**Assignment No. 7**

**YOLO Object Detection**

**Problem Statement:**

Object detection using YOLO and Pretrained Model.

**Objective:**

* Understand the architecture and working of the YOLO (You Only Look Once) model.
* Implement YOLO using a pretrained model for object detection.
* Evaluate the performance on images or videos and use it for real-time detection.

**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 (e.g., NVIDIA CUDA GPU) for faster inference
  + At least 8 GB RAM
  + Webcam or external camera for real-time object detection

**Libraries Used:**

* **ultralytics:** For YOLO implementation.
* **NumPy:** For numerical operations.
* **OpenCV:** For image and video processing.
* **torch and torchvision:** For handling deep learning models.
* **Matplotlib:** To visualize detected objects.
* **PIL (Python Imaging Library):** To handle image input.

**Theory:**

**YOLO (You Only Look Once):**

* A fast and accurate object detection model. Unlike other methods, YOLO applies a single neural network to the entire image, dividing it into grids, each grid predicting bounding boxes and class probabilities. It’s popular for real-time object detection due to its speed.
* **Architecture:**
  + YOLO uses a convolutional neural network (CNN).
  + Pretrained models like YOLOv8 are trained on large datasets (COCO, Pascal VOC) and can detect multiple object types.

**Real-time Detection:** YOLO’s architecture allows high-speed detection, making it suitable for real-time applications.

**Methodology:**

1. **Data Preparation:**
   * Collect or use provided image/video datasets containing objects.
   * Preprocess images to the required size for YOLO (e.g., 640x640 pixels).
2. **Model Loading:**
   * Use a pretrained YOLOv8 model from the ultralytics library.
   * Load the pretrained model weights for efficient object detection.
3. **Inference:**
   * Pass input images to the YOLO model to generate predictions.
   * YOLO divides the image into a grid and predicts bounding boxes for objects within each grid cell.
   * Confidence scores indicate how likely the object is to belong to a certain class.
4. **Post-Processing:**
   * Use Non-Maximum Suppression (NMS) to filter overlapping bounding boxes.
   * Apply a confidence threshold (e.g., 0.5) to keep the most relevant object detections.
5. **Visualization:**
   * Use OpenCV to draw bounding boxes and display labels and confidence scores on the image or video.
   * Display the final annotated image or video.
6. **Evaluation:**
   * Measure the model's performance using metrics like accuracy, precision, recall, and F1-score.
   * Perform qualitative analysis on different datasets and videos.

**Advantages:**

* **Real-time Detection:** YOLO can process images at very high speeds.
* **High Accuracy:** Pretrained on large datasets, YOLO can detect multiple objects with good precision.
* **Versatility:** YOLO works well with both image and video inputs, making it suitable for many applications.

**Limitations:**

* **Small Object Detection:** YOLO may miss smaller objects due to the grid-based approach.
* **Trade-off Between Speed and Precision:** While fast, there may be some trade-off in precision.
* **Overlapping Objects:** YOLO can struggle with detecting complex overlapping objects.

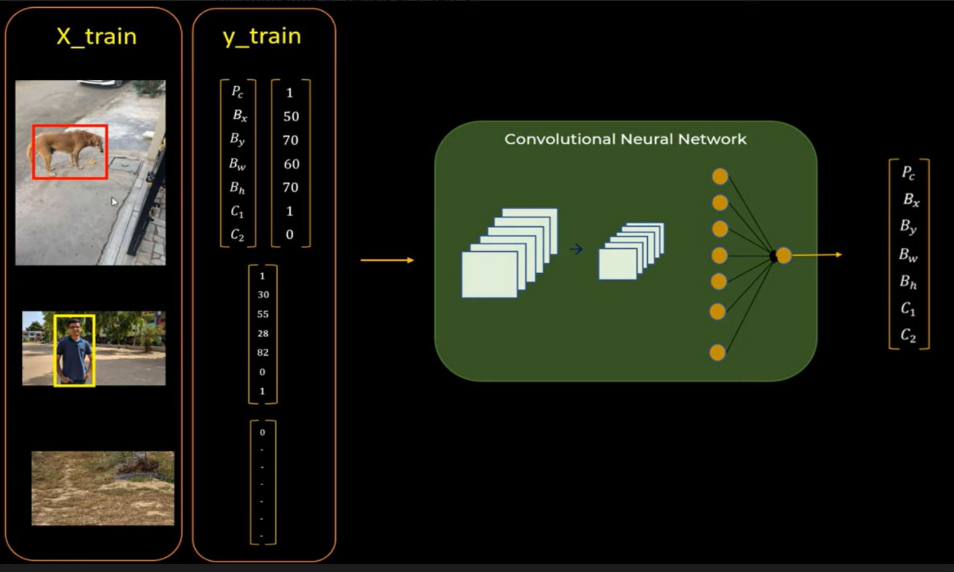
**Applications:**

* **Autonomous Vehicles:** Detect obstacles, pedestrians, and vehicles.
* **Surveillance Systems:** Real-time monitoring of objects or individuals.
* **Healthcare:** Detecting features in medical images.
* **Retail:** Object detection in products and inventory management.
* **Gaming and AR/VR:** Real-time interaction through object detection and tracking.

**Working/Algorithm:**

* **Initialization:**
  + Load YOLOv8 model weights.
  + Define image size (e.g., 640x640).
* **Preprocessing:**
  + Convert the image to a tensor and normalize pixel values.
  + Resize the image to YOLO’s input size.
* **Prediction:**
  + Pass the preprocessed image through the YOLO model.
  + YOLO divides the image into an S x S grid (e.g., 13x13) and predicts bounding boxes.
* **Post-Processing:**
  + Apply Non-Maximum Suppression (NMS) to retain only the most confident bounding boxes.
  + Filter predictions based on confidence threshold.
* **Drawing and Visualization:**
  + Draw bounding boxes around detected objects and label them using OpenCV.
  + Display the annotated image or video stream.

**Diagram:**

****

**Conclusion:**

Object detection using YOLO and pretrained models like YOLOv8 provides an efficient way to identify objects in images and videos. Its speed makes it ideal for real-time applications, and with proper fine-tuning, YOLO models can achieve good accuracy even in complex environments. Despite limitations such as handling small objects and overlapping entities, YOLO remains a leading choice for object detection tasks across various industries.