Name: Aziz Sayyad
Roll No: 281069
Batch: A3

Assignment No: - 7

Problem Statement:
Implement object detection using YOLO (You Only Look Once) with a pretrained deep learning model to identify and locate multiple objects within images or video streams.

Objective:

- To understand the working of YOLO object detection algorithm.
- To apply a pretrained YOLO model for detecting objects in images/videos.
- To preprocess input images for YOLO model compatibility.
- To evaluate detection accuracy using bounding boxes and class labels.
- To visualize detected objects in real-time using OpenCV.

S/W Packages and H/W apparatus used:

- **Operating System:** Windows/Linux/MacOS
- **Kernel:** Python 3.x
- **Tools:** Jupyter Notebook, Anaconda, or Google Colab
- **Hardware:** CPU with minimum 8GB RAM; GPU recommended for faster inference
- **Libraries and packages used:** OpenCV, TensorFlow/Keras or PyTorch, NumPy, Matplotlib, Pretrained YOLO weights and configuration files

Theory:
YOLO (You Only Look Once) is a real-time object detection algorithm that frames object detection as a single regression problem, directly predicting class probabilities and bounding box coordinates from entire images in one evaluation.

Key Concepts:

- **Single Neural Network:** YOLO applies a single CNN to the whole image.
- **Grid Division:** The image is divided into an $S \times S$ grid, and each grid cell predicts bounding boxes and class probabilities.
- **Bounding Box Prediction:** Each box is defined by coordinates (x, y, w, h) and confidence score.
- **Class Prediction:** Probability distribution over possible object classes.
- **Pretrained Models:** YOLOv3, YOLOv4, YOLOv5, and YOLOv8 are commonly used pretrained models available on COCO dataset.

Methodology:

1. **Data Acquisition:** Select input images or video stream for object detection.
2. **Model Preparation:**

  - Download pretrained YOLO weights and configuration files.
  - Load COCO class labels.

3. **Preprocessing:**
   - Resize input image to YOLO’s required dimensions (e.g., 416×416).
   - Normalize pixel values.
4. **Model Execution:**
   - Pass input through YOLO CNN.
   - Generate bounding boxes, confidence scores, and class probabilities.
5. **Postprocessing:**
   - Apply Non-Max Suppression (NMS) to remove duplicate overlapping boxes.
   - Retain boxes with high confidence scores.
6. **Output Visualization:** Draw bounding boxes and labels on detected objects using OpenCV.
7. **Evaluation:** Measure detection accuracy, precision, recall, and inference speed.

Advantages:

- Real-time object detection with high speed.
- Single CNN makes it computationally efficient.
- Can detect multiple objects in a single image.
- Pretrained models save time and resources.
- Scalable to different versions (YOLOv3, YOLOv4, YOLOv5, YOLOv8).

Limitations:

- Accuracy may be lower for small objects compared to other detectors.
- Requires GPU for high-speed performance.
- Complex architecture, harder to implement from scratch.
- Pretrained models limited to fixed classes (e.g., 80 classes in COCO).
- Sensitive to input image size and aspect ratio.

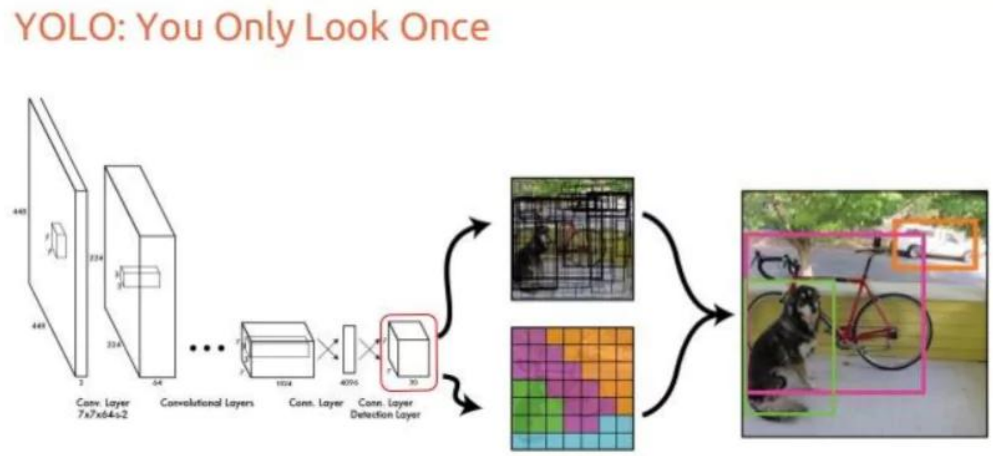
Applications:

- Real-time surveillance and security monitoring.
- Autonomous driving (pedestrian, traffic sign, and vehicle detection).
- Retail analytics (customer tracking, product detection).
- Robotics and drones (navigation and obstacle detection).
- Medical imaging for detecting anomalies.

Working / Algorithm:

1. Import required libraries (OpenCV, NumPy, etc.).
2. Load YOLO pretrained weights and configuration files.
3. Load COCO class labels.
4. Read input image/video.
5. Preprocess input (resize, normalize).
6. Pass through YOLO CNN to get predictions.
7. Apply Non-Max Suppression to filter bounding boxes.
8. Draw bounding boxes and labels on detected objects.
9. Display or save the output.

Diagram:



Conclusion:

Object detection using YOLO and pretrained models provides a fast and efficient solution for real-time applications. By leveraging pretrained weights, YOLO eliminates the need for large-scale training while still achieving high detection accuracy. Although it faces challenges with small objects and requires powerful hardware for best performance, YOLO remains one of the most widely used algorithms in object detection and computer vision applications.