# **Assignment 7: Object Detection using YOLO and Pretrained Model**

## **Problem Statement**

Traditional computer vision methods struggle with **real-time object detection** because they require multiple passes for image classification and localization. The challenge is to build a system that can **detect and localize multiple objects in a single image or video frame in real time**.

This assignment focuses on implementing **YOLO (You Only Look Once)** with a pretrained model to detect objects in uploaded images.

## **Objective**

* To understand and implement **YOLO-based object detection**.
* To use a pretrained YOLO model for real-time detection.
* To allow image uploads and visualize **bounding boxes + class labels** on detected objects.

## **Requirements**

* **Operating System**: Windows / Linux / macOS (Google Colab recommended)
* **IDE / Platform**: Jupyter Notebook / Google Colab

### **Libraries and Packages Used**

* **OpenCV** → Image processing, video frame handling.
* **TensorFlow / PyTorch** → Backend for deep learning models.
* **Ultralytics YOLO (PyPI package)** → Pretrained YOLO models (YOLOv8).
* **Matplotlib** → Visualization.
* **NumPy** → Matrix operations.

## **Theory**

### **Definition**

**YOLO (You Only Look Once)** is a state-of-the-art, real-time object detection system. Unlike earlier detectors (R-CNN, Fast R-CNN) that use multiple passes, YOLO divides an image into a grid and predicts **bounding boxes + class probabilities simultaneously** in a single forward pass.

### **Structure**

1. **Input Image** – Resized to fixed dimensions (e.g., 640×640).
2. **Convolutional Layers** – Extract features from image.
3. **Grid Division** – Image is divided into *S × S* grid cells.
4. **Bounding Box Prediction** – Each cell predicts box coordinates (x, y, w, h).
5. **Confidence Score** – Probability of object presence.
6. **Class Probabilities** – Prediction of object category (car, person, dog, etc.).
7. **Output** – Final detections after **Non-Maximum Suppression (NMS)**.

## **Methodology**

1. **Dataset / Pretrained Model**
   * Use **YOLOv8 pretrained weights** (trained on COCO dataset with 80 object classes).
   * Load model directly via ultralytics package.
2. **Model Inference**
   * Upload an image.
   * Pass it through YOLO model.
   * Model outputs bounding box coordinates, confidence scores, and labels.
3. **Visualization**
   * Draw bounding boxes with labels on the image using OpenCV or YOLO’s built-in .plot() method.

## **Advantages**

* **Real-time performance**: YOLO is extremely fast compared to traditional detectors.
* **Single forward pass**: Detects objects in one evaluation, making it efficient.
* **Generalizable**: Works on multiple objects simultaneously.
* **Pretrained models available**: Eliminates the need for large training datasets.

## **Limitations**

* Struggles with **small object detection** (due to grid division).
* Requires high computational power for training from scratch.
* Performance depends on the **quality of pretrained weights**.
* Less accurate than two-stage detectors (e.g., Faster R-CNN) in some cases.

## **Working / Algorithm**

1. Take input image.
2. Resize image to YOLO’s expected input size.
3. CNN backbone extracts deep features.
4. Grid cells predict bounding boxes + class probabilities.
5. Apply **Non-Maximum Suppression (NMS)** to remove duplicate detections.
6. Draw bounding boxes and class labels on detected objects.
7. Display final annotated image.

## **Conclusion**

This assignment successfully implements **YOLO for real-time object detection** using a pretrained model. The system can detect and localize multiple objects within a single uploaded image, proving YOLO’s strength in real-world applications such as **autonomous driving, surveillance, healthcare, and robotics**.