**YOLO (You Only Look Once)**

**Object Detection in YOLO:**

The input given is an image along with ground truth values of the bounding boxes.The entire input image is divided into a square grid, the grid cell containing the center of the object is responsible for detecting that object. Each grid cell will predict B bounding boxes and the confidence scores associated with those boxes. This score indicates the accuracy of the box and how confident the model is that the box contains the object. Confidence score is IoU of predicted values and the ground truth values of the bounding box.

The YOLO family of algorithms is essentially an object regression problem. The detection process involves dividing the input image into a certain grid according to certain specifications, traversing the entire image once, and when an object is detected in each grid it is drawn into a prediction box based on the grid's anchor box, which in turn predicts the object result directly.

The YOLO v1 algorithm divides the image into 7 × 7 grids, each with two prediction boxes, and gives the results directly based on the IOU (Intersection over Union) of the prediction box .

YOLO v2 divides the image into 13 × 13 grids, each with five a priori frames, and draws the final prediction boxes based on the results of the anchor box by bounding box regression .

YOLO v3 and YOLO v4 add three different scale prediction networks to the previous YOLO versions, this is used to detect medium-scale objects and small objects within large objects.

Each prediction network consists of three predetermined anchor boxes, and when the object is detected, bounding box regression is performed on the different prediction networks, and the final prediction box is output.

The detection network of YOLO v4 mainly consists of the CSPDarknet53 network, SPP (Spatial Pyramid Pooling), PANet (Path Aggregation Network). CSPDarknet53 is the backbone feature extraction network, which mainly consists of the BasicConv convolutional network module and CSP residual network module. SPP consists of the BasicConv convolutional network module and max pooling. PANet mainly fuses features of different feature layers by upsampling and downsampling to obtain higher semantic information.

Loss function

The three primary parts of the YOLO v4 loss function are the

1. confidence loss
2. classification loss
3. regression loss.

IOU is the concept of intersection and merge.

IOU=A∩B / A∪B

The loss function for YOLO v4 is obtained by summing the three loss-functions above:

LOSS=LOSSCls+LOSSconf+LOSSCIOU

What is YOLOv4?

YOLOv4 is a **SOTA (state-of-the-art)** real-time **Object Detection** model. It was published in April 2020 by Alexey Bochkovsky; it is the 4th installment to YOLO.YOLO is a one-stage detector. The One-stage method is one of the two main state-of-the-art methods used for the task of Object Detection, which prioritizes on the inference speeds. In one-stage detector models ROI (Region of Interest) is not selected, the classes and the bounding boxes for the complete image are predicted. Thus, this makes them faster than two-stage detectors. Other examples are FCOS, RetinaNet, SSD

The first version of YOLO was written in the Dark Net Framework (which is a high performance open source framework for implementing neural networks written in C and CUDA). DarkNet is typically a backbone network.

It divides the object-detection task into a regression task followed by a classification task. Regression predicts classes and bounding boxes for the whole image in a single run and helps to identify the object position. Classification determines the object's class.

Steps To store the labels in Json File:

1. The code first imports the necessary libraries, including cv2, which is the OpenCV library that provides the functionality to load and use the YOLOv4 model, and json, which is used to write the detected objects to a JSON file.
2. Next, it loads the model configuration and weights files using the cv2.dnn.readNetFromDarknet() function and assigns the result to the net variable.
3. Then it defines the input size for the model and class labels, the class labels are read from obj.names file.
4. The video is opened using cv2.VideoCapture(video\_path) and a loop runs for each frame in the video.
5. Inside the loop, the code reads the next frame from the video using cap.read().
6. Then, it prepares the frame for the model by creating a blob from the image using cv2.dnn.blobFromImage(frame, size=input\_size, depth=cv2.CV\_8U) and passing it to the model as input using net.setInput(blob).
7. Next, the model performs object detection on the input frame by calling out = net.forward().
8. The model's output is parsed to extract the object labels and bounding box coordinates.
9. For each detected object, the code creates a dictionary containing the object's class label, confidence level, and bounding box coordinates.
10. These dictionaries are appended to a list called objects
11. Finally, the code uses the json.dump() function to write the objects list to a JSON file.