Vehicle Identification from Traffic Surveillance Videos

Vibhanshu Singh Sindhu,
Software Engineering,
Delhi Technological University,
Delhi,India
vibhanshusinghsindhu_2k18se1
31@dtu.ac.in

***Abstract*—The ability to detect & follow vehicles, both individual & public transport ones, is very useful in numerous applications such as the safety of the drivers, preventing & monitoring road accidents etc. Using continuous video stream from CCTV cameras from all over the world, this project deals with the concept of vehicle detection with the support of computer vision algorithm in real-time frame. The proposed framework capitalizes on YOLOv4 to achieve faster object detection in real time, and using the proposed dataset it was tested on various conditions such as rain, low visibility, daylight, snow and night. This project can further help in the development of a framework for vehicular accident detection in real time.**

Keywords—object detection, YOLOv4, vehicle-identification.

I. Introduction

We live in a world where vehicular traffic has become a crucial part of our lives today which effects human activities in a lot of different ways. Thus, proper management of this traffic has become crucial, especially in urban cities where people commute on daily basis. This is also a matter of utmost concern from the point of view of loss of human lives as well the damage to property, which is increasing year-by-year. Despite numerous advancements being done on making traffic flow as smooth as possible, there are millions of accidents reported on yearly basis. In fact, road traffic accidents have been classified as the 9th foremost basis of human life loss. According to the data, approximately 1.26 million people lose their lives in road accidents on an yearly basis which account for 2.2 percent of all casualties worldwide and in addition to that nearly 25-55 million people are injured or disabled every year. With this surge in increase every year, it is predicted to be the 5th leading cause of human casualties by the year 2030.

Currently, a lot of techniques are used as a preventive measure to help monitor the accidents on the roads. CCTVs are installed at the intersection of roads, radars are placed on highways to capture the cases of over-speeding cars etc. Despite all these measures many lives are lost due to the delay in timely reporting of these accidents which is a consequence of these technologies relying on human observation of these footages which leads to late medical aid provided to the wounded. Also, this does not support any immediate response of spontaneous events and requires an extensive amount of determination from the human operatives.

Thus now, in efforts to overcome this gruelling task of providing medical help on time without the need of a human operator, computer vision algorithms are used. Hence this paper proposes the first step in the detection of vehicular collisions, i.e. an efficient model to detect all the various types of vehicles traversing on roads on a real-time reference frame which is trained on well-developed training sets using the new and efficient computer vision based YOLOv4 object detection algorithm. This model is efficient for real-time conditions such as weather changes, sunlight altercations etc.

The outline of the rest of the paper is as follows. Section II briefly explains experimental design. Section III outlines the research methodology of the paper. Section IV explains the performance measures used to evaluate the performance of our model. Section V exemplifies the results of the experiment. Section VI discusses the conclusion and future areas of work.

II. Experimental Design

All the experiments were conducted using Google Colaboratory which used virtual GPU with 12.72GB Main Memory (RAM) and 68.40GB Disk Space. All programs were written in *Python* – 3.8. Video processing was done using *OpenCV*4.1.

A. Dataset

The dataset used for this project’s accomplishment is a collection of 100 mp4 videos which contains CCTV footage of numerous road accidents happened all round the world. These videos have various types of vehicles traversing on roads, including truck, motorcycle, bus and of course car. These videos are shot in various weather and various sunlight conditions, such as broad daylight, night, hail, snow, rain etc. Now, the videos of this dataset are converted into 17000 images by breaking each second of the video in 5 frames, i.e. fps=5. Now, for the detection of the vehicles, we are using the images in the ratio of 80:20 for training and testing, respectively. Link: https://drive.google.com/drive/u/0/folders/1b_-BsRsl6OQS1t9r2VlO1i71jzK3f4rQ.

B. Research Variables

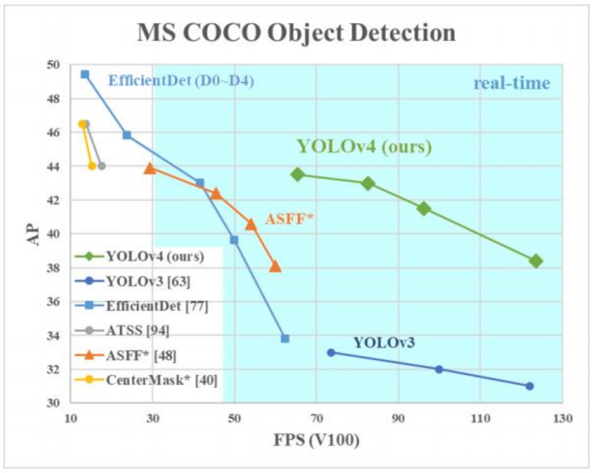
We are using the traffic video surveillance as dataset from which the videos are considered frame wise & a particular frame which is converted into blob (Binary large object) is pre-processed to detect the vehicles on the road. Thus, input variable is the videos from traffic surveillance converted to blob framewise & Output variables are the dimensions of the predicted bounding boxes of the vehicles detected.

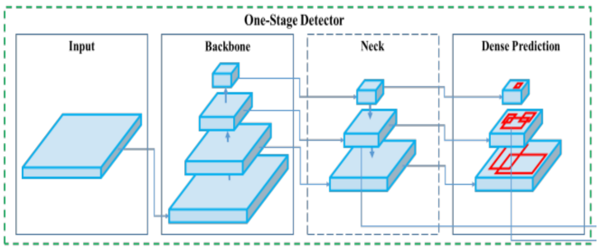
C. *Techniques Used*

To break the videos into several frames we have used 'ffmpeg' tool with the help of git bash with frame per second(fps) = 5 on each video. The annotation, i.e. the labelling of several vehicles in these frames is done through the 'labelimg' tool using git bash. The object detection algorithm used for training & testing of the model is YOLOv4 which is the latest in its domain.

III. Research Methodology

This section describes our proposed framework. We have used a new, latest object detection algorithm YOLOv4 for the detection of the vehicles in real-time footage.





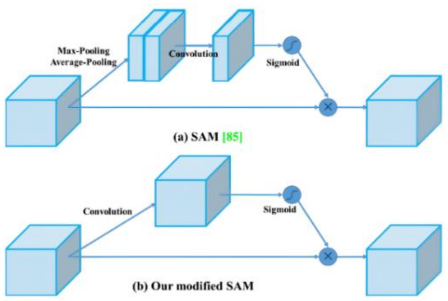


Figure 5: Modified SAM.

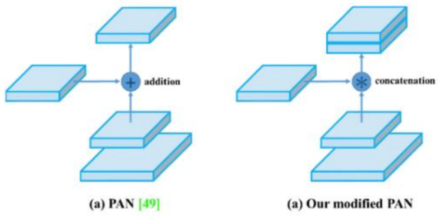


Figure 6: Modified PAN.

Our foremost aim is to deliver a simple yet rapid method for detecting the various types of vehicles on the road with great efficiency. The anticipated vehicle detection algorithm comprises of the following vital tasks:

A: Data Pre-processing

B: Feature Mining

C: Vehicle Identification

The projected purpose of the proposed framework is achieved via the following stages:

A. *Data Pre-processing*

In this phase of the framework, we first download the YOLOv4 darknet framework and the standard weights file for training and testing. Now, we make a custom cfg file according to the requirements of our model to be trained, i.e. containing four classes namely car, truck, motorcycle and bus. And thus, according to our four classes we make following changes in our standard YOLOv4 cfg file: batch=64, subdivisions=64, width=416, height =416, max_batches = 8000 (number of classes*2000), steps = 7200 (90 percent of max_batches) and in every convolutional layer above yolo layer, set filters = 27(number of classes + 5)*3, and in every yolo layer below convolutional layer, set classes = 4. We now make changes in the standard Makefile of the YOLOv4 darknet framework. Set GPU=1, CUDNN=1, OPENCV =1, CUDNN_HALF =0.

Now, we download our dataset into the darknet folder of YOLOv4. Now, we first break these videos of our dataset into frames. We do this with the help of git bash and 'ffmpeg' tool with setting fps=5 for every video. Now for every second of each video we have 5 frames, i.e. images in the jpg format. Now we have to annotate these frames and label the various kinds of vehicles traversing on the roads in various weather conditions. For this, we use git bash and the 'labelimg' tool. Alongside labelling the vehicles in frames in rectangle boxes, the labelimg tool generates a .txt file for each image containing the label of the class, coordinate of the x-centre, coordinate of the y-centre, width of the rectangle box, and height of the rectangle box respectively for each of the vehicle labelled in the frame. Now, we split these .txt files in the ratio of 80:20 for training and testing txt respectively. Now, we create a name file with the names of all the four classes, and a data file where we pass training and testing files to the train and valid respectively, the names file, backup=backup and classes=4.

B. *Feature Mining*

With the help of the custom YOLOv4 darknet framework we processed, we train our model for 8000 iterations. At the end when our model training is completed, we get the 'best weights' file which is suitable according to our model, has been trained and tested on our dataset and can be used to detect vehicle in videos.

C. *Vehicle Identification*

With the best weights file we got from training our model, and the custom made cfg file from earlier we take any video input and break it into current frame, then convert it into blob, and framewise predict the vehicles in the video with the output containing the bounding box surrounding the predicted vehicle and also the class of the vehicle among the four pre-determined classes, i.e. car, truck, motorcycle and bus with the accuracy percentage with which they are predicted.

IV. Performance Measures

There are various metric to measure the accuracy of object detection models. Thus, we didn't limit ourselves to one of these measures. To evaluate our model's performance, we have taken in account the following performance measures:

Precision

Recall

F1 Score: It is the harmonic mean (HM) of precision and recall.

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

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

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

TP = True positive

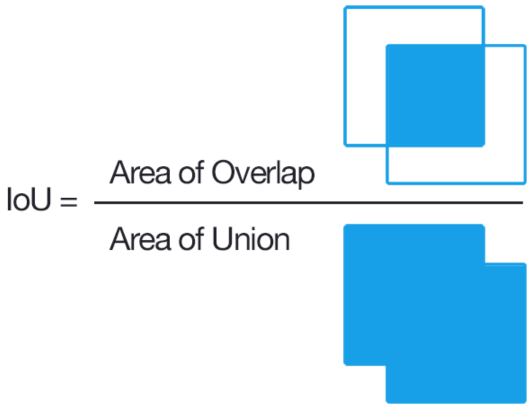
TN = True negative

FP = False positive

FN = False negative

IoU (Intersection over union): It measures how much our predicted boundary overlaps with the real object boundary.

The IoU threshold set is 0.5.



AP (Average Precision): It basically finds the area under the precision-recall curve.

$$AP = \Sigma\,(r_{n+1} - r_n)\, p_{interp}(r_{n+1})$$

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

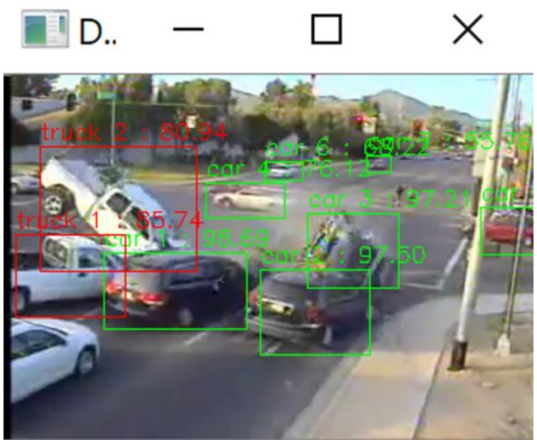
mAP (Mean Average Precision): It is defined as the average of the AP determined for all the classes.

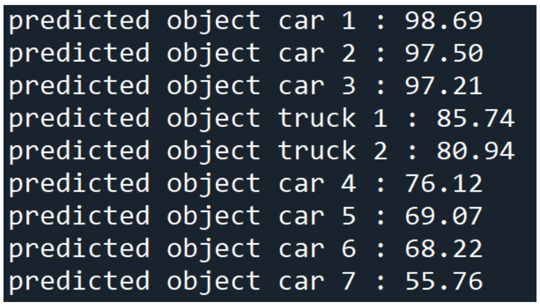
As it is calculated at IoU threshold of 50%, i.e. 0.50, it is generally written as mAP@0.50.

Also, the confidence threshold is set at 25%, i.e. 0.25.

V. Results

With the best weights file, we achieved after training our model, we ran a code using it to identify vehicles in real-time CCTV footages and some of the results are shown below for example:

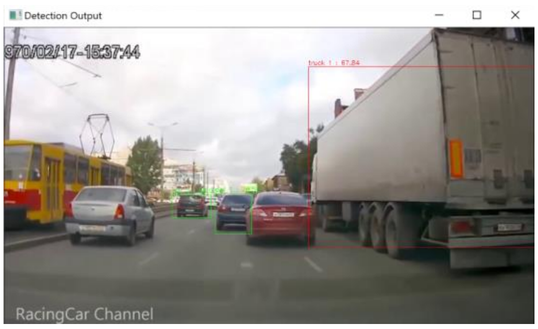


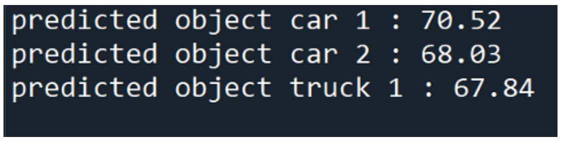


```
predicted object car 1 : 98.69
predicted object car 2 : 97.50
predicted object car 3 : 97.21
predicted object truck 1 : 85.74
predicted object truck 2 : 80.94
predicted object car 4 : 76.12
predicted object car 5 : 69.07
predicted object car 6 : 68.22
predicted object car 7 : 55.76
```

This is a perfect prediction in which all the cars and trucks in the frame are identified with great precision. This suggests that our model has been trained good and the weights file we get after training is of good use in the identification of vehicles on the road.

Class of vehicles in frame: car, truck.

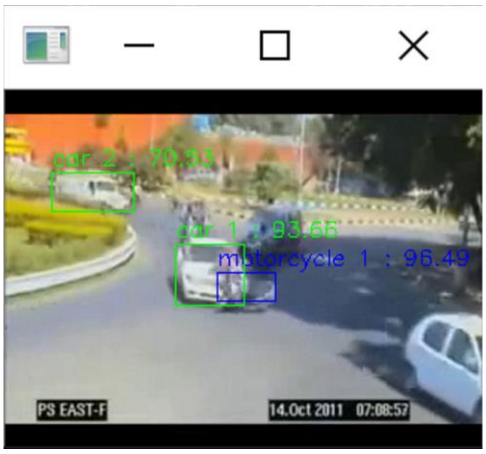


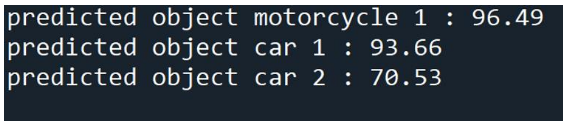


```
predicted object car 1 : 70.52
predicted object car 2 : 68.03
predicted object truck 1 : 67.84
```

In this prediction we can see, not all the vehicles on the road were identified in this frame, i.e. only 3 of the 5 vehicles in this frame were identified. Also, the accuracy is not as good in the previous example. This suggests that our model isn't perfect and can be improved.

Class of vehicles in frame: car, truck.

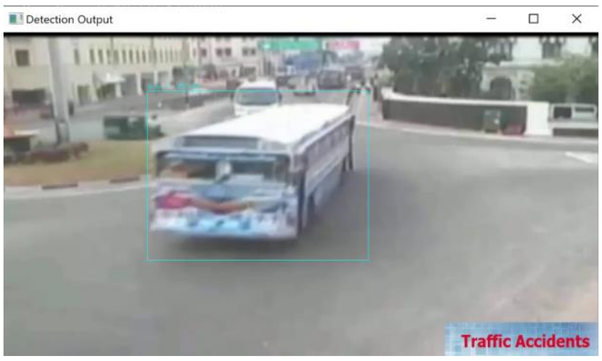


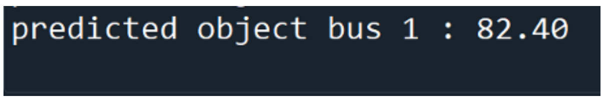


```
predicted object motorcycle 1 : 96.49
predicted object car 1 : 93.66
predicted object car 2 : 70.53
```

As we can see, the vehicle of class 'motorcycle' is identified with great precision in the above frame shown. This signifies that our model works good on various classes of vehicles.

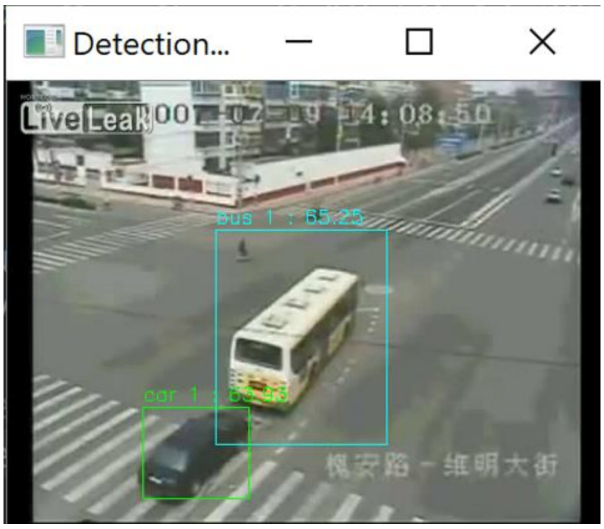
Class of vehicles in frame: car, motorcycle.

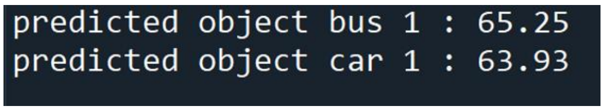




```
predicted object bus 1 : 82.40
```

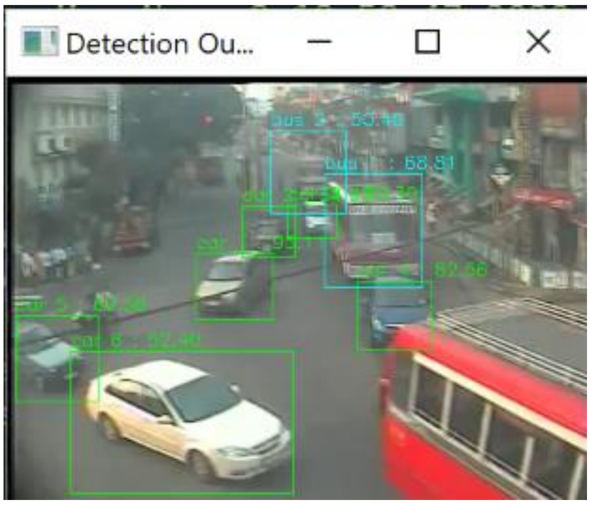
Class of vehicles in frame: bus.

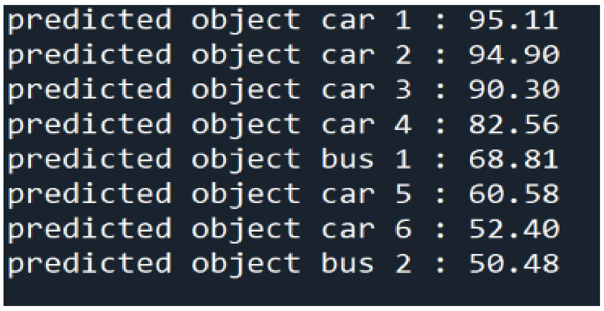




```
predicted object bus 1 : 65.25
predicted object car 1 : 63.93
```

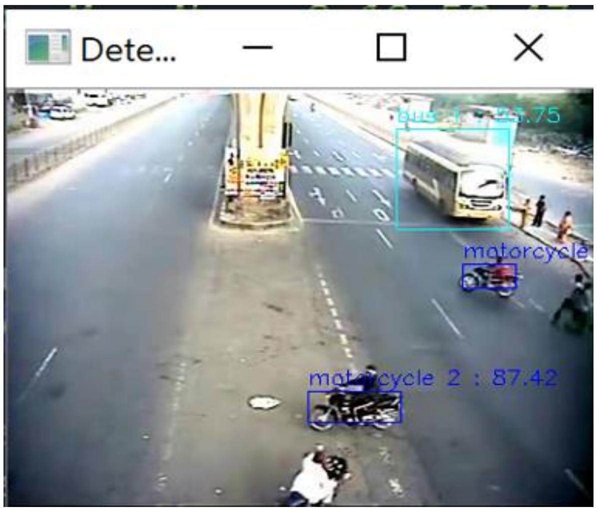
Class of vehicles in frame: bus, car.

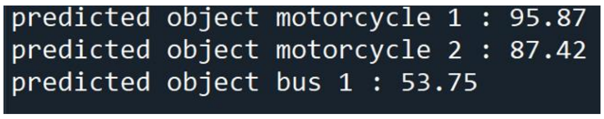




```
predicted object car 1 : 95.11
predicted object car 2 : 94.90
predicted object car 3 : 90.30
predicted object car 4 : 82.56
predicted object bus 1 : 68.81
predicted object car 5 : 60.58
predicted object car 6 : 52.40
predicted object bus 2 : 50.48
```

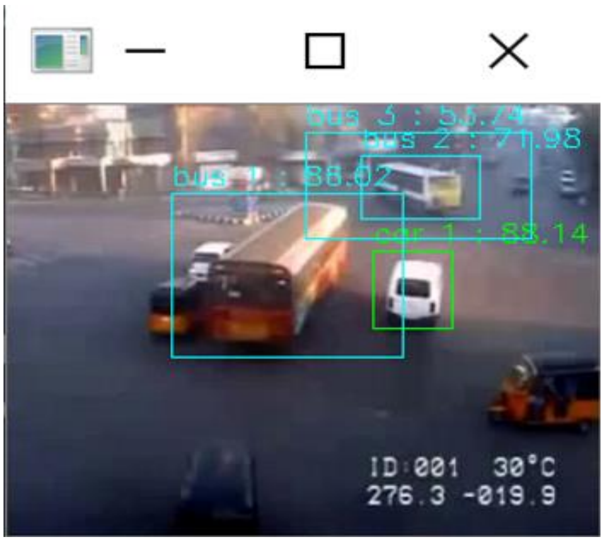
Class of vehicles in frame: bus, car.

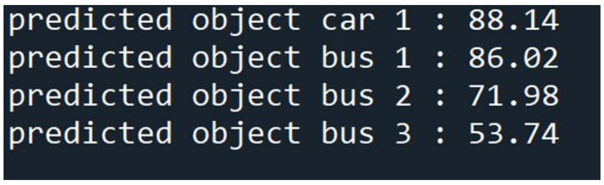




```
predicted object motorcycle 1 : 95.87
predicted object motorcycle 2 : 87.42
predicted object bus 1 : 53.75
```

Class of vehicles in frame: motorcycle, car.

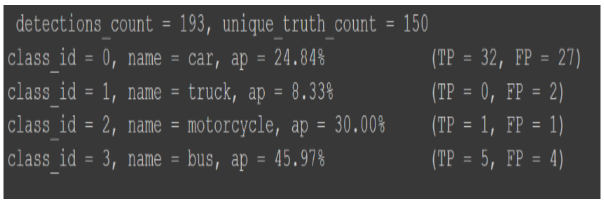




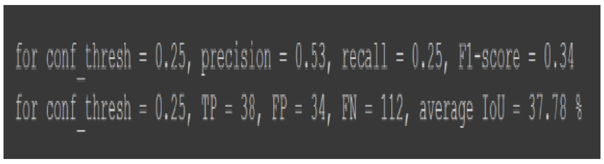
```
predicted object car 1 : 88.14
predicted object bus 1 : 86.02
predicted object bus 2 : 71.98
predicted object bus 3 : 53.74
```

Class of vehicles in frame: bus, car.

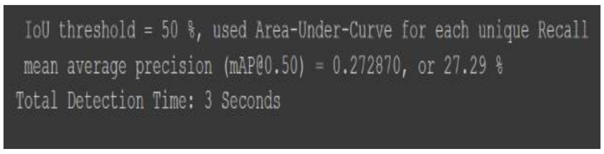
The value of the performance measures after training the model are shown below:



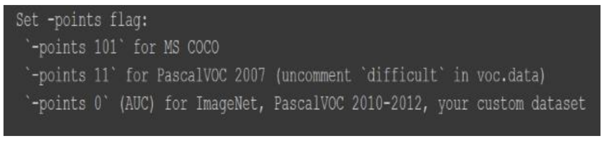
```
 detections_count = 193, unique_truth_count = 150
class_id = 0, name = car, ap = 24.84%           (TP = 32, FP = 27)
class_id = 1, name = truck, ap = 8.33%          (TP = 0, FP = 2)
class_id = 2, name = motorcycle, ap = 30.00%    (TP = 1, FP = 1)
class_id = 3, name = bus, ap = 45.97%           (TP = 5, FP = 4)
```



```
for conf_thresh = 0.25, precision = 0.53, recall = 0.25, F1-score = 0.34
for conf_thresh = 0.25, TP = 38, FP = 34, FN = 112, average IoU = 37.78 %
```



```
 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.272870, or 27.29 %
Total Detection Time: 3 Seconds
```



```
Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

Precision = 0.53

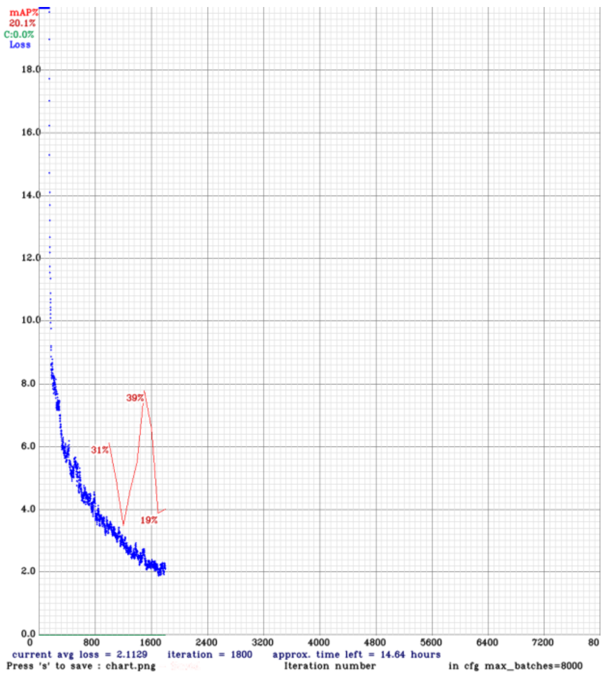
F1-score = 0.34

Recall = 0.25

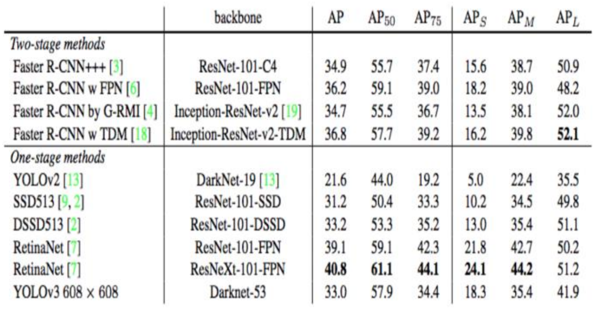
Average IoU = 37.78%

Mean average precision (mAP@0.50) = 27.29%

Best mean average precision = 39% (chart shown below)



The standard result of some famous detection algorithm, for compariosn purposes are shown below:



	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

COCO for YOLOv3

VI. Conclusion and Future Work

A new framework object detection algorithm has been used to detect various types of vehicles traversing on road from the real-time CCTV footage in this paper. We have seen some good results with this new algorithm. The precision achieved is good for start and the used dataset and the average IoU is also appreciable for the model.

For future work:

a) We would use a new dataset with more number of videos which are shooted in various weather and sunlight conditions.

b) We would work towards the second and third step of vehicular- collision detection, i.e. vehicle tracking and accident detection, respectively.

VII. Acknowledgements

We would like to express our sincere thank to Google Colaboratory for providing the needed GPU hardware for carrying out the experiments and Google Drive for providing the videos used in this dataset.

VIII. References

[1] "Computer Vision-based Accident Detection in Traffic-Surveillance" https://arxiv.org/pdf/1911.10037.pdf

[2] "Vehicle Tracking using video surviellance" by Sandesh Srestha https://www.intechopen.com/books/intelligent-system-and-computing/vehicle-tracking-using-video-surveillance

[3] "A vision-based video crash detection framework for mixed traffic flow environment considering low-visibility conditions." https://doi.org/10.1155/2020/9194028.

[4] "Yolov4: Optimal Speed and accuracy of object detection" https://arxiv.org/abs/2004.10934.