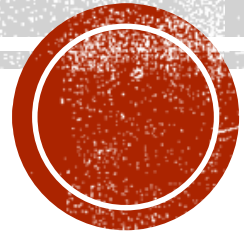
YOU ONLY LOOK ONCE: UNIFIED, REAL-TIME OBJECT DETECTION

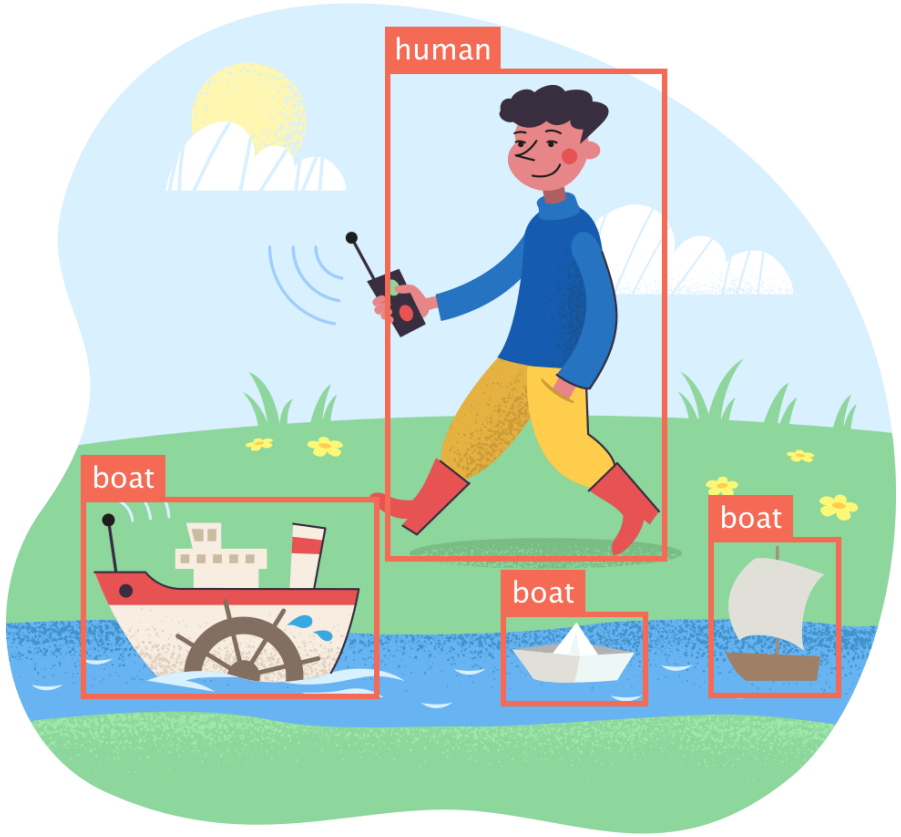
Proceedings of the IEEE conference on computer vision and pattern recognition.

- Charishma Paruchuri
- Deepika Narne



CONTENT :

- Introduction
- Goal of the paper
- Prior work
- YOLO Architecture
- Proposed Methodology
- Our Contributions
- Evaluation & Results
- Conclusion



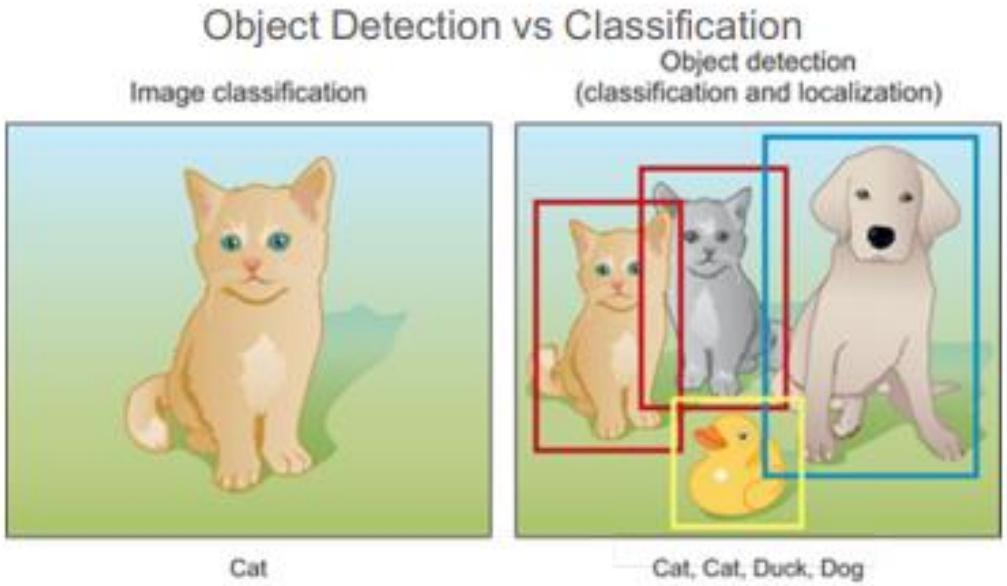
INTRODUCTION

Applications :

- Self-driving cars
- Security and surveillance
- Robotics
- Gaming
- Many more..

GOAL OF THE PAPER

- Object detection is the problem of both locating AND classifying objects.
- Goal of YOLO algorithm is to do object detection both fast AND with high accuracy.

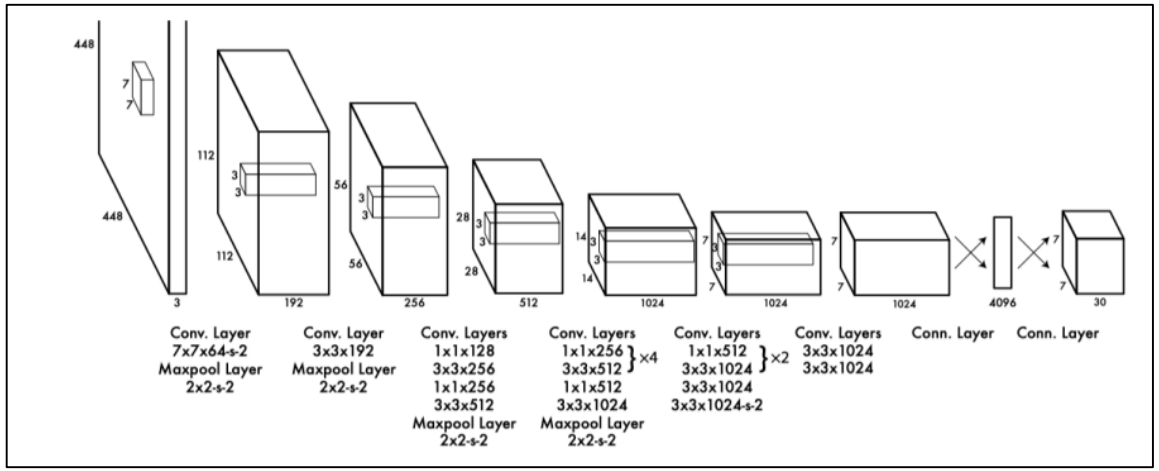


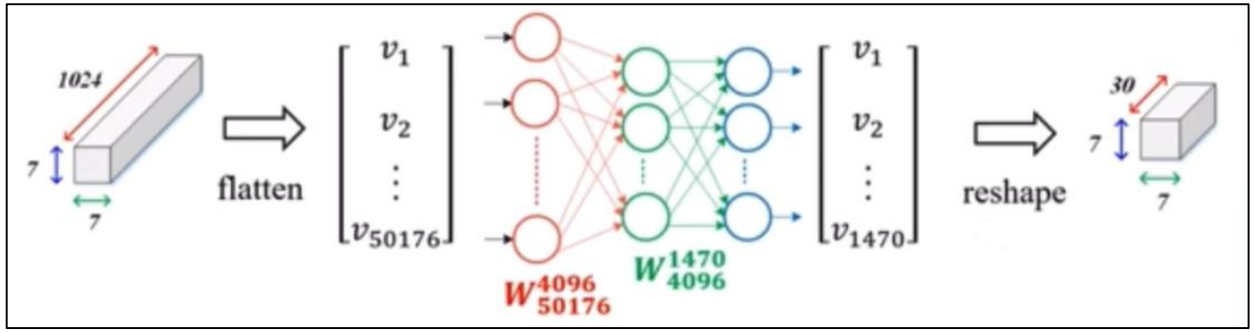
PRIOR WORK

- **Region-based Convolutional Neural Networks (R-CNN):** R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. While R-CNN achieved state-of-the-art performance at the time, it was relatively slow and computationally expensive.
- **Fast R-CNN:** Fast R-CNN addressed some of the speed and computational efficiency issues of R-CNN by performing feature extraction and region proposal simultaneously. However, it still relied on a two-stage pipeline involving region proposal and classification.
- **Faster R-CNN:** Faster R-CNN further improved upon the speed and accuracy of Fast R-CNN by introducing a region proposal network (RPN) that shared convolutional features with the object detection network. The drawback is it looses accuracy.

YOLO ARCHITECTURE

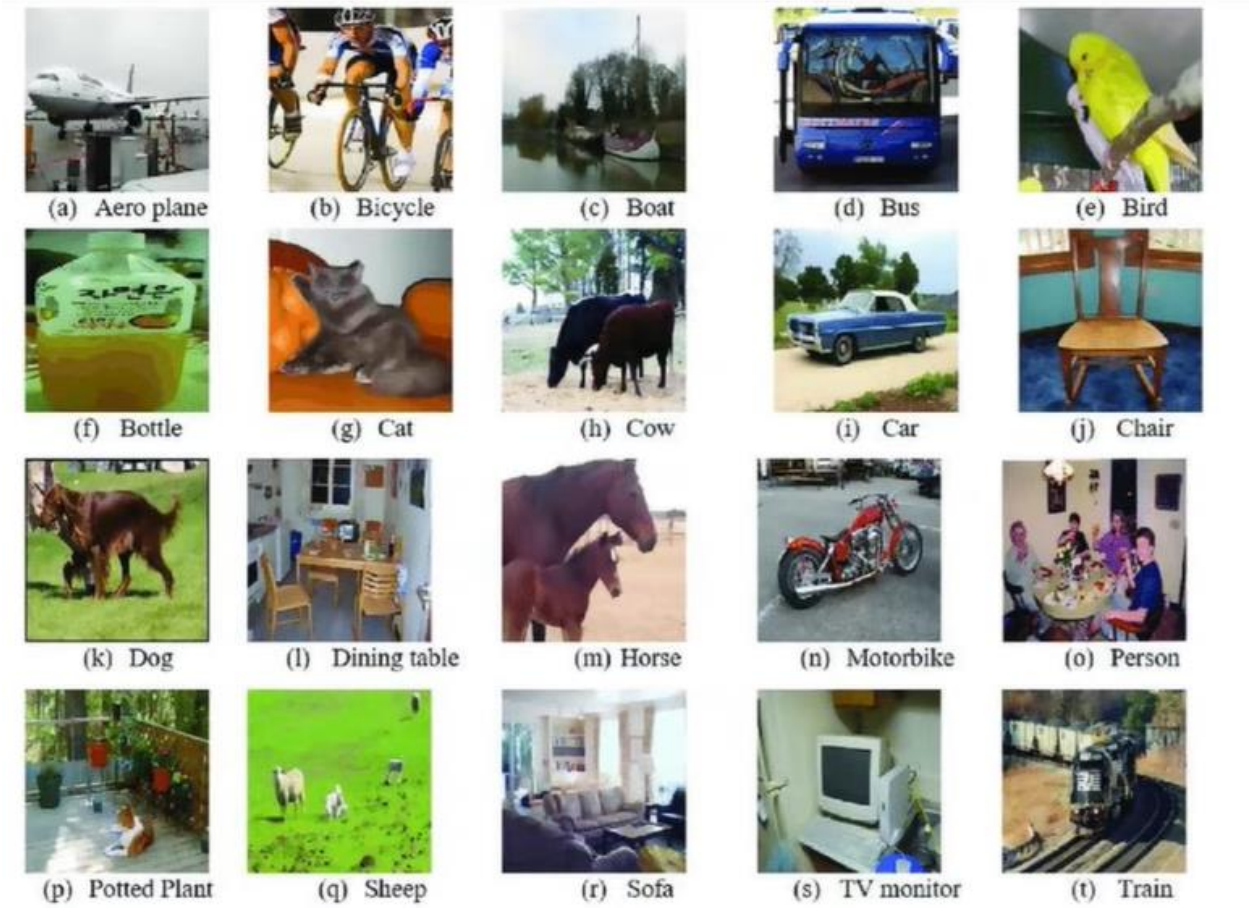
- Inspired by GoogleNet Model.
- The initial convolutional layers of the network extract feature from the image while the fully connected layers predict the output probabilities and coordinates.
- It has 24 convolutional layers followed by 2 fully connected layers.
- Flatten the last conv map 7x7x1024 to 50176 feature vector before passing through fully connected layers.





DATA SET

- PASCAL VOC (visual object class) Dataset
- 20 Classes



PROPOSED METHODOLOGY – YOLO V1

- In this paper, You Only Look Once version 1 is proposed.
- It looks at the entire image at once, and only once — hence the name You Only Look Once — which allows it to capture the context of detected objects.
- ➢ It runs a single convolutional network on the input images.
- ➢ YOLO v1 is a regression problem because the network directly predicts the bounding box coordinates and object class probabilities as regression targets.

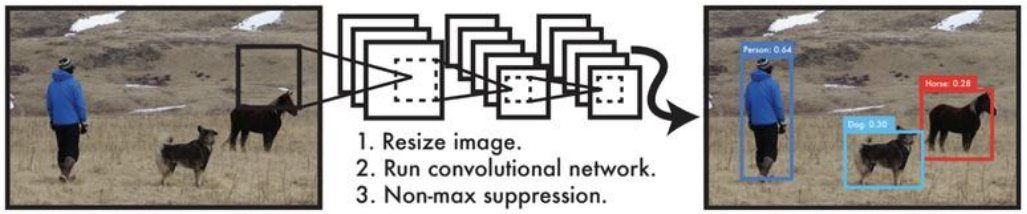
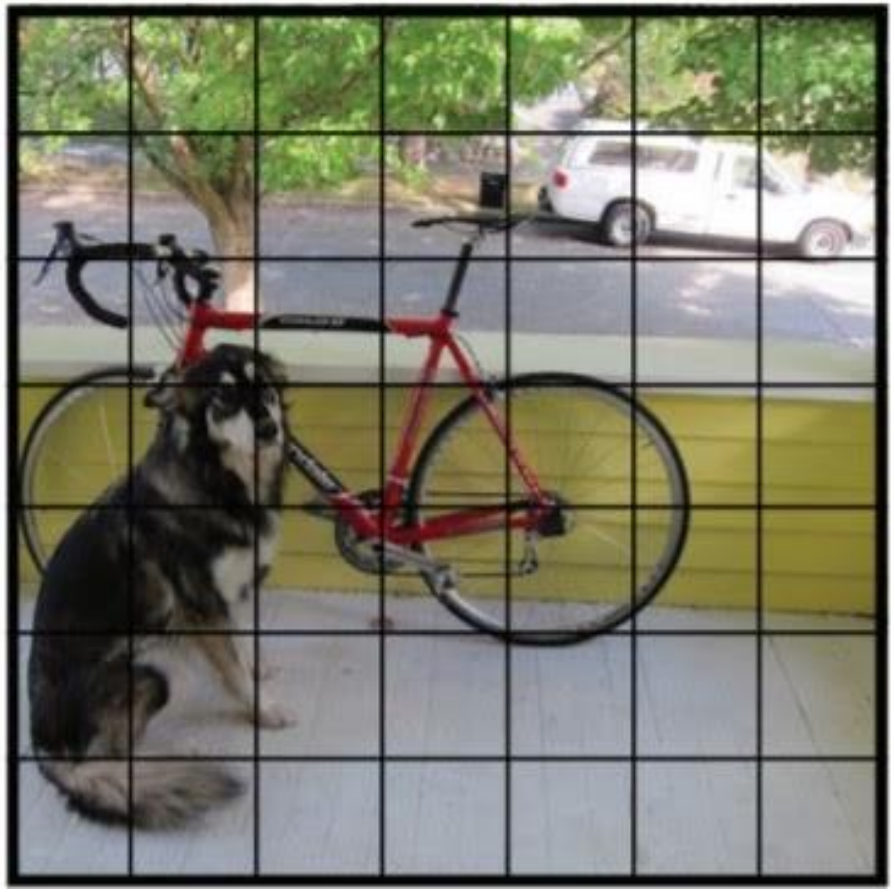


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

STEP - 01

- Split the image into S x S grid. (S = 7 in the paper)



- Each cell will output a prediction with a corresponding bounding box.

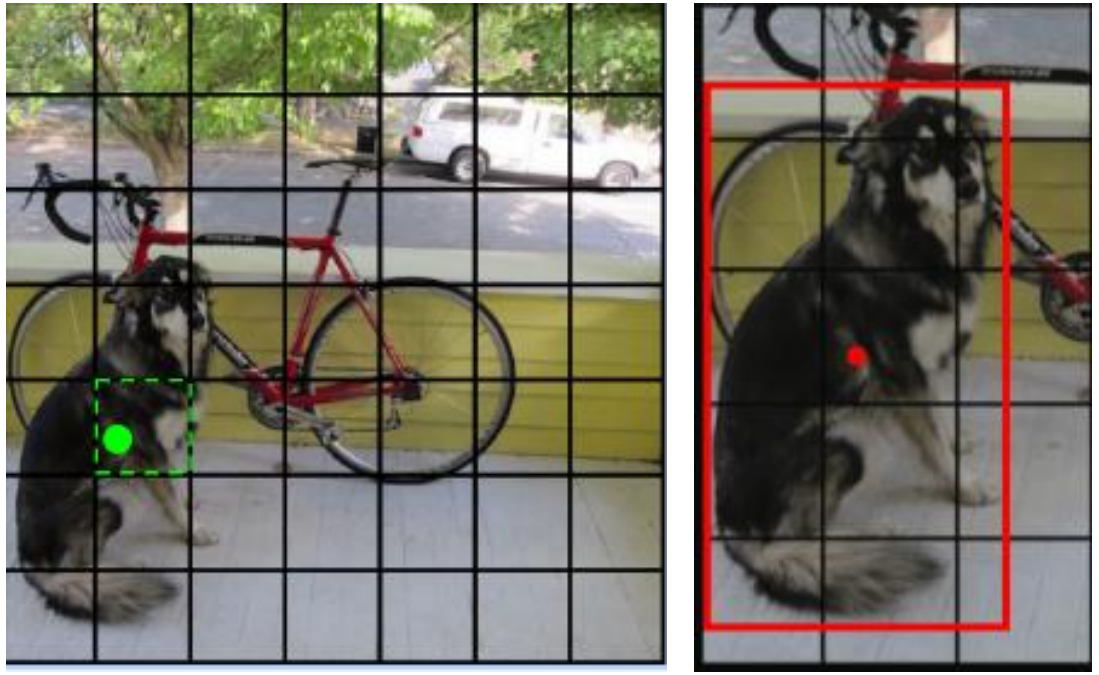
- Each object is in multiple bounding boxes, how do we get a single bounding box ?

STEP - 02

- Find one cell that is responsible for outputting that object.
- Cell that contain object midpoint is the cell that is responsible.
- A bounding box prediction consists of 5 elements:

  - x, y, w, h values

  - P(Object) - the probability that there is an object whose center falls within this grid cell.
- x and y values are relative to the grid cell but the width and height (w,h) are predicted relative to the entire image.
- Given those four values (x,y,w,h), we can calculate the location and size of the bounding box within the image.



STEP- 03

- Each grid cell predicts 2 bounding boxes and confidence scores for those boxes.
- These confidence scores reflect how confident the model is that the box contains an object.

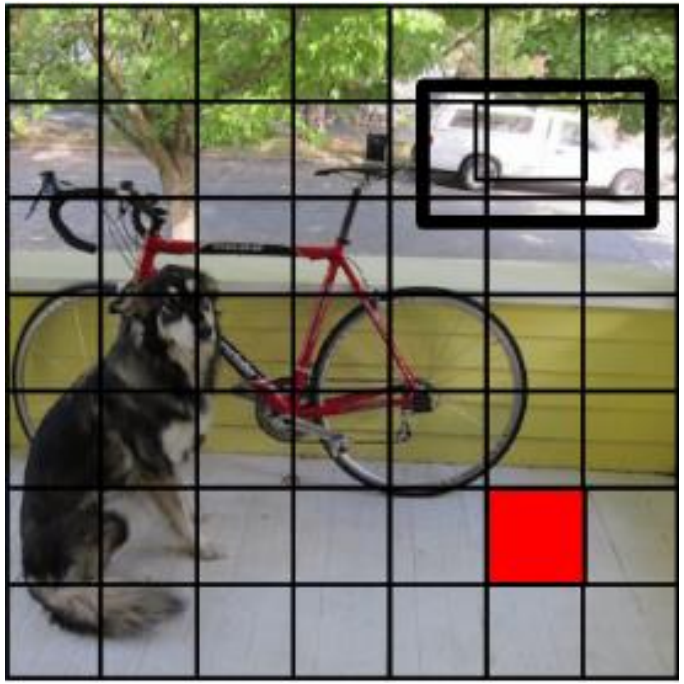
➢ **P(Object)** - The probability of an object having the center in the grid cell.

➢ **IOU (Intersection over Union)** - overlap between predicted bounding box and ground truth box

➢ If P(Object) is closer to 1, the model is more confident that an object's center exists within the grid cell.

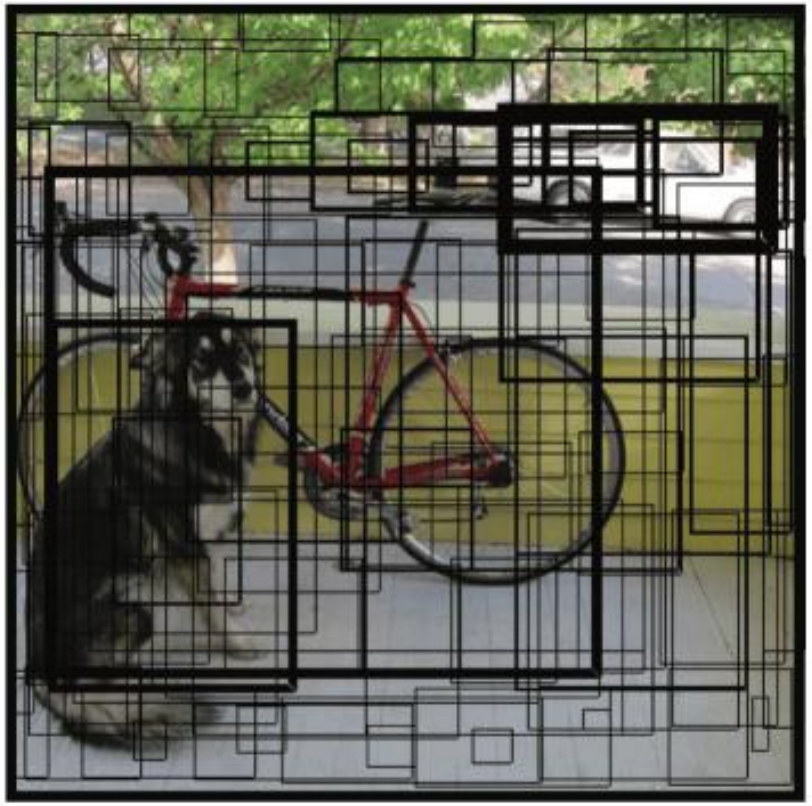
➢ If IoU(bbox, gt) is closer to 1, the model is more confident that the bounding box well approximates the ground truth.

➢ confidence_score = P(Object) × IoUtruthpred



CONT

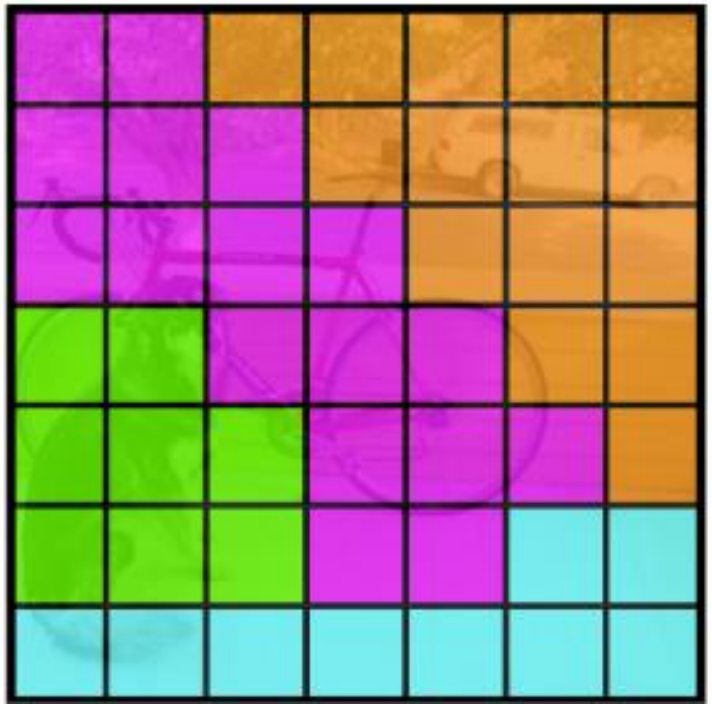
All bounding boxes may look like the below. Bolder lines indicate higher confidence scores.

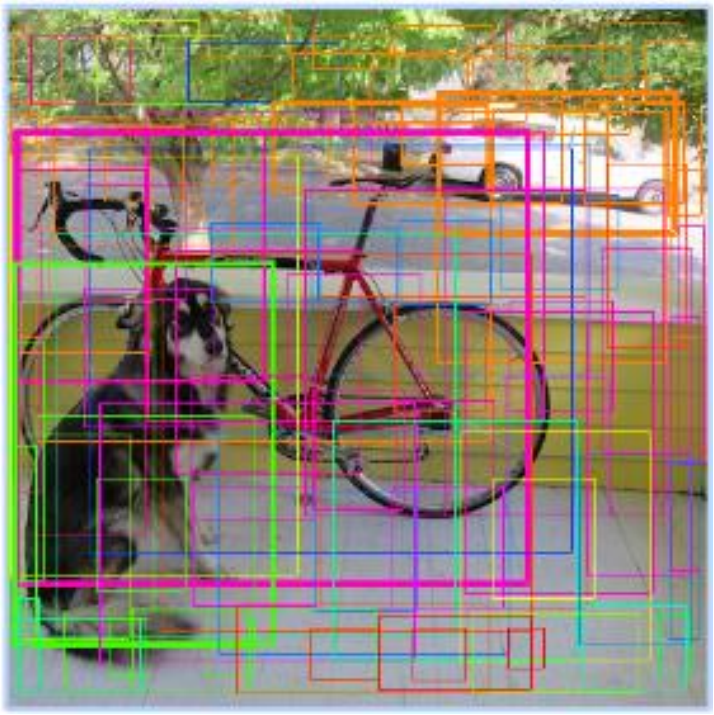


STEP - 04

Predicting Class probabilities –

- YOLO predicts 20 class probabilities per grid cell. It tells what kind of object is most likely to exist in a grid cell. Each grid cell color indicates the most probable class for that cell.





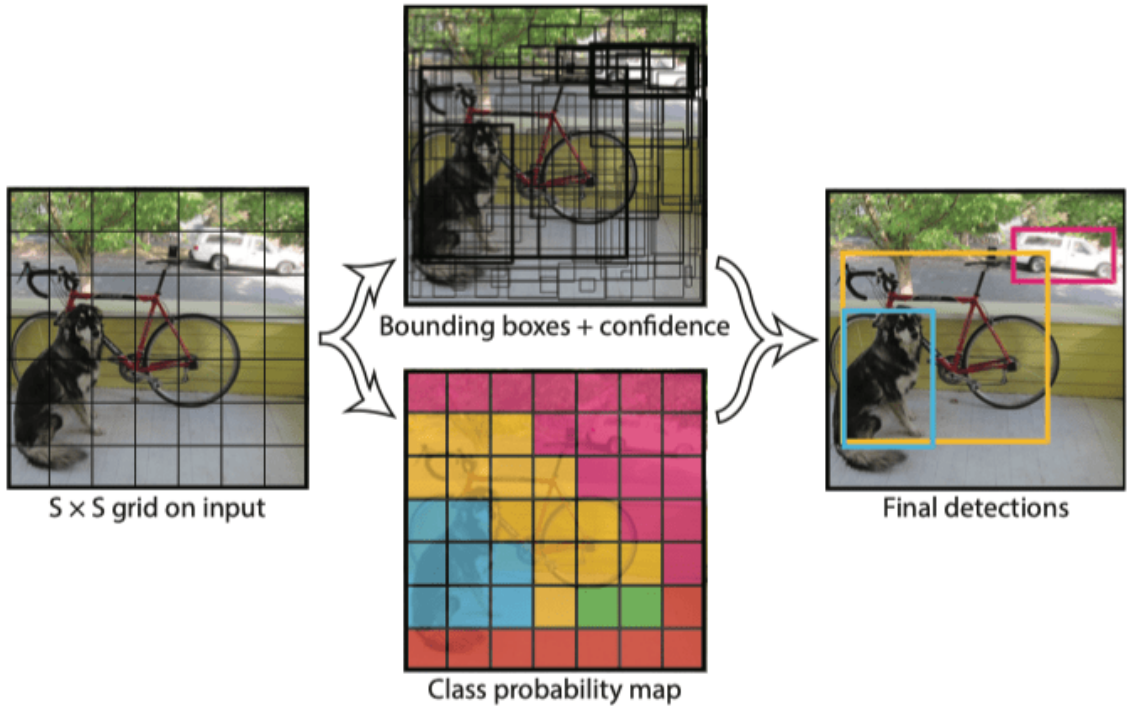
STEP - 05

Non-maximal suppression

Discard bounding box with high overlap

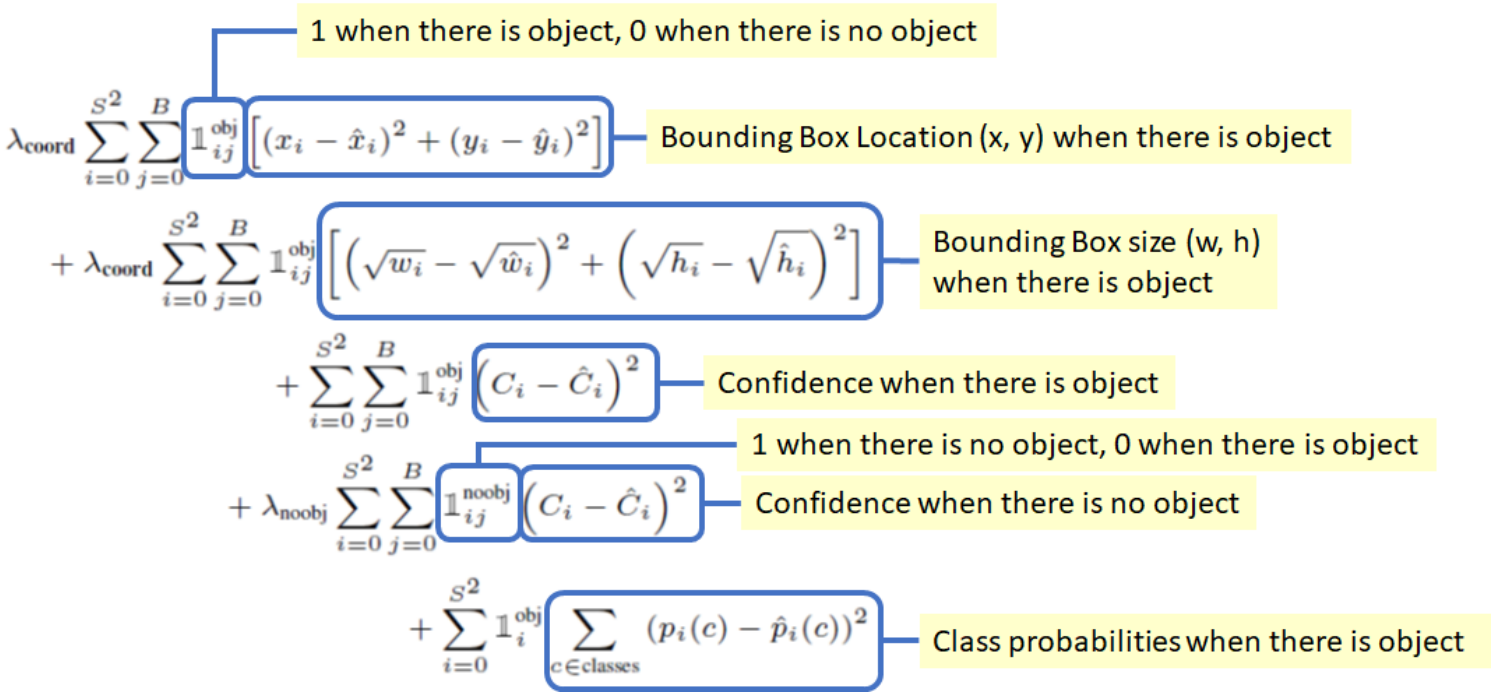
keep the bounding box with highest confidence

Repeat the process until there are no more bounding boxes left to process.

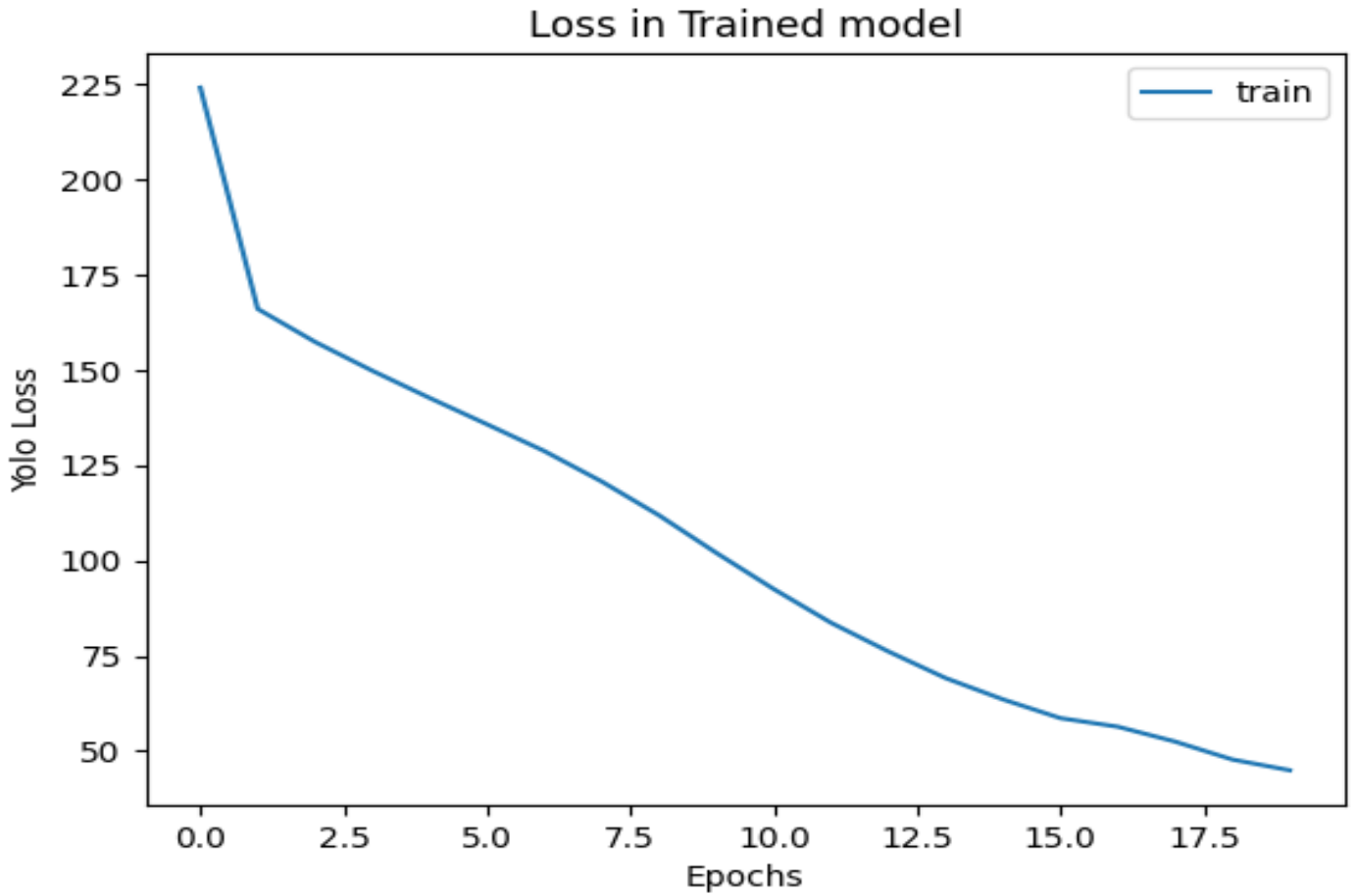


LOSS FUNCTION

- YOLO is trained to minimize a loss function that combines localization and classification losses.
- LOSS = Object loss + classification loss + Box regression loss



LOSS


Loss in Trained model
train
225
200
175
150
125
100
75
50
Yolo Loss
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

OUR CONTRIBUTIONS

- Batch Normalization is used in the network to improve its performance and stability during training. It is used to normalize the activations of the previous layer effectively centering and scaling the distribution of the activations.
- Faster training: By normalizing the inputs to each layer, batch normalization reduces the amount of internal covariate shift, which is the phenomenon of the distribution of the input values changing during training. This can help speed up training by reducing the number of iterations required to reach convergence.
- Improved generalization: By reducing internal covariate shift, batch normalization can help the network generalize better to new data by making it less sensitive to the distribution of the training data.
- Higher learning rates: Batch normalization can allow the network to use higher learning rates during training, which can help speed up training and reduce the risk of getting stuck in local optima

OUR RESULTS

colab.research.google.com/drive/1yAU3nXsQbSdJI-IAE9JrKxRqFY9xpeYI#scrollTo=pJmIOsgNvDAb

yolo_v1_C&D.ipynb

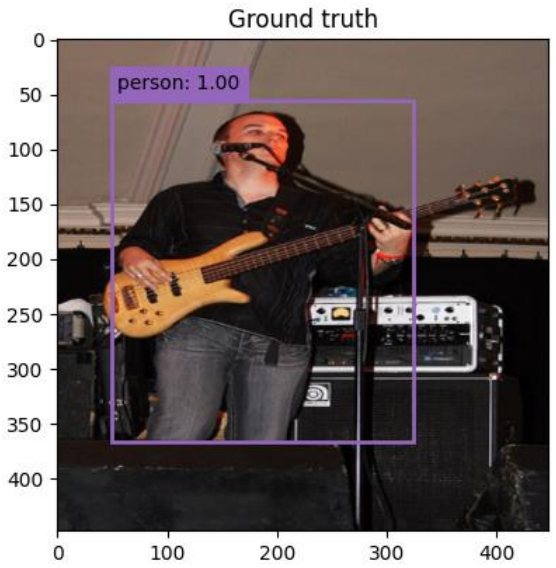
File Edit View Insert Runtime Tools Help Saving...

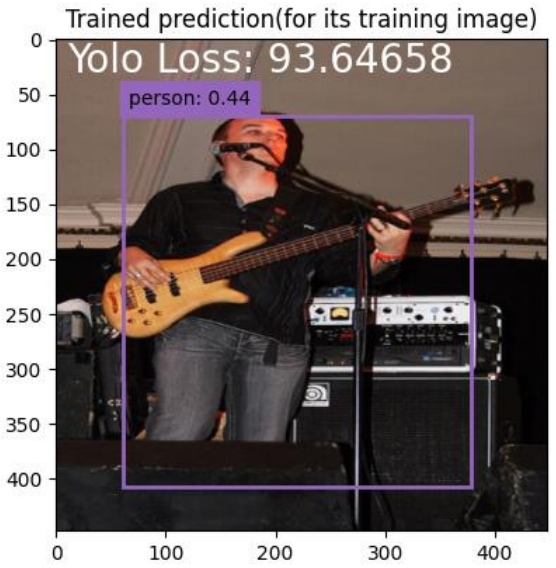
Comment Share

+ Code + Text

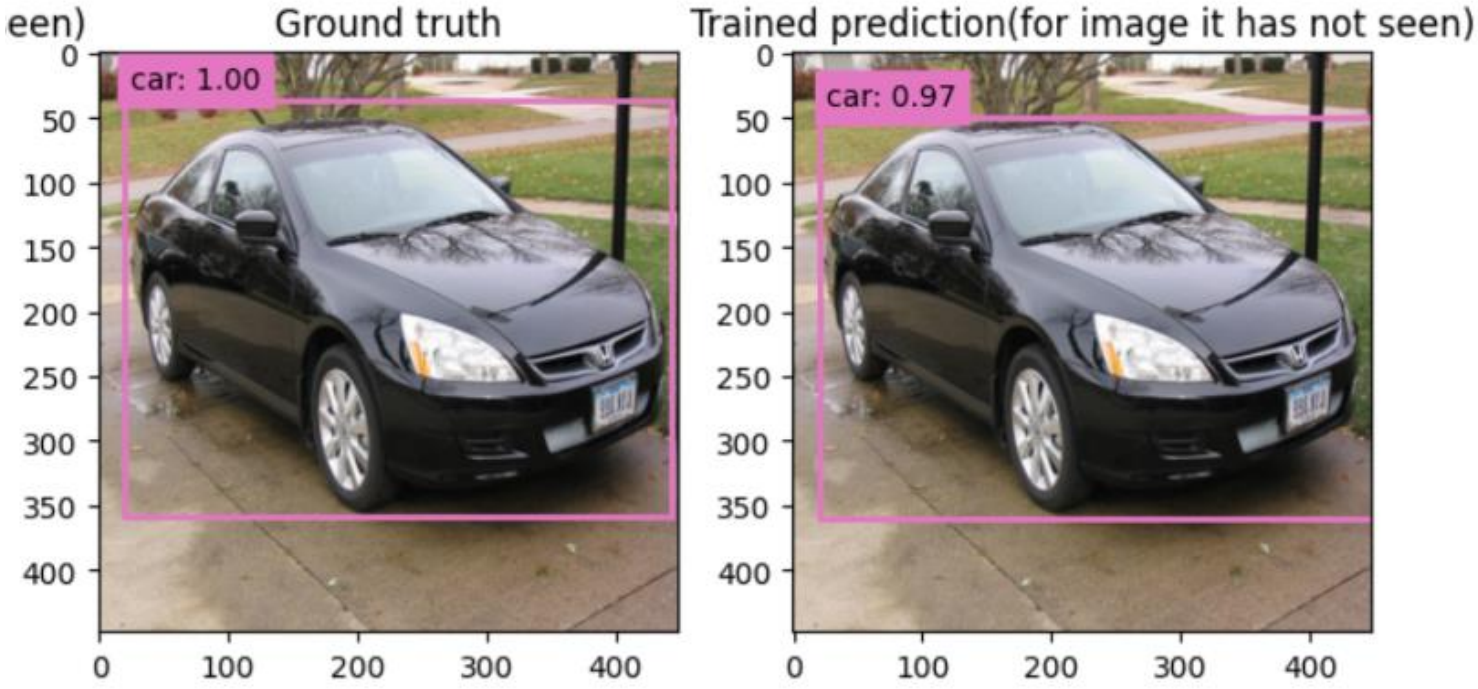
Connect

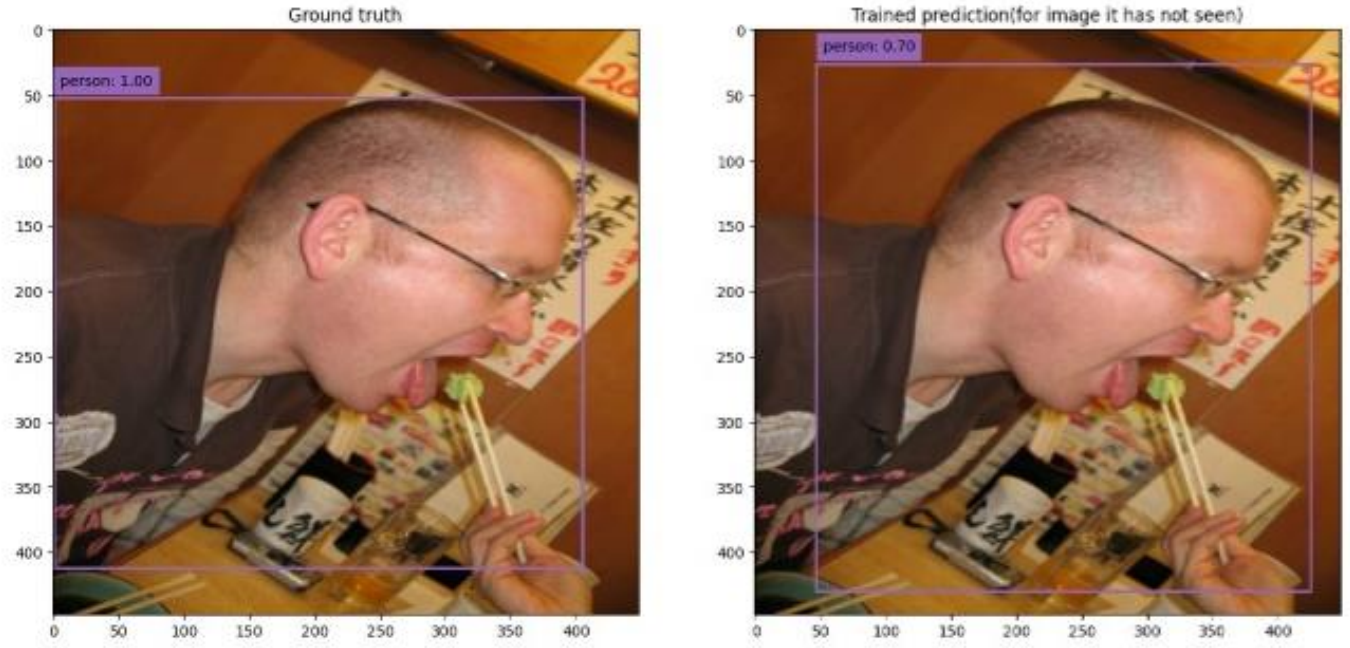
```
        bboxes = cellboxes_to_boxes(trained_model(x))
        bboxes = non_max_suppression(bboxes[i], iou_threshold=0.5, threshold=0.3, box_format="midpoint")
        plot_image(x[i].permute(1,2,0).to("cpu"), bboxes, ax4)
    pred = trained_model(x)
    loss = loss_fn(pred, y)
    ax4.text(10, 30, 'Yolo Loss: {:.5f}'.format(loss.item()), fontsize=20, color='white')
    break
```





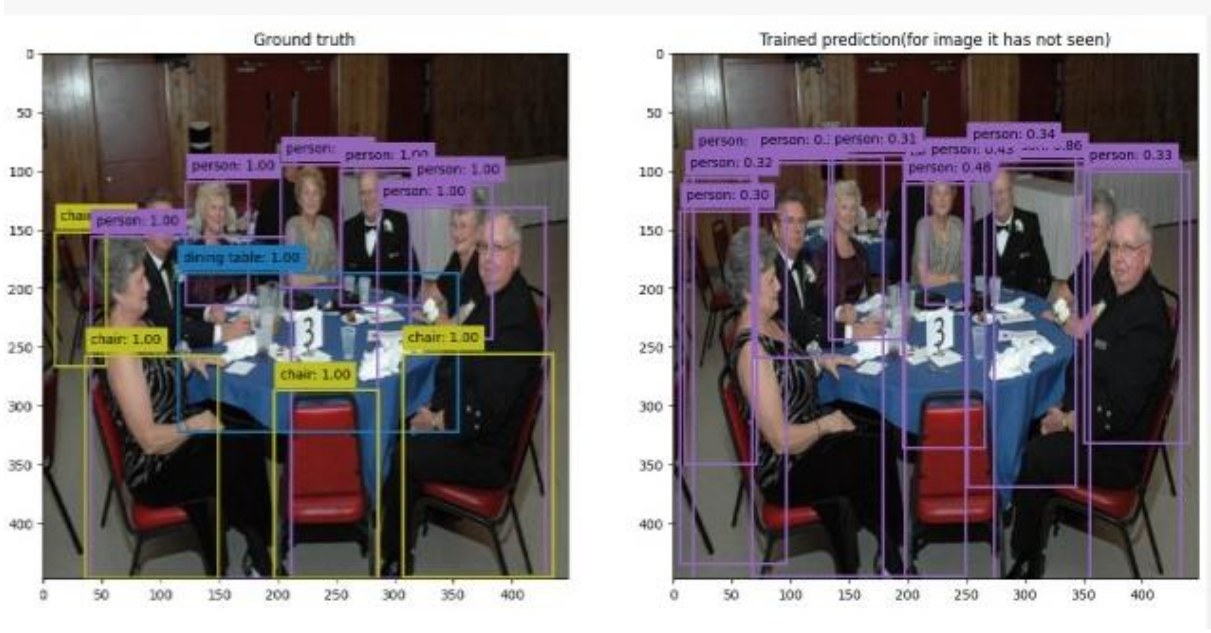
CONT


een)
Ground truth
Trained prediction(for image it has not seen)
car: 1.00
car: 0.97
0
50
100
150
200
250
300
350
400
0
100
200
300
400


Ground truth
Trained prediction(for image it has not seen)
person: 1.00
person: 0.70
0
50
100
150
200
250
300
350
400

CONCLUSION

- YOLO v1 shows enormous speed improvement with batch normalization.
- However, It still struggles with small objects that appear in groups, such as animals or crowds and it also imposes strong spatial constraints on bounding box predictions since each grid cell can only have one class.
- YOLO V2 is updated version of YOLO V1 designed for faster detection using a series of optimizations. It includes features like passthrough layers to handle smaller objects, high-resolution classifier, anchor boxes to predict bounding boxes.



THANK YOU