**OBJECT DETECTION**

* **AIM:** Our aim is to identify and localize object within a image.
* Object detection using **YOLO (You Only Look Once)** is an efficient process designed for **real-time detection** of objects in images or video streams. YOLO is a popular object detection model because of its speed and balance between speed and accuracy.
* **STEPS:**

1. **Importing Library**

* **OpenCV**
* **PyTorch**
* **YOLOv5 Repository**
* **Numpy**
* **Matplotlib**
* **Scikit-Learn** (for evaluation and hyperparameter tuning)

1. **Importing dataset**

* In these project we are using image dataset to identify the object present in the image.

1. **Divide the image into Grid.**

* YOLO divides the input image into a grid of cells, usually 13x13 or 19x19.
* By dividing the image into a grid, YOLO can simultaneously detect multiple objects across different regions of the image.
* Each grid cell predicts several bounding boxes and their corresponding class probabilities.

1. **Predict Bounding Boxes**

* For each grid cell, YOLO predicts a set number of bounding boxes (usually 3-5).
* YOLO predicts multiple bounding boxes for each grid cell to handle overlapping or multiple objects. The confidence score indicates how likely it is that the box contains an object, regardless of the object class.

1. **Class Prediction**

* YOLO assigns class probabilities to each predicted bounding box.
* This step classifies the object inside the bounding box (e.g., car, dog, person).
* For each bounding box, YOLO predicts class probabilities, representing the likelihood that the box belongs to a particular object class. This allows YOLO to not only locate objects but also classify them.

1. **Non Maximum Supression (NMS)**

* Apply non-maximum suppression to remove overlapping bounding boxes.
* NMS eliminates multiple boxes that predict the same object, keeping only the box with the highest confidence score.
* When YOLO predicts multiple boxes for the same object, non-maximum suppression ensures that only the best bounding box (highest confidence) is retained. This reduces redundancy and improves detection accuracy.

1. **Final Output**

* Output the final bounding boxes, confidence scores, and class labels.
* The final step is to output the detected objects' positions, sizes, and classes.

**Q: Why Yolo?**

 **Real-time Detection**: YOLO is designed for real-time applications like autonomous driving or drone navigation. Its fast inference speed makes it suitable for live video feeds.

 **Single Pass Architecture**: YOLO performs object detection in a single forward pass, which is significantly faster compared to region-based approaches like Faster R-CNN.

 **Efficient and Lightweight**: YOLO’s architecture is computationally efficient, allowing it to run on low-power devices like drones or edge devices.

 **High Speed-Accuracy Trade-off**: While YOLO may not be as accurate as some slower, two-stage models, it provides a good balance between speed and accuracy, making it ideal for applications where both are important.

 **Versatility**: YOLO is capable of detecting multiple objects of different sizes in the same image, making it a flexible tool for various tasks such as security surveillance, traffic monitoring, and more.