection using the YOLO model. We will use a pretrained model to detect objects in images or video streams. This involves:

* Understanding the architecture of YOLO.
* Loading and fine-tuning a pretrained YOLO model.
* Evaluating its performance on different images or video streams for real-time object detection.

**3. Software and Hardware Packages Used:**

* **Software Packages:**
* Python 3.10 or later
* Jupyter Notebook or Google Colab
* Anaconda for environment management
* YOLOv8 pretrained model weights
* **Hardware Packages:**
* GPU-enabled machine for faster training and inference (e.g., NVIDIA CUDA GPU)
* At least 8 GB RAM for processing
* Web camera or external camera (for real-time object detection)

**4. Libraries Used:**

* ultralytics: For implementing YOLO models.
* NumPy: Array processing for numerical operations.
* OpenCV: Image and video processing.
* torch and torchvision: For deep learning model handling.
* Matplotlib: Visualization of detected objects.
* PIL (Python Imaging Library): For handling image data.

**5. Theory:**

* YOLO (You Only Look Once) is a state-of-the-art object detection model. Unlike traditional models that process an image in a sliding window manner, YOLO applies a single neural network to the entire image, dividing it into grids. Each grid predicts bounding boxes and the probability of classes within those boxes. Key concepts include:
* YOLO Architecture: It uses a convolutional neural network (CNN) to detect objects and predict their bounding boxes.
* Pretrained Models: Models trained on large datasets like COCO (Common Objects in Context) to detect a variety of objects.
* Real-time Detection: YOLO can process images at high speeds, making it ideal for applications requiring real-time analysis.

**6. Methodology:**

1. **Data Preparation:**

* Image/Video Collection: Gather images or videos containing objects that you want to detect.
* Preprocessing: Resize images to the input size required by the YOLO model (e.g., 640x640). Convert images to a format suitable for the model.

1. **Model Loading:**

* Pretrained Model: Load a pretrained YOLO model, YOLOv8, from the ultralytics library. YOLO models are typically pretrained on large datasets like COCO.

1. **Inference:**

* Input Image to Model: Pass the input image to the YOLO model. The model processes the image and outputs predictions.
* Bounding Box Prediction: YOLO divides the image into a grid (e.g., 13x13, 26x26). Each cell in the grid predicts a set number of bounding boxes, object confidence scores, and class probabilities.
* Object Confidence Score: Represents how confident the model is that an object is present in a particular bounding box.
* Class Probabilities: Likelihood that a detected object belongs to a specific class (e.g., person, car, dog).

1. **Post-Processing:**

* Non-Maximum Suppression (NMS): Removes redundant bounding boxes with lower confidence scores. NMS ensures that only the most relevant boxes are retained for each detected object.
* Thresholding: Set a confidence threshold (e.g., 0.5) to filter out weak detections and focus only on objects with high confidence scores.

1. **Visualization:**

* Draw Bounding Boxes: Use OpenCV to draw bounding boxes around detected objects.
* Labeling: Display the class name and confidence score on top of each bounding box.
* Display Results: Show the annotated image or video stream with detected objects.

1. **Evaluation:**

* Metrics: Calculate accuracy, precision, recall, and F1-score based on the predictions and ground truth labels.
* Qualitative Analysis: Evaluate how well the model performs on different types of images and adjust parameters as needed.

**7. Advantages:**

* **Real-time Detection:** Capable of processing images quickly, making it suitable for video feeds.
* **High Accuracy:** Even with a single forward pass, YOLO can detect multiple objects with good precision.
* **Pretrained Models:** Leverages large datasets, allowing users to use out-of-the-box detection without needing extensive training.

**8. Limitations:**

* **Struggles with Small Objects:** YOLO’s grid-based approach can sometimes miss smaller objects due to spatial constraints.
* **Trade-off Between Speed and Accuracy:** While faster than many detection models, YOLO might compromise slightly on precision.
* **Complex Objects:** It can be less effective when detecting complex or overlapping objects.

**9. Applications:**

* **Autonomous Vehicles:** Detecting pedestrians, vehicles, and obstacles in real-time.
* **Surveillance:** Monitoring objects and people in security systems.
* **Healthcare:** Detecting abnormalities in medical imaging (e.g., X-rays, MRIs).
* **Retail:** Product detection and inventory management using cameras.
* **Gaming and AR/VR:** Real-time interaction with virtual environments through object tracking.

**10. Working/Algorithm:**

**Step 1:** Import required libraries (ultralytics.YOLO, cv2).  
**Step 2:** Load pretrained YOLOv8 model (YOLO('yolov8n.pt')).  
**Step 3:** Initialize webcam using cv2.VideoCapture(0).  
**Step 4:** For each frame:

Capture image from webcam.

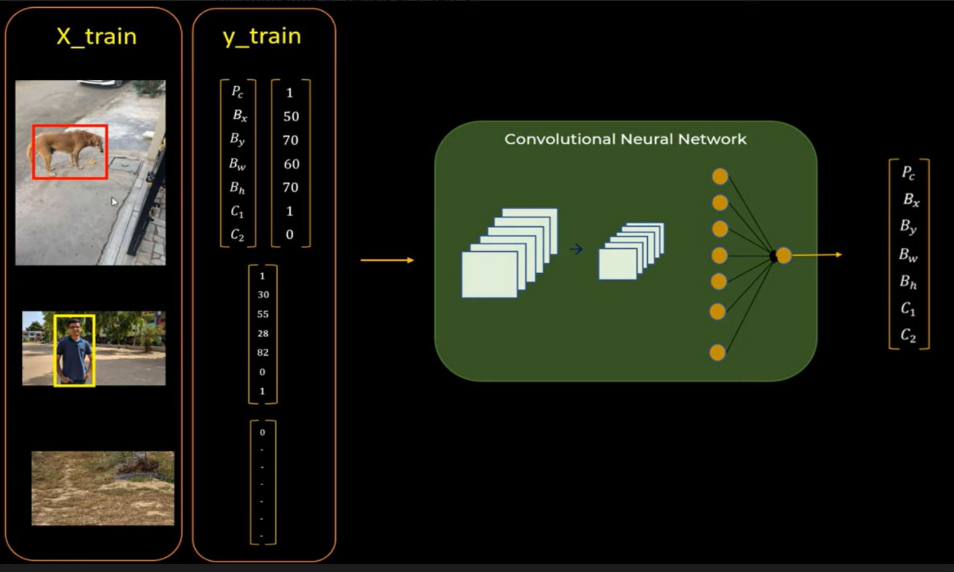
Pass frame into YOLO model to detect objects.

Plot bounding boxes and labels on detected objects.

Display annotated frame in OpenCV window.  
**Step 5:** Exit loop when user presses q.  
**Step 6:** Release webcam and close all windows.

* The final output is an annotated image or video stream showing detected objects with their corresponding labels and bounding boxes.

**11. Diagram:**

****

**12. Conclusion:**

* Object detection using YOLO and pretrained models allows for efficient and effective identification of objects in images and videos. By leveraging the speed and accuracy of YOLO, various real-time applications are possible. This practical assignment demonstrates the implementation of YOLO for detecting multiple objects with high accuracy. Despite some limitations in detecting small or overlapping objects, YOLO remains a popular choice for object detection tasks in diverse fields.