

CALIFORNIA STATE UNIVERSITY SAN MARCOS

PROJECT SIGNATURE PAGE

PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

PROJECT TITLE: Performance and Accuracy Analysis in Object Detection

AUTHOR: Guei-Sian Peng

DATE OF SUCCESSFUL DEFENSE: 11/26/2019

THE PROJECT HAS BEEN ACCEPTED BY THE PROJECT COMMITTEE IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
SCIENCE IN COMPUTER SCIENCE.

Dr. Nahid Ebrahimi Majd
PROJECT COMMITTEE CHAIR


SIGNATURE

11/26/2019
DATE

Dr. Xin Ye
PROJECT COMMITTEE MEMBER


SIGNATURE

11/26/2019
DATE

Performance and Accuracy Analysis in Object Detection

Guei-Sian Peng

California State University at San Marcos, Computer Science Department

Advisor: Dr. Nahid Majd

Date of Delivery: Nov 26, 2019

Abstract

We analyze the efficiency of the state-of-the-art, object detection systems that have recently been introduced for fast and accurate object detection in images, video streams and real-time videos. We implemented and analyzed the efficiency of YOLOv2, YOLOv3 and SSD object detection systems. In this project, we introduce the functionality of these 3 systems, the metrics to evaluate the efficiency of object detection algorithms and present the results of our implementations for small-scale datasets. We also present an efficiency analysis of these three systems for large-scale datasets.

Object detection is an intelligent computer vision technique, similar to our humans' visions, for locating instances of objects in images, video or real-time surveillance. It has been researched for several years and has been improved to an unprecedented level. It also has been adopted across our daily lives from our cellphones, video surveillance, and object tracking to pedestrian recognition and so forth. There are various detection models such as Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN), You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD), etc.

We studied the most recent and most efficient systems: YOLOv2, YOLOv3, and SSD. YOLOv3 and SSD are considered the best methods with the highest accuracy and fastest speed that can be achieved in object detection in images and video streams. We also studied the efficiency of all three systems for real-time object detections.

The results demonstrated that YOLOv3 is the most accurate but slowest object detection system while SSD is the fastest one with the lowest accuracy. YOLOv2 has a lower accuracy than YOLOv3 but it is faster. For object detection in recorded images and videos, YOLOv3 is the best one since it detects the objects with the highest accuracy, however for real-time video-streams, SSD provides the best one since it is the fastest one. Since there is a trade-off between accuracy and speed in all these systems, the most appropriate system for each application depends on the application requirements.

Acknowledgements

I would like to express my thanks of gratitude to my Advisor Dr. Nahid Majd for all the instructions, assistance, and supports to my graduate project. Especially, when doing this project, there were some difficulties I encountered, and she truly helped me to solve those issues. I also deeply appreciate innumerable hours she spent on our meeting, her rapid response to my email, and her patience with my questions and problems. I will always remember the words she said to me after each meeting, "you did a great job", or "you have great progress" that encouraged me to believe myself that I can finish this project. Thank her so much for everything she has done for me. Moreover, I would like to thank my colleagues Dr. Lee Dajung to help me clarify the confusion I faced at the beginning of educating myself object detection and encouraged me and built confidence for me.

Furthermore, I would like to express my sincere thanks to my wife, Joanne. Without your support and care, I would not have gone so far. Thank her for never complaining about many unnecessary difficulties in our lives she had been through and tolerated. Finally, I am grateful to my parents for all their unconditional support and love during my studies abroad. Thank them for taking good care of themselves so that I can focus on my studies without any worries.

I. Table of Contents

I.	Table of Contents	3
II.	Table of Figures	4
III.	Table of Tables	4
IV.	List of Abbreviations and Definitions.....	6
1.	Introduction.....	7
1.1	Difference between classification and object Detection	8
1.2	Difference between one stage and two stage object detectors	8
1.3	The brief introduction of the YOLO and SSD	9
1.3.1	YOLO	9
1.3.3	Major distinction between YOLO and SSD.....	9
2.	Description of Algorithms and Dataset.....	11
2.1	YOLO detector:.....	11
2.2	SSD Algorithm:	14
2.3	COCO dataset	15
3.	Implementation	16
3.1	One image output:.....	16
3.2	Video outputs:	19
3.3	Real time output:.....	20
4.	Evaluation approach.....	22
4.1	A fundamental concept of mAP for an object detector's accuracy evaluation	22
4.2	The standard evaluation of speed performance criterion for object detectors	27
4.3	Performance Discussion.....	27
5.	Conclusion and future work.....	29
6.	References.....	30

II. Table of Figures

Figure 1. Algorithms for Object detection [10].	7
Figure 2. The concept of the classification, Localization, and object detection [14].....	8
Figure 3. The fundamental architecture of YOLO with 24 convolutional layers followed by 2 fully connected layers [1].	11
Figure 4. An example of a bounding box with its output elements	12
Figure 5. An example of normalization [17].....	12
Figure 6. Bounding boxes with dimension priors and location prediction. The color blue rectangle box is the predicted boundary box and the color black dotted rectangle is the anchor [2].....	13
Figure 7. SSD network architecture SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences [4]......	15
Figure 8. An example of detecting a shopping cart in an image. The predicted bounding box is drawn in red while the ground-truth bounding box is drawn in green. Our goal is to compute the Intersection of Union between these bounding boxes.....	22
Figure 9. An example of PR curve [17].	23
Figure 10. A traditional way to interpolate base on the 11 equally space levels [17].....	24
Figure 11. The new approach for interpolation [17].	24

III. Table of Tables

Table 1. The comparison of speed and accuracy among YOLO versions.	9
Table 2. Darknet-53 [3].....	14
Table 3. One image outputs of the implementation.	19
Table 4. Video outputs of the implementation.....	20
Table 5. Real-time outputs of the implementation.....	21
Table 6. The YOLOv2 real-time output of the implementation.	21
Table 7. Four situations of two conditions combination.....	23
Table 8.The mAP, and mAR of the YOLOv3 (416x416) [21].	26
Table 9. The mAP, and mAR of the SSD (300x300) [21].	27
Table 10. The mAP comparison between YOLOv3 and SSD [21].	27
Table 11. The FPS comparison for YOLOv3 and SSD [4].....	27
Table 12. The comparison of one image testing between YOLOv3 and SSD.....	28
Table 13. The comparison of video testing between YOLOv3 and SSD	28
Table 14. The comparison of real-time testing between YOLOv3, SSD, and YOLOv2	28

IV. List of Abbreviations and Definitions

Object classification: differentiating an object in an image

Object localization: identifying the location of the, and then drawing a bounding box around it.

Object detection: the combination of object classification and localization.

CNN: convolutional neural network

YOLO: you only look once. A new object detection approach, which uses features learned by a deep convolutional network to detect objects in an image.

SSD: single shout mutibox detector

$IOU_{predict}^{truth}$: intersection over Union

TP: true positive

TN: true negative

FP: false positive

FN: false negative

PR curve: Precision- recall curve

mAP: mean average precision

mAR: mean average recall

FPS: frame per second

COCO dataset: Common Objects in Context

OpenCV: open computer vision. An open-source library of functions that allow for real-time computer vision.

1. Introduction

Object detection, detecting instances of semantic objects, is a computer vision and image processing technology [6]. A decade ago, distinction any object such as a person, animal, building, car, human face and so on, via computer was considered an unattainable task, especially like the difference between a cat and a dog even with a significant advance in the state of artificial intelligence [7].

However, in 2012, Alex Krizhevsky et al. designed a convolutional neural network model called AlexNet, which significantly improved the performances in labeling pictures (classification) in the ImageNet challenge. and outperformed all previous models. Because of this breakthrough in the convolution neural network, many other CNN models were springing up all over this field such as VGGNet, inceptionNet, and ResNet and so forth. Nonetheless, the convolution neural network has been studied since the 90s. Why had it taken so long to reach this turning point? In addition to the contribution to the development of CNN on classification from Alex Krizhevsky, there were also two catalysts leading CNN to this flourish, which were first, GPUs – giving lots of computing power and second, large dataset – increasing accuracy via training.

Although these approaches are impressive, image classification is too simple than the human's complex visual understanding because, in reality, a complicated scene we view are objects and different backgrounds overlapping each other. Except for classifying these objects, we need to identify their locations, boundaries, and differences, or relations. That is the reason why the term, object detection, come out in several years later from 2012.

Recently years, object detection has been employed ubiquitously from the use of video surveillance, tracking object, and pedestrian recognition, just to name a few. There are multiple deep learning approaches for computer vision systems such as Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN), Single Shot MultiBox Detector (SSD), You Only Look Once (YOLO) and so on as the figure 1 shown below. Joseph Redmon, the designer of the YOLO algorithm, claimed that the detectors nowadays are able to detect objects with a highly accurate detection rate, at a level greater than 99 percent accuracy [7].

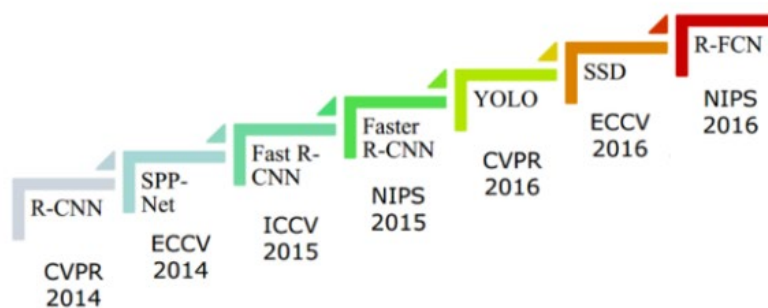


Figure 1. Algorithms for Object detection [10].

In this project, we are going to explain how to evaluate the performances for an object detector and analyze both the YOLO model (YOLOv2, and YOLOv3) and the SSD model in accuracy and speed. In addition, I will implement three object detectors in CPU (central processing unit) with python language in PyCharm,

an integrated development environment (IDE) design particularly for python by using the given pre-trained weight and configuration of the YOLOv2, YOLOv3, and SSD convolution neural network algorithms.

1.1 Difference between classification and object detection

In the past decade, image processing has significantly progressed, from 2012, a turning point of the CNN for classification, to 2015, another turning point of object detection thanks to the appearance of first created object detector, R-CNN, up to the present. Before we dive into object detections, we first define the differences between classification, localization, and object detection.

Classification is to predict the labels of objects in images while detection involves not only classification but also the localization of those objects in an image. In classification, for example, it is assumed that an object takes a remarkable portion of an image like the image on the far left in figure 2, and we predict and classify it as a cat. If we, like the image shown in the middle of figure 2, find a bounding box around that object, it is known as localization.

The combination of classification and localization along with the multiple objects in an image as the image shown on the far right of figure 2 contains multiple objects – cat, dog, and duck presented in different locations with different scales and sizes. We then try to find all the objects, predict their labels and draw a bounding box around them, which is called object detection.

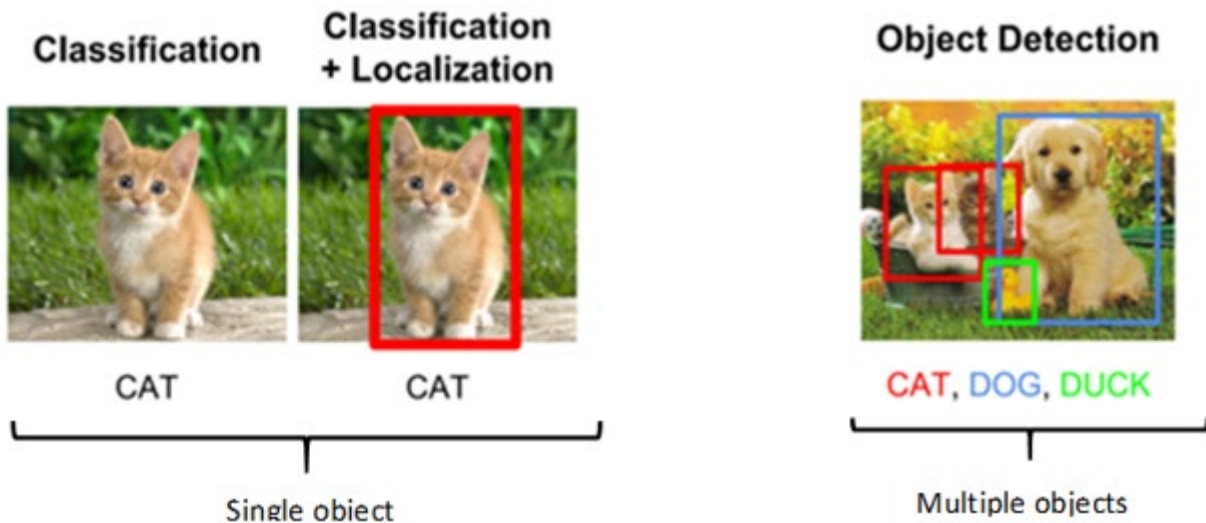


Figure 2. The concept of the classification, Localization, and object detection [14].

1.2 Difference between one stage and two stage object detectors

As for object detection tasks, we, in general, can separate them into two different groups (one stage or two stage) – either leverage a regional proposal network to search objects, and then process these interested region candidates and output a final prediction via the second network like R-CNN (two stage), or make a fixed number of predictions on grid like YOLO or SSD based on regression (one stage). The big problem with the R-CNN family of network is its speed, which is very slow compared to one stage approaches, although it seems to be more accurate.

“The YOLO algorithm improves the image processing a thousand times faster from 20 seconds per image to 20 milliseconds per image [7].” Therefore, object detection, detecting instances of objects from a particular class in an image according to the definition, has become more popular in just a few years, one stage approaches in particular.

Therefore, we will focus on the YOLO and SSD vision system since the R-CNN approach takes too much time to do the computing in spite of its higher accuracy. A fast and accurate computing algorithm for object detection is extremely crucial so that it would allow the computer to perform complex tasks, like the human visual system, close to real-time such as driving a car without any extra sensors but only with a camera [1].

1.3 The brief introduction of the YOLO and SSD

1.3.1 YOLO

The first version of the YOLO algorithm to detect objects was published in 2016, named YOLOv1 [1]. Its main difference from other conventional visual system is that instead of working on repurposes classifiers to perform detection, Joseph frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities, which are predicted directly from full images in one evaluation via a single neural network, which is able to be optimized end-to-end directly on the detection performance [1]. In 2017, next year after first publish, YOLO9000(YOLOv2) was published with various improvements, titled “YOLO9000: Better, Fast, Stronger”, due to the changes of some features like higher resolution classifier, anchor boxes, Darknet 19 convolutional layers architecture, and multiple uses of dataset for training like PASCAL VOC and COCO and so on [2]. It is faster and more precise. The latest version published in 2018 called YOLOv3 trained on the COCO dataset containing 80 labels, which will be used in our project, with a little bigger network but still fast enough and very accurate compared to YOLO9000 [3]. There are differences among YOLOv1, YOLOv2, and YOLOv3, for example, YOLOv3, instead of using the softmax approach from the last version, uses logistic classifiers for each class and gives the score to the objects for every bounding box [11]. Moreover, the comparison of speed and accuracy of three versions are shown in table 1.

Model	Speed	Accuracy
YOLOv1	low	low
YOLOv2 (YOLO9000)	fastest	median
YOLOv3	median	highest

Table 1. The comparison of speed and accuracy among YOLO versions.

1.3.2 SSD

Another one stage approach, the first Single shot multiBox Detector (SSD), was published by C. Szegedy et al. in late 2016. SSD uses a VGG16 as the base network due to its high-quality image classification, and then convolutional feature layers that gradually reduce in size are added to its end so that it is able to predict object at different scales by the aspect ratio. The core of SSD, by applying small convolutional filters to feature maps, is to predict class scores and offsets for a fixed set of the default bounding box. However, this model has been improved and implemented by many researchers. For example, instead of taking VGG16 as a feature extractor or object classifier for SSD, ResNet, or MobileNet are utilized by researchers.

1.3.3 Major distinction between YOLO and SSD

The YOLO model predicts the confidence score for an object and predicts the probability of each class given that there is an object existing next, whereas the SSD network does not predict the confidence score for each object, but it gives the probability that a class is present in a given bounding box [15]. Moreover, the YOLO model adopts an intermediate fully connected layer while the SSD replaces it with a convolutional filter [4].

2. Description of Algorithms and Dataset

2.1 YOLO detector:

You only look once (YOLO), a single convolutional neural network different from prior detectors, frames object as a regression problem to spatially separated bounding boxes and associated class probabilities directly from full images in one evaluation [1]. The full network YOLOv1 architecture with 24 convolutional layers and 2 fully connected layers is shown below in figure 3. In our project, we are using YOLOv3, rather than using YOLOv1, as our model to detect objects.

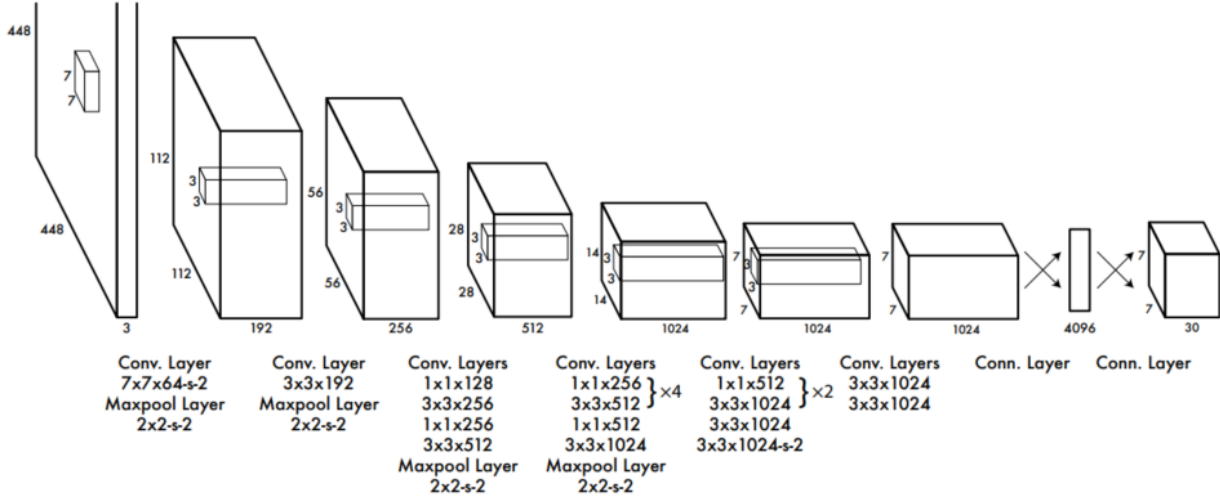
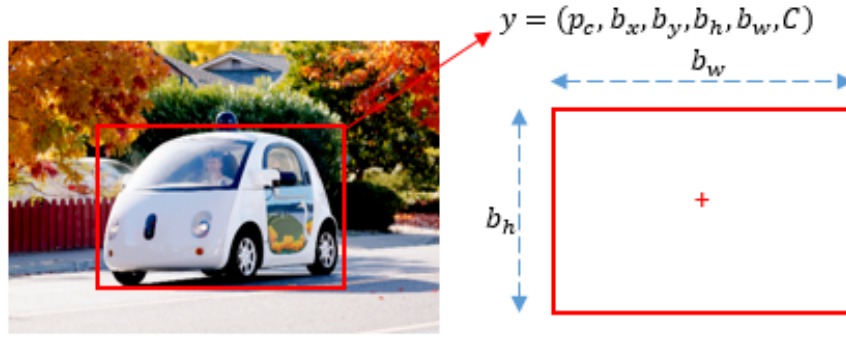


Figure 3. The fundamental architecture of YOLOv1 with 24 convolutional layers followed by 2 fully connected layers [1].

How YOLO works:

YOLO algorithm divides any given image into the $S \times S$ grid. Each grid cell on the input image predicts a fixed number of boundary boxes (anchor boxes) for an object. As for each boundary box the network outputs offset 4 element values (b_x, b_y, b_h, b_w), one confidence p_c , and C conditional class probabilities. The coordinates (b_x, b_y) represent the bounding box's center relative to the bounds of the grid cell in the input image. The b_w and b_h are box's width, and height respectively. The confidence p_c is the probability that a box contains an object and how accurate is the boundary box, which is equivalent to $\Pr(object) * IoU_{predict}^{truth}$. C conditional class probabilities, $\Pr(class_i | object)$, are the probabilities the objects belong to $class_i$ given an object is present. An example of a bounding box with its output elements is illustrated in figure 4.



p_c : confidence of an object being present in the bounding box

b_x : the offset of the x coordinate of the predicted bounding box's center

b_y : the offset of the y coordinate of the predicted bounding box's center

b_h : the offset of the bounding box's height that contains an object

b_w : the offset of the bounding box's width that contains an object

C : class of the object being detected (e.g 1.person,2.bick,3.car...)

Figure 4. An example of a bounding box with its output elements

Normalization: The bounding box's width and height are normalized by the image's width and height, so the value of the wide and height falls between 0 and 1. Similarly, the bounding box's x and y coordinates are offset with a particular grid cell location, so they are also bounded between 0 and 1. An example is displayed in figure 5 below [1].

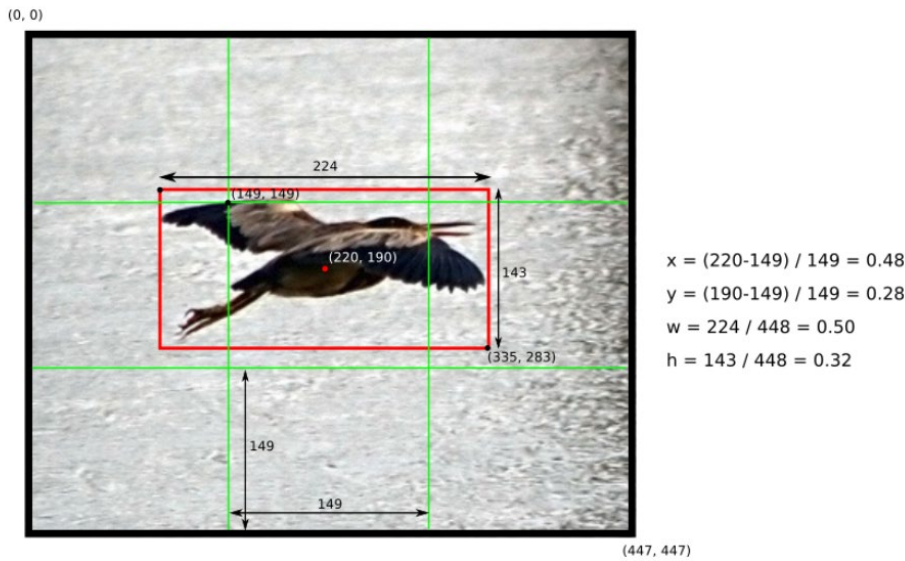


Figure 5. An example of normalization [17].

YOLOv3, using the sigmoid function, predicts the coordinates of the box's center relative to the location of filter application, and the width and height of the box as offsets from cluster centroids as the figure 6 showed below.

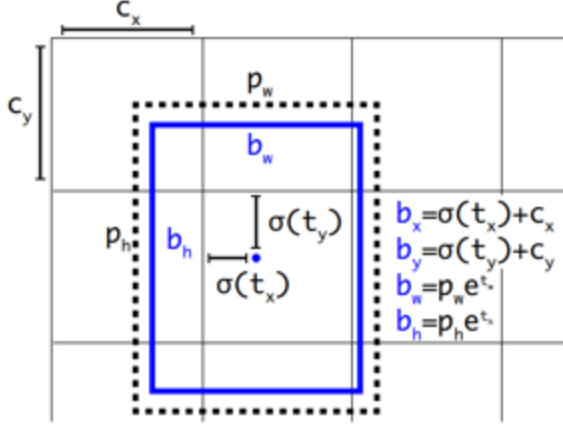


Figure 6. Bounding boxes with dimension priors and location prediction. The color blue rectangle box is the predicted boundary box and the color black dotted rectangle is the anchor [2].

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

$$Pr(object) * IoU(b, object) = \sigma(t_0)$$

Where

t_x, t_y, t_w, t_h are predictions made by YOLO.

c_x, c_y is the top left corner of the grid cell of the anchor.

p_w, p_h are the width and height of the anchor.

c_x, c_y, p_w, p_h are normalized by the image width and height.

b_x, b_y, b_w, b_h are the predicted boundary box.

$\sigma(t_0)$ is the box confidence score.

YOLOv3 has 3 different scales, at each scale predict 3 anchor boxes. In our case, with COCO dataset, the tensors at each scale is $N \times N \times [3 \times (4 + 1 + 80)]$ for the 4 bounding box offsets, 1 objectness and 80 class predictions [1].

YOLOv3 applies the k-means cluster to determine the priors (anchors). There are pre-select 9 clusters whose width and height are (10×13), (16×30), (33×23), (30×61), (62×45), (59×119), (116×90), (156×198), (373×326) that are split up evenly across three scales. Each group is assigned to a specific feature map above in detecting objects.

In addition, YOLOv3 uses a new network called Darknet-53 for performing feature extraction as table 2 shown below. The network uses successive 3×3 and 1×1 convolutional layer. Nonetheless, it has some shortcut connections now, but relatively larger with 53 convolutional layers compared to the YOLO 9000 with 19 layers, and YOLO v1 with 24 layers. This new network works more accurate on extracting features than Darknet-19 adopted in YOLOv2 or other networks like ResNet-101 or ResNet-152 [3].

	Type	Filters	Size	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
1x	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3 / 2	64 x 64
2x	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3 / 2	32 x 32
8x	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	16 x 16
8x	Convolutional	256	1 x 1	
	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	8 x 8
4x	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 2. Darknet-53 [3]

2.2 SSD detector:

Single Shot MultiBox Detector (SSD), a single convolutional neural network, is not complex compared with other methods providing object proposals that are removed by SSD. The SSD architecture in figure 7 is separated into two parts, a base network contributing high quality image classification applied to the front (SSD takes the VGG16 network as the base network to extract feature maps.), and several convolutional feature layers added afterward to predict objects detection. There are several features in the SSD model [4]:

- Multi-scale feature maps for detection: As for the rear network. The sizes of convolutional feature layers added decrease gradually, which allows us to predict objects at different scales. Besides, each feature layer uses a different convolutional model to predict detections.
- Convolutional predictors for detection: Each added feature layer or any existing feature layer from the base network can generate a fixed set of detection predictions using a set of convolutional filters that are displayed on the top of SSD architecture in Figure 7. For a feature layer with the size $m \times n$ and p channel, SSD applies small convolution filters that are 3×3 to calculate the location and class scores for each cell to make predictions for a fixed set of the default bounding box. Each

contains its own boundary with offset shape to its default box and scores for all classes plus one for no object present (a class 0 is reserved for no object). The YOLO model, instead of using a convolutional filter, adopts an intermediate fully connected layer that is discarded by SSD.

- **Default boxes and aspect ratios:** A set of default bounding boxes with each feature map cell are associated together in terms of multiple feature maps at the front network. The position of each default boxes relative to its corresponding cell is fixed. At each grid cell in feature map, the offsets relative to the default box shapes in the cell, and also the per-class scores that indicate the presence of a class instance in each of those boxes are predicted. To be Specific, for each box out of k at a given location, c class scores and the 4 offsets are calculated relative to the original default box shape. These results in a total of $(c + 4)k$ filters that are applied around each location in the feature map, yielding $(c + 4)kmn$ outputs for a $m \times n$ feature map.”

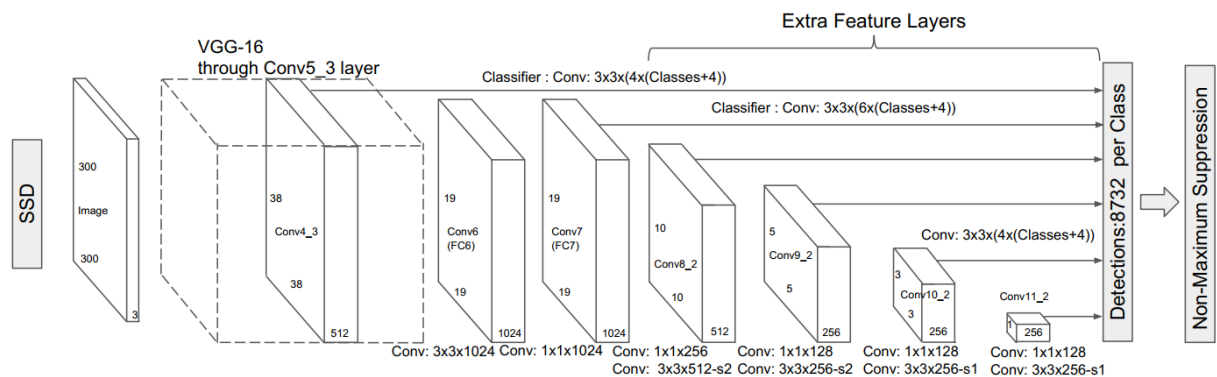


Figure 7. SSD network architecture SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences [4].

2.3 COCO dataset

Common Object in Context (COCO) is a large-scale image dataset created for object detection, segmentation, person key-points detection, stuff, and caption. Its dataset contains 330K images, 1.5 million object instances along with greater than 200K labeled. All of the objects are classified into 80 categories. It also provides Matlab, Python, and Lua APIs, which support programmer to load, parse, and visualize the annotations of all images. More information like data, paper, annotation format, are described in the COCO website. Taking the annotation for object detection as an example, the format looks like below [19]:

```
Annotation {
    "id"           : int,
    "image_id"     : int,
    "category_id"  : int,
    "segmentation" : RLE or [polygon],
    "area"         : float,
    "bbox"         : [x, y, width, height],
    "iscrowd"      : 0 or 1,
}

Categories [ {
    "id"           : int,
    "name"         : str,
    "supercategory" : str,
} ]
```

3. Implementation

We are going to implement one image input detection, a video input detection, for YOLOv3, and SSD and real-time detection for YOLOv2, and YOLOv3, and SSD. Before we start to implement, we have to download the pre-trained model file and model configuration file first.

- YOLOv2: YOLOv2.weight and YOLOv2.cfg.
- YOLOv3: YOLOv3.weight and YOLOv3.cfg.
- SSD: SSD_deploy.caffemodel and SSD_deploy.prototxt.

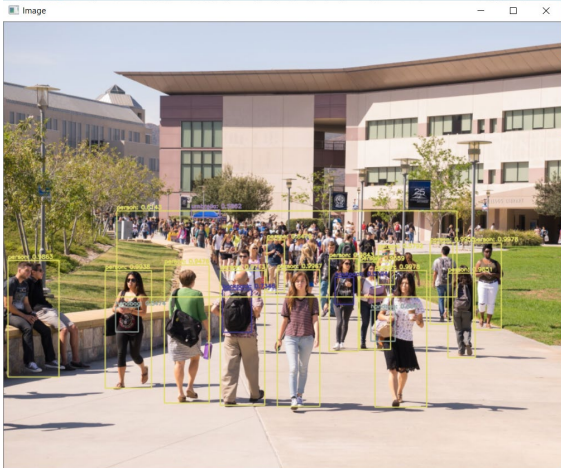



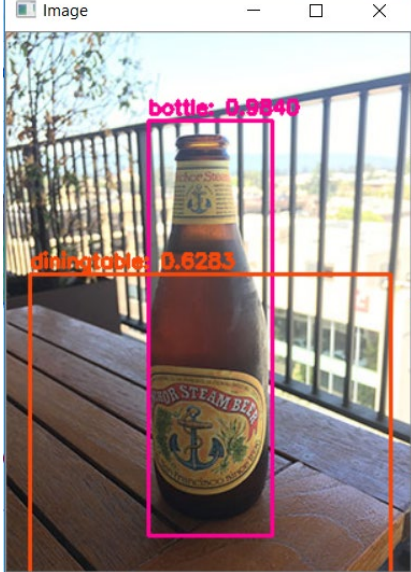

Besides, we have to install PyCharm as our IDE (integrated development environment) and import some libraries as below:

- Python 3.7
- Numpy
- Imutils
- OpenCV

In the tables below, we displayed the outputs of YOLOv3 and SSD model application, we marked the differences between two models with circles in color red in the table 3 of one image output in section 3.1, and in the table 4 of video output in section 3.2 because it is not possible to maintain the same situation and circumstance for real-time testing, whose output was presented in table 5. Also, we downloaded a pre-trained model of the YOLOv2 and implemented it in real-time. Our programming implementation was able to complete thanks to the author Adrian Rosebrock, Ph.D. providing various examples of explaining how to play with images processing [20].

3.1 One image output:

Input: image	YOLOv3	SSD
1		

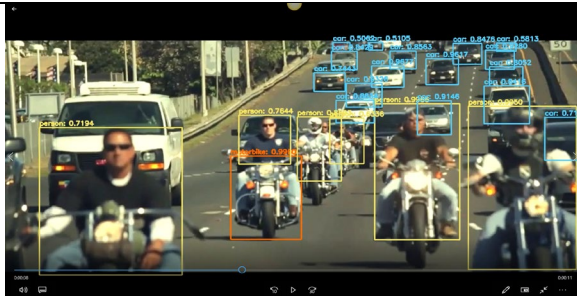

Input: image	YOLOv3	SSD
2	 A screenshot of the YOLOv3 'Image' window showing a crowd of people walking on a paved path in front of a large building. Multiple yellow bounding boxes are drawn around individual people, indicating object detection.	 A screenshot of the SSD 'Output' window showing the same crowd scene. A red circle is drawn around a group of people in the background, highlighting a specific area of interest.
3	 A screenshot of the YOLOv3 'Image' window showing a horse and rider jumping over a hurdle. Several bounding boxes are present: a green box for the horse with label 'horse: 0.9956', a yellow box for the rider with label 'person: 0.9918', and a blue box for the hurdle with label 'can: 0.5283'. Other smaller boxes are visible for the background.	 A screenshot of the SSD 'Output' window showing the same horse and rider scene. A red circle is drawn around the horse and rider, highlighting the main subject. Bounding boxes and confidence scores are visible: 'horse: 99.90%' and 'person: 56.05%'.
4	 A screenshot of the YOLOv3 'Image' window showing a beer bottle on a wooden bench. Two bounding boxes are present: a pink box for the bottle with label 'bottle: 0.9840' and an orange box for the bench with label 'dining table: 0.6283'.	 A screenshot of the SSD 'Output' window showing the same beer bottle scene. A red circle is drawn around the bottle, highlighting it. A bounding box and confidence score are visible: 'bottle: 100.00%'.

Input: image	YOLOv3	SSD
5		
6		
7		

Input: image	YOLOv3	SSD
8		
9		
Accuracy	High	Low
Time	0.84 ~ 0.9 (second / per frame)	0.17 ~ 0.23 (second / per frame)
Speed	Slow	Fast

Table 3. One image outputs of the implementation.

3.2 Video outputs:

Input: video	YOLOv3	SSD
Motor_ Bike (at the 8 th second)		

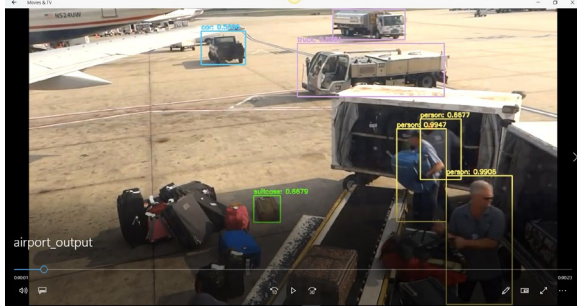





Input: video	YOLOv3	SSD
Airport (at the 1 st second)		
Car_chase_1 (at the 7 th second)		
Car_chase_2 (at the 12 th second)		
Accuracy	High	Low
Time	0.888 ~ 0.9180 sec / per frame	0.1830 ~ 0.237 second / per frame
Speed	Slow	Fast

Table 4. Video outputs of the implementation.

3.3 Real time output:

Due to the reason that it is impossible to maintain exactly the same circumstance for a real-time situation, we skip accuracy comparison here.

Input: real-time	YOLOv3	SSD
Location: Coffee shop		
FPS	1.39	14.72
Speed	Slow	Fast

Table 5. Real-time outputs of the implementation.

In addition to the comparison between YOLO and SSD algorithms, we also tried to download and test the YOLOv2 model, which is a light version of YOLO architecture that has a higher speed with a little bit of tradeoff of accuracy decreasing. The result is in table 6.

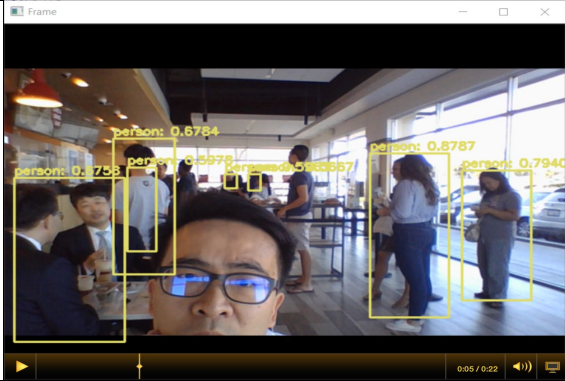
Input: real-time	YOLO v2
Location: Coffee shop	
FPS	9.05

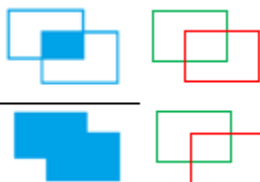
Table 6. The YOLOv2 real-time output of the implementation.

4. Evaluation approach

To evaluate the performance of an object detector. Normally, we focus on the accuracy (mean average precision, mAP) and consecutive images' average process rate of objects detection per second (Frames per second, FPS). In terms of accuracy, there are many different approaches used to evaluate the accuracy of a model or an algorithm for object detection, but mAP is the primary one. In terms of speed, the FPS is the standard one. Before analyzing it, we, at first, have to understand some basic concepts such as confidence score, IoU, precision, recall and so on for accuracy, and the FPS for the speed of performance.

4.1 A fundamental concept of mAP for an object detector's accuracy evaluation

- Confidence score: the reflects the probability that an anchor box contains an object. It is usually predicted by a classifier.
- Ground truth bounding box (B_{gt}): represents the desired output of an algorithm on an input, for example, the hand labeled bounding box from the testing set that specify where the objects are in the image.
- Predicted bounding box (B_p): represents a rectangle region generated from model detector that indicates the location of the object predicted.
- Intersection over union (IoU): an evaluation metric used to measure the area encompassed by both the ground-truth bounding box (B_{gt}) the predicted bounding box (B_p) [18]:

$$\text{IoU} = \frac{\text{Area of Overlap } (B_{gt} \cap B_p)}{\text{Area of Union } (B_{gt} \cup B_p)}$$


The following figure 8 is an example of detection a shopping cart in an image. Based on this image we can have a basic understanding of IoU.

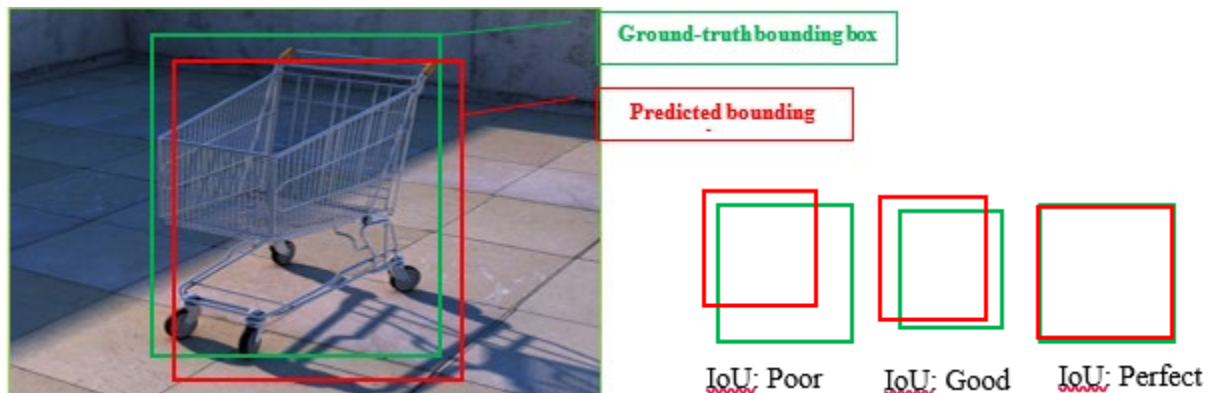


Figure 8. An example of detecting a shopping cart in an image. The predicted bounding box is drawn in red while the ground-truth bounding box is drawn in green. Our goal is to compute the Intersection of Union between these bounding boxes.

- **Threshold:** we predefine a threshold of IoU (for instance, 0.5) in classifying whether the prediction is a true positive or a false positive.
- **True positive (TP):** A true positive test result is one that detects the condition when the condition is present.
- **True Negative (TN):** A true negative test result is one that does not detect the condition when the condition is absent.
- **False positive (FP):** A false positive test result is one that detects the condition when the condition is absent.
- **False Negative (FN):** A false negative test result is one that does not detect the condition when the condition is present.

The following table 7 illustrates these four situations of two conditions combination.

Test	Condition	Present	Absent
Positive		TP	FP
Negative		FN	TN

Table 7. Four situations of two conditions combination.

- **Precision:** the number of true positive divided by the sum of true positive and false positive.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **Recall:** the number of true positives divided by the sum of true positives and false negatives. However, the sum is just the number of ground-truths, so it is not necessary to count the number of false negatives:

$$\text{Recall} = \frac{TP}{TP+FN}$$

Through the threshold value setting for the confidence score at various levels, different pairs of precision and recall are generated with recall on the x-axis and precision on the y-axis, which can be drawn to a precision-recall curve indicating their association. Precision-recall curve (PR), as figure 9, can be utilized to measure the performance of the detector.

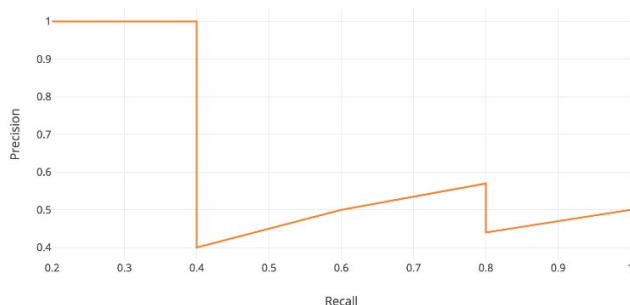


Figure 9. An example of PR curve [17].

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

However, it is not easy to compare the accuracy among different detectors when PR curves intersect with each other. This is why the average precision (AP), based on the precision-recall curve, takes the responsibility for the measurement of object detection accuracy. AP, essentially, is the precision averaged across all unique recall levels [19].

- Average precision (AP): The definition of AP is to computer the area under the precision-recall curve above. The average precision's value always falls between 0 and 1 based on the PR curve since Precision and recall are always between 0 and 1, too.

$$AP = \int_0^1 p(r)dr$$

Before we compute AP, we first interpolate the precision at multiple recall levels in order to reduce the impact of the wiggles in the curve. The interpolated precision P_{interp} at a certain recall level r is defined as the highest precision found for any recall level $r' \geq r$:

$$P_{interp}(r) = \max_{r' \geq r} P(r')$$

The conventional way is to select 11 equally spaced recall levels (i.e., 0.0, 0.1, 0.2, ... 1.0), presented in figure 10.

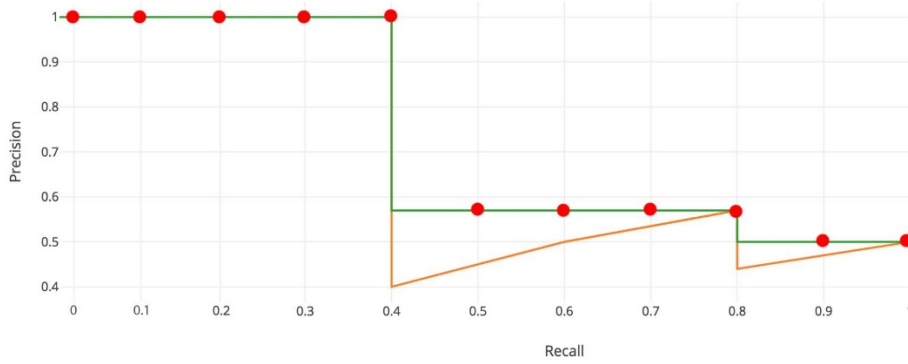


Figure 10. A traditional way to interpolate base on the 11 equally space levels [17]

However, a new approach said to be more capable of improving precision and measuring differences between methods with low AP becomes a standard way used to sample the curve at all unique recall levels (r_1, r_2, \dots) whenever the maximum precision value drops. An example is shown in figure 11.

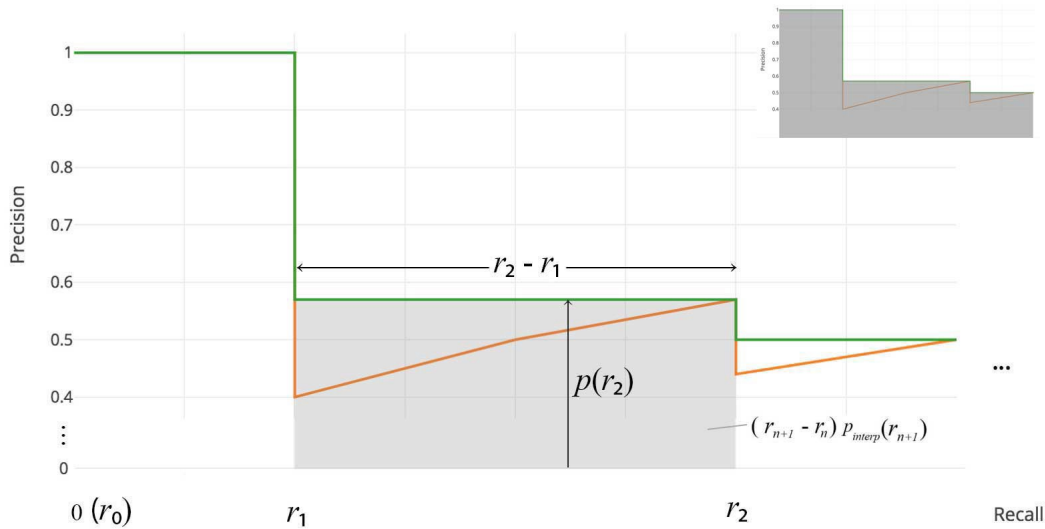


Figure 11. The new approach for interpolation [17].

AP then can be defined as the area under the interpolated PR curve, which can be calculated using the formula as below (The definition is known as area under curve, AUC):

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{interp}(r_{i+1})$$

$$p_{interp}(r_{i+1}) = \max_{r' \geq r_{i+1}} p(r')$$

Where r_1, r_2, \dots, r_n is the recall levels in an increasing order where the precision is first interpolated.

The calculation of AP only involves one class. However, in object detection there are usually $K > 1$ class. Mean average precision (mAP) is defined as the mean of AP across all K classes like below.

- $mAP = \frac{\sum_{i=1}^K AP_i}{K}$
- Average recall (AR): Like AP, computing the average recall value on a per-class basis, can be used to compare detector performance.
- $mAR = \frac{\sum_{i=1}^K AR_i}{K}$

According to the COCO Object Detection where we download the images, there are features and 12 evaluation metrics used for characterizing the performance of an object detector on COCO [19]:

The features of evaluation in COCO:

- AP and AR are averaged over multiple (IoU) threshold whose values gradually increased in 0.05 from .50 to .95. This is a new breakthrough in contrast with tradition concept, where AP is only calculated at the .50 single IoU value., which corresponds to the metric $mAP^{IoU=.50}$ above since this approach brings about a better localization.
- Traditionally, when AP is averaged over all categories, it is known as "mean average precision" (mAP). However, between AP and mAP now, the new evaluation provided by COCO has no difference, similar to AR and mAR.
- There are more small objects than large objects in COCO where contains approximately 41% small objects (area $< 32^2$), 34% medium objects ($32^2 < \text{area} < 96^2$), and 24% large objects (area $> 96^2$).
- Given a fixed number of detections per image, AR is the maximum recall, averaged over categories and IoUs, and computed on a per-category basis.
- There are at most 100 top-scoring detections in each image across all categories when metrics are computed.
-

The 12-evaluation metrics:

- $AP^{IoU=.50:.05:.95}$, which mean that mAP is averaged over 10 IoU thresholds (ie., 0.5, 0.55, 0.6, ..., 0.95), (primary challenge metric)
- $AP^{IoU=.50}$, (Pascal metric)
- $AP^{IoU=.75}$, (strict metric)

In addition to different IoU thresholds, there are also mAP calculated across different object scales: these variants of mAP are all averaged over IoU threshold (ie., 0.5, 0.55, 0.6, ..., 0.95).

- AP^{small} , which is mAP for small objects, approximately 41% in COCO: $area < 32^2$
- AP^{medium} , which is mAP for medium objects, approximately 34% in COCO.: $32^2 < area < 96^2$
- AP^{large} , which is mAP for large objects, approximately 24% in COCO: $area > 96^2$

Like mAP, the mAR metric also has many variations. One set of mAR variants vary across different numbers of detections per image:

- $AR^{max=1}$, which is mAR given 1 detection per image
- $AR^{max=10}$, which is mAR given 10 detections per image
- $AR^{max=100}$, which is mAR given 100 detections per image.

The other set of mAR variants vary across the size of detected objects:

- AR^{small} , which is mAR for small objects: $area < 32^2$
- AR^{medium} , which is mAR for medium objects: $32^2 < area < 96^2$
- AR^{large} , which is mAR for large objects: $area > 96^2$

The GLUON website provides the entire accuracy results of YOLOv3 and SSD trained by GPU as table 8 and table 9 shown below; however, we will focus on the first mAP, a primary basis, which are calculated over multiple the IoU thresholds. Hence, we extract only the first mAP from both tables, and the result is presented in table 10.

YOLOv3 (416x416)				
Average Precision	(AP) @[IoU=0.50:0.95	area= all	maxDets=100] =	0.358
Average Precision	(AP) @[IoU=0.50	area= all	maxDets=100] =	0.568
Average Precision	(AP) @[IoU=0.75	area= all	maxDets=100] =	0.384
Average Precision	(AP) @[IoU=0.50:0.95	area= small	maxDets=100] =	0.168
Average Precision	(AP) @[IoU=0.50:0.95	area= medium	maxDets=100] =	0.382
Average Precision	(AP) @[IoU=0.50:0.95	area= large	maxDets=100] =	0.525
Average Recall	(AR) @[IoU=0.50:0.95	area= all	maxDets= 1] =	0.29
Average Recall	(AR) @[IoU=0.50:0.95	area= all	maxDets= 10] =	0.431
Average Recall	(AR) @[IoU=0.50:0.95	area= all	maxDets=100] =	0.444
Average Recall	(AR) @[IoU=0.50:0.95	area= small	maxDets=100] =	0.235
Average Recall	(AR) @[IoU=0.50:0.95	area= medium	maxDets=100] =	0.471
Average Recall	(AR) @[IoU=0.50:0.95	area= large	maxDets=100] =	0.625

Table 8. The mAP, and mAR of the YOLOv3 (416x416) [21].

SSD (300x300)				
Average Precision	(AP) @[IoU=0.50:0.95	area= all	maxDets=100] =	0.251
Average Precision	(AP) @[IoU=0.50	area= all	maxDets=100] =	0.429
Average Precision	(AP) @[IoU=0.75	area= all	maxDets=100] =	0.258
Average Precision	(AP) @[IoU=0.50:0.95	area= small	maxDets=100] =	0.06
Average Precision	(AP) @[IoU=0.50:0.95	area= medium	maxDets=100] =	0.268
Average Precision	(AP) @[IoU=0.50:0.95	area= large	maxDets=100] =	0.421
Average Recall	(AR) @[IoU=0.50:0.95	area= all	maxDets= 1] =	0.233
Average Recall	(AR) @[IoU=0.50:0.95	area= all	maxDets= 10] =	0.344

SSD (300x300)					
Average Recall	(AR) @[IoU=0.50:0.95	area= all	maxDets=100] =		0.366
Average Recall	(AR) @[IoU=0.50:0.95	area= small	maxDets=100] =		0.114
Average Recall	(AR) @[IoU=0.50:0.95	area= medium	maxDets=100] =		0.398
Average Recall	(AR) @[IoU=0.50:0.95	area= large	maxDets=100] =		0.571

Table 9. The mAP, and mAR of the SSD (300x300) [21].

Model		AP
YOLOv3	Average Precision (AP) @[IoU=0.50:0.95 area= all maxDets=100] =	0.358
SSD	Average Precision (AP) @[IoU=0.50:0.95 area= all maxDets=100] =	0.251

Table 10. The mAP comparison between YOLOv3 and SSD [21].

4.2 The standard evaluation of speed performance criterion for object detectors

Frame per second (FPS) is a standard evaluation criterion used to measure how fast it is for a proposed network model to detect objects frames per second on average. Also, it has been called frame rate, or frame frequency. If the rate is higher, it means that it has a better performance to deal with more images each second. According to the previous study, the FPS of YOLOv3 and SSD are given as table 11 shown. YOLOv3 can process 29 frames per second while SSD is able to process 59 frames per second. The FPS might be varied depending on the performance of hardware devices like GPU, CPU and so on. However, we at least can claim that SSD model detection has a higher speed than YOLOv3.

Model	Frame Per Second (FPS)
YOLOv3 (416x416)	21
SSD (300x300)	59

Table 11. The FPS comparison for YOLOv3 and SSD [4].

4.3 Performance Discussion

According to the mAP table provided above trained by researches of previous studies, we can claim that YOLO is more accuracy than SSD, while SSD is faster than YOLO. Similarly, the result of our implementation outputs matches the studied results regardless of whether one image, video, and real-time testing, like the tables 12, 13, 14 shown below respectively. In terms of one image, and video testing, from table 12 and table13, we can see YOLO has a better performance in accuracy, which we can trace back to the section 3.1 and 3.2 we can simply observe the difference between two detectors and circle them in color red, whereas it takes more time to process one frame, approximately 4 to 5 times more. Besides, in terms of real-time implementation, the output result shows no difference between image and video, within one second, SSD detector processes 10 times more frames than YOLO. However, with the YOLOv2 model, its velocity, which is able to process about 9 frames per second, is way better than YOLO although it is slightly slower than SSD's 15 frames almost per second.

One image output result:

Model	YOLOv3	SSD
Accuracy	High	Low
Time	0.84 ~ 0.9 (second / per frame)	0.17 ~ 0.23 (second / per frame)
Speed	slow	Fast

Table 12. The comparison of one image testing between YOLOv3 and SSD

Video output result:

Model	YOLOv3	SSD
Accuracy	High	Low
Time	0.888 ~ 0.9180 sec / per frame	0.1830 ~ 0.237 second / per frame
Speed	slow	fast

Table 13. The comparison of video testing between YOLOv3 and SSD

Real time output result:

Model	YOLOv3	SSD	YOLOv2
FPS	1.39	14.72	9.05

Table 14. The comparison of real-time testing between YOLOv3, SSD, and YOLOv2

5. Conclusion and future work

According to our implementation results of one image and video, and real time object detection. It is quite obvious that YOLO performed more accurate than SSD because some objects in the same image, or video were not detected by SSD pre-trained model but were detected by YOLO pre-trained model simply through observation. However, is also very clear for speed in the image and video detection between two models but opposite in contrast to accuracy that SSD surpassed the YOLO model. Admittedly, YOLO has better performance in accuracy while SSD has a better performance in speed. If users care more about the accuracy of detection, it is better to choose YOLO, while SSD might be applied if speed is taken into account. Or if users need high accuracy, but a slight decrease in accuracy can be acceptable, then it might be a good alternative to switch to the YOLOv2 model. There is always a tradeoff between speed and accuracy depending on what we need and focus on.

In this project, we did a small-scale evaluation of the rate in speed and accuracy through image, video, and real time instead of going through the process of mAP calculation due to the time, financial limitation. In order to process a large scale of the dataset, we have to possess a very powerful and expensive GPU to deal with it, and it still takes a great time to complete the process. Besides, if we are based on the same dataset such as all the same pictures and data from COCO, the result of the evaluation of performance in accuracy will be almost the same as previous studies due to the same model trained with the same images. These results of the mAP studied by researchers previously can be searched online. Therefore, we decide to use the random images, videos and real-time camera to measure the performance between two detectors directly. However, these data do not have a ground truth box marked in an annotation file. That is another reason we do not follow the abovementioned steps to compute the mAP. Hence, in the future we might consider evaluates mAP through a dataset with its provided ground truth to verify our output if we would like to prove that we can get resemble consequence, or further plan on creating our own small dataset and manually draw ground truth box with label attached for each object, so we can evaluate our own training model with the YOLO or SSD architectures and gradually increase our data to a certain reasonable level, which might be considered to be more objective.

6. References

1. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You only look once: Unified, real-time object detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
2. J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger". Proceedings of the IEEE conference on computer vision and pattern recognition, arXiv:1612.08242 [cs.CV], 2017.
3. J. Redmon and A. Farhadi, "Yolov3: An incremental improvement". Preprint arXiv:1804.02767 [cs.CV], 2018.
4. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in European Conference on Computer Vision (ECCV), 2016, pp. 21–37.
5. Pulli, Kari; Baksheev, Anatoly; Korniyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue: 40:40–40:56. doi:10.1145/2181796.2206309 (inactive 2019-07-14).
6. Object detection, https://en.wikipedia.org/wiki/Object_detection
7. How computers learn to recognize objects instantly, https://www.ted.com/talks/joseph_redmon_how_a_computer_learns_to_recognize_objects_instantly#t-12277, April 2017, Redmon, Joseph's speech in TED.
8. Joseph Chet Redmon's personal website, <https://pjreddie.com/>
9. Intel unveils the intel neural compute stick 2 <https://newsroom.intel.com/news/intel-unveils-intel-neural-compute-stick-2/#gs.sbwxdp>, Nov 14th, 2018.
10. Beginner's Guide to Object Detection Algorithms, <https://towardsdatascience.com/beginners-guide-to-object-detection-algorithms-6620fb31c375>, April 27th, 2019.
11. Object Detection YOLO v1, v2, v3, <https://medium.com/@venkatakrishna.jonnalagadda/object-detection-yolo-v1-v2-v3-c3d5eca2312a>, Jan 30th, 2019.
12. A Beginner's Guide to Object Detection, <https://www.datacamp.com/community/tutorials/object-detection-guide>, April 19, 2018
13. OpenCV, <https://en.wikipedia.org/wiki/OpenCV>
14. Object Detection and Image Classification with YOLO, <https://www.kdnuggets.com/2018/09/object-detection-image-classification-yolo.html>, Michal Maj, Sep, 2018.
15. An overview of object detection: one-stage methods, <https://www.jeremyjordan.me/object-detection-one-stage/#ssd>, Jeremy Jordan, July 11th, 2018.
16. Understanding YOLO, <https://hackernoon.com/understanding-yolo-f5a74bbc7967>, Mauricio Menegaz, October 14th, 2019.
17. mAP (mean Average Precision) for Object Detection, https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, Jonathan Hui, March 6th, 2018
18. An Introduction to Evaluation Metrics for Object Detection, <https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/>, Nick Zeng, Dec 6th, 2018
19. Common Objects in Context, <http://cocodataset.org/>
20. Pyimagesearch, <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>, Adrian Rosebrock
21. GluonCV: a deep learning toolkit for computer vision, https://gluon-cv.mxnet.io/model_zoo/detection.html#idl