



Real-time object detection with YOLO and it's Application

Geo Thomas

CET Trivandrum

Contents

1.Introduction

2.YOLO?

3.How YOLO detect real-time object

4.YOLOv1 , YOLOv2, YOLOv3

5.Implementation of YOLO with OpenCV

6.Performance chart YOLO vs other model

7.Real scenario with YOLO

8.Conclusion

Introduction

- Object detection is a computer technology related to computer vision and image processing.
- It is widely used in computer vision tasks such as activity recognition, face detection, face recognition, video object co-segmentation. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, or tracking a person in a video.
- Methods for object detection generally fall into either machine learning-based approaches or deep learning-based approaches.
- YOLO is mainly come under deep learning-based approach.



YOLO?

- You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev
- family of models are a series of end-to-end deep learning models designed for fast object detection
- There are three main variations of the approach till , they are YOLOv1, YOLOv2, and YOLOv3



How YOLO Work's?

- Prior detection systems repurpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered detections.
- YOLO use totally different approach.
- Apply a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.
- It looks at the whole image at test time so its predictions are informed by global context in the image.

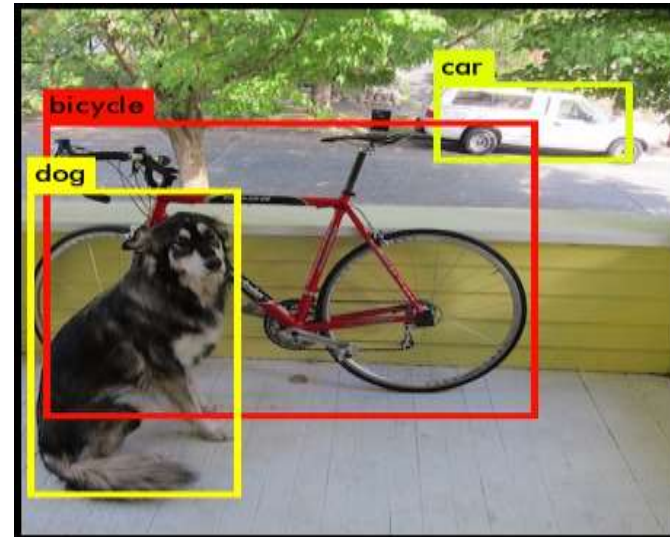
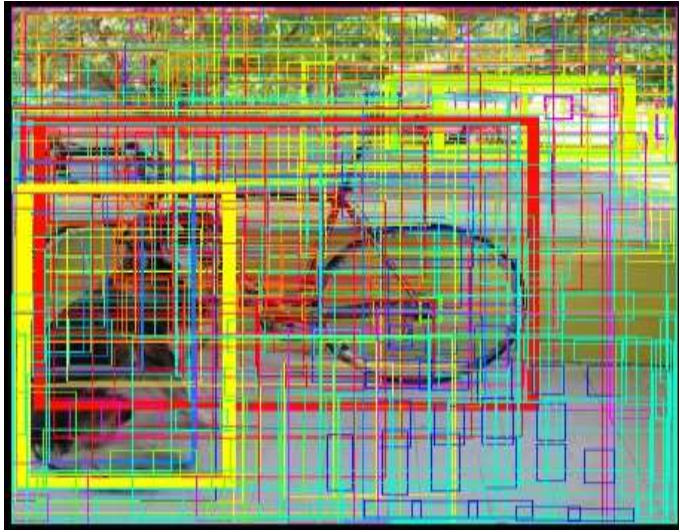
- YOLO use Darknet network.
- The YOLO network splits the input image into a grid of $S \times S$ cells.
- Each grid cell predicts B number of bounding boxes and their objectness score along with their class predictions.
- Coordinates of B bounding boxes -YOLO predicts 4 coordinates for each bounding box (bx,by,bw,bh) with respect to the corresponding grid cell.



- Objectness score (P_0) – indicates the probability that the cell contains an object
- Class prediction – if the bounding box contains an object, the network predicts the probability of K number of classes.
- The predicted bounding boxes may look something like as follow



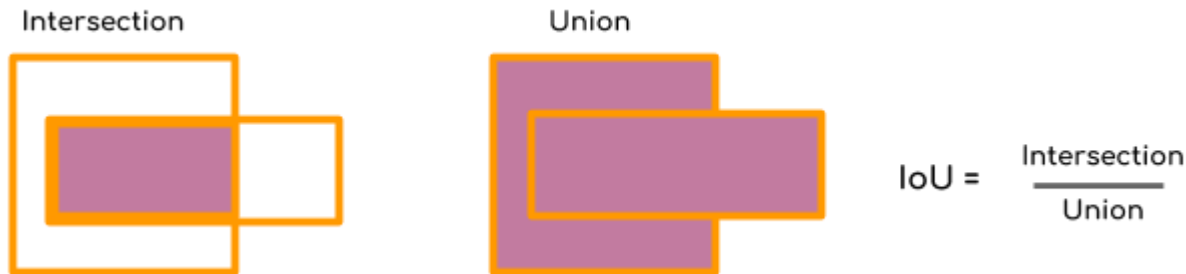
- Finally, the confidence score(Objectness score) for the bounding box and the class prediction are combined into one final score that shows probability that this bounding box contains a specific type of object.




- It turns out that most of these boxes will have very low confidence scores, so only keep the boxes whose final score is above some threshold.
- Non-maximum Suppression (NMS) intends to cure the problem of multiple detections of the same image.

Non-maximum Suppression

- Non-maximum Suppression or NMS uses the very important function called “Intersection over Union”, or IoU



- Define a box using its two corners (upper left and lower right): (x1, y1, x2, y2) rather than the midpoint and height/width.
- Find the coordinates (xi1, yi1, xi2, yi2) of the intersection of two boxes where :
 - xi1 = maximum of the x1 coordinates of the two boxes
 - yi1 = maximum of the y1 coordinates of the two boxes
 - xi2 = minimum of the x2 coordinates of the two boxes
 - yi2 = minimum of the y2 coordinates of the two boxes

- 
- The area of intersection by this formula
 - $\text{area_intersection} = (x_2 - x_1) * (y_2 - y_1)$
 - calculate the area of union
 - $\text{union_area} = (\text{area of box 1} + \text{area of box 2}) - \text{area_intersection}$
 - Therefore $\text{IoU} = \text{area_intersection} / \text{union_area}$
 - Implement non-max suppression, the steps are
 - Select the box that has the highest score.
 - Compute its overlap with all other boxes, and remove boxes that overlap it more than a certain threshold which we call `iou_threshold`.
 - Go back to above two step and iterate until there are no more boxes with a lower score than the currently selected box

YOLOv1

YOLOv2

YOLOv3

It uses Darknet framework which is trained on ImageNet-1000 dataset, 30 FPS

Second version YOLO is named as YOLO9000, 40 FPS

The previous version has been improved for an incremental improvement which is now called YOLO v3, 45 FPS

It could not find small objects if they are appeared as a cluster

Higher Resolution Classifier,
Fine-Grained Features, Multi-Scale Training

Feature Pyramid Networks (FPN)

This architecture found difficulty in generalisation of objects if the image is of other dimensions different from the trained image

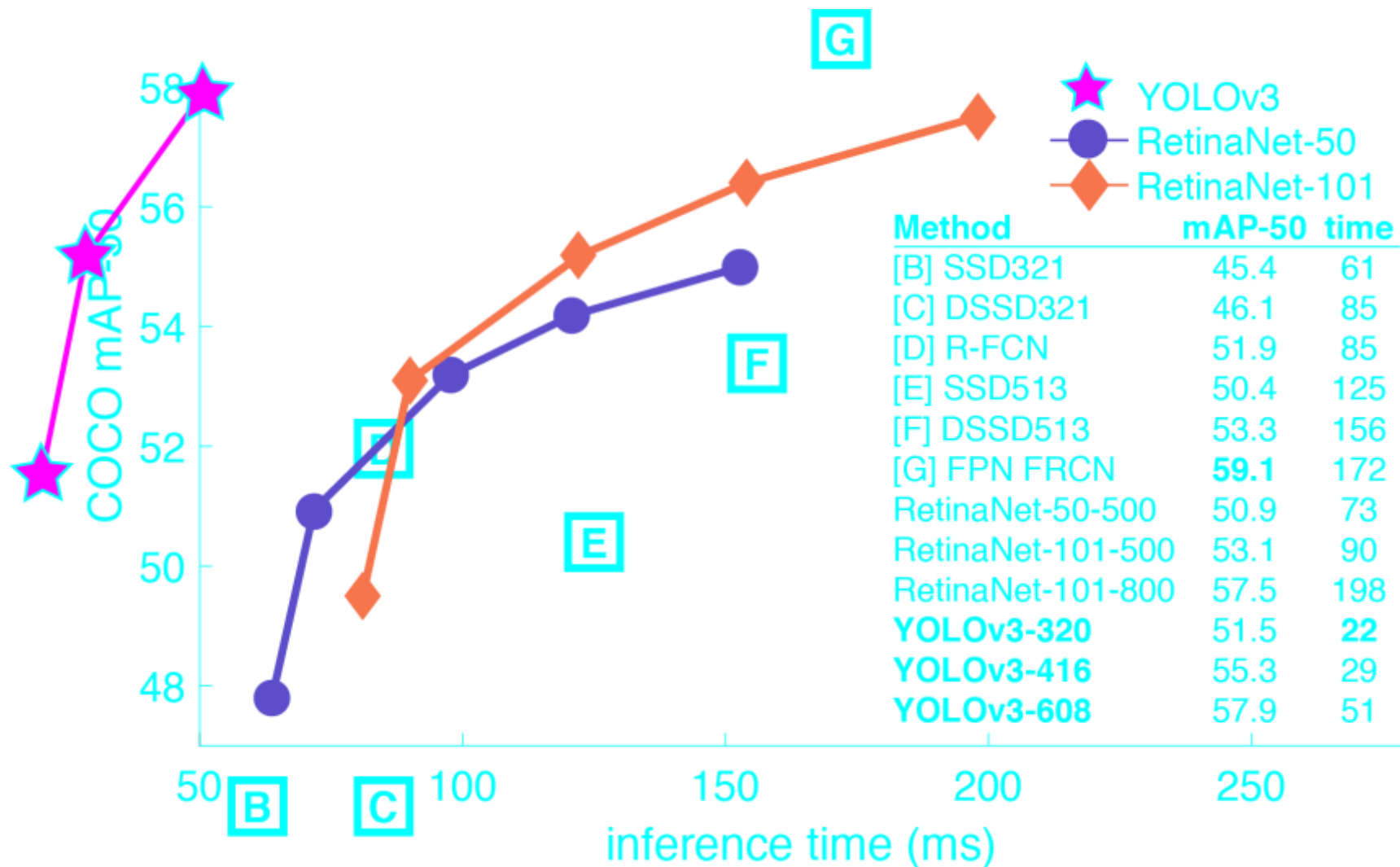
Darknet 19: YOLO v2 uses Darknet 19 architecture

Darknet-53: the predecessor YOLO v2 used Darknet-19 as feature extractor and YOLO v3 uses the Darknet-53 network for feature extractor which has 53 convolutional layers

Basic Implementation of YOLO with OpenCV

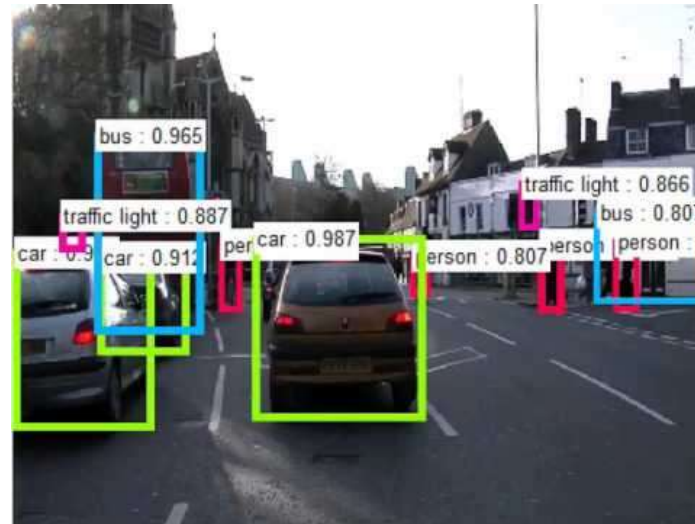
- use OpenCV to implement YOLO algorithm as it is really simple
- Step 1 — Install the dependencies for Windows,Linux
 - Python3,Opencv
- Step 2 — Install the DarkNet/YOLA
 - DarkNet: Originally, YOLO algorithm is implemented in DarkNet framework.
 - The CFG and WEIGHTS files and COCO.
- Step 3 – the command or code need to render the input image or video

Performance chart YOLO vs other model



Real scenario with YOLO

- The core of the self-driving car's brain is YOLO Object Detection.
- The Reason why YOLO Used in Self Driving cars:
 - --->Extremely Fast
 - --->Contextually Aware
 - --->A Generalized Network





Conclusion

- YOLO, a unified model for object detection, is simple to construct and can be trained directly on full images. YOLOv3 is orders of magnitude faster (45 frames per second) than other object detection algorithms.
- YOLO is the fastest general-purpose object detector and YOLO pushes the state-of-the-art in real-time object detection. YOLO also generalizes well to new domains making it ideal for applications that rely on fast, robust object detection.

References

- <https://ieeexplore.ieee.org/document/7780460>
- <https://ieeexplore.ieee.org/abstract/document/8740604>
- <https://ieeexplore.ieee.org/document/8621865>
- <https://medium.com/@venkatakrishna.jonnalagadda/object-detection-yolo-v1-v2-v3-c3d5eca2312a>
- <https://towardsdatascience.com/real-time-object-detection-with-yolo-9dc039a2596b>
- <https://pjreddie.com/darknet/yolo/>