**You only Look Once (YOLO)**

For YOLO, detection is a straightforward regression dilemma which takes an input image and learns the class possibilities with bounding box coordinates. YOLO divides every image into a grid of S x S and every grid predicts N bounding boxes and confidence. The confidence reflects the precision of the bounding box and whether the bounding box in point of fact contains an object in spite of the defined class. YOLO even forecasts the classification score for every box for each class. You can merge both the classes to work out the chance of every class being in attendance in a predicted box.

So, total SxSxN boxes are forecasted. On the other hand, most of these boxes have lower confidence scores and if we set a doorstep say 30% confidence, we can get rid of most of them.

**Single Shot Detector (SSD)**

SSD attains a better balance between swiftness and precision. SSD runs a convolutional network on input image only one time and computes a feature map. Now, we run a small 3×3 sized convolutional kernel on this feature map to foresee the bounding boxes and categorization probability.

SSD also uses anchor boxes at a variety of aspect ratio comparable to Faster-RCNN and learns the off-set to a certain extent than learning the box. In order to hold the scale, SSD predicts bounding boxes after multiple convolutional layers. Since every convolutional layer functions at a diverse scale, it is able to detect objects of a mixture of scales.

**YOLO Vs SSD**

1. SSD is a healthier recommendation. However, if exactness is not too much of disquiet but you want to go super quick, YOLO will be the best way to move forward. First of all, a visual thoughtfulness of swiftness vs precision trade-off would differentiate them well.

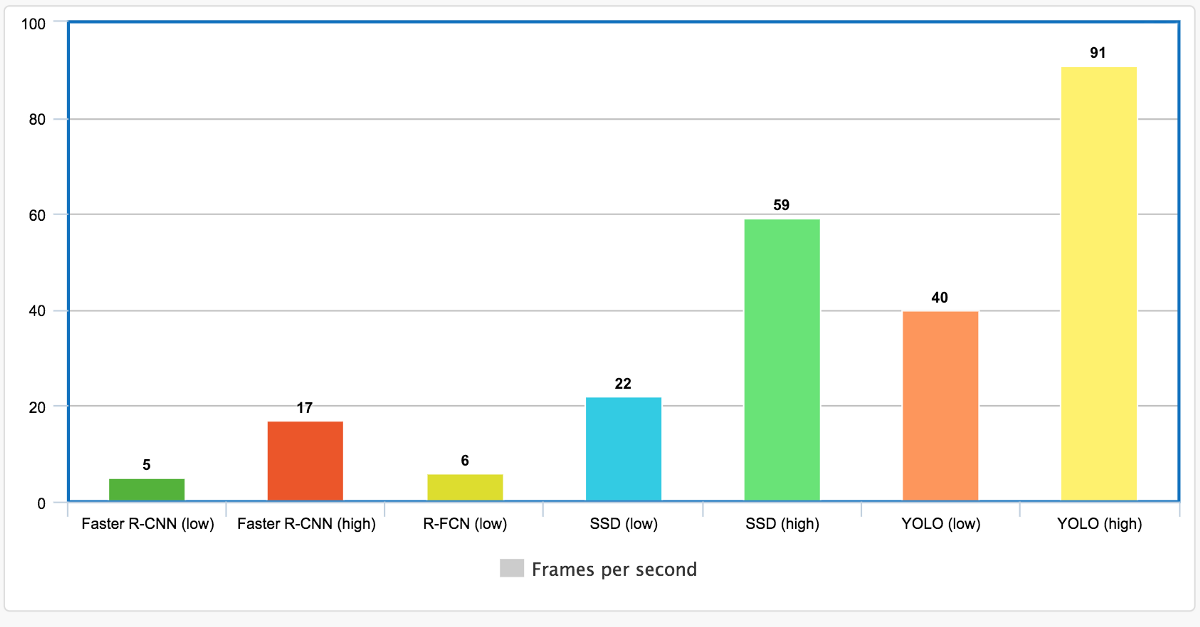
SSD is a better option as we are able to run it on a video and the exactness trade-off is very modest. While dealing with large sizes, SSD seems to perform well, but when we look at the accurateness numbers when the object size is small, the performance dips a bit.

2. The trade-off between speed and accuracy is accompanied with computational power available. The YOLO model is suitable for high-speed outputs, where accuracy is not that high… whereas SSDs provide higher accuracies with high-speed outputs with a higher computation time.

Hence choose SSDs on good microprocessors, else YOLO is the goto for microprocessor-based computations.

SSDs: mAP: 0.83 Time /epoch: 12 minutes

YOLOs: mAP: 0.85 Time/epoch: 11 minutes



**Future Work**

1. As the Traffic system is very much time sensitive and so in order to make our algorithm more efficient we can not afford high computational cost in terms of time to count the number of cars. So we have taken an image frame before a unit of time then counted the no. of vehicles in the image frame instead of constant monitoring on the lane. So, this approach of processing a single frame reduces the computational cost by a large margin than processing a video instead. Thus, we do not need very powerful machines in terms of CPU and GPU to process a single frame of image.

But if we have powerful resources(powerful CPUs and GPUs) installed at server side, then we can afford constant monitoring on the lanes by processing larger number of frames per unit time in order to get better result and make the algorithm more efficient.

2. In place of arduino camera module, we can use high resolution

camera for higher detection accuracy.

3. Lanes having emergency vehicles can be given more priority than

lanes devoid of them, thus, allowing their faster movement.

**Installation**

To implement YOLO -

1. Go to <https://pjreddie.com/darknet/yolo/>
2. Copy these commands and run in terminal sequentially-

git clone <https://github.com/pjreddie/darknet>

cd darknet

Make

3. We already have the config file for YOLO in the cfg/ subdirectory. we have to download the pre-trained weight file.

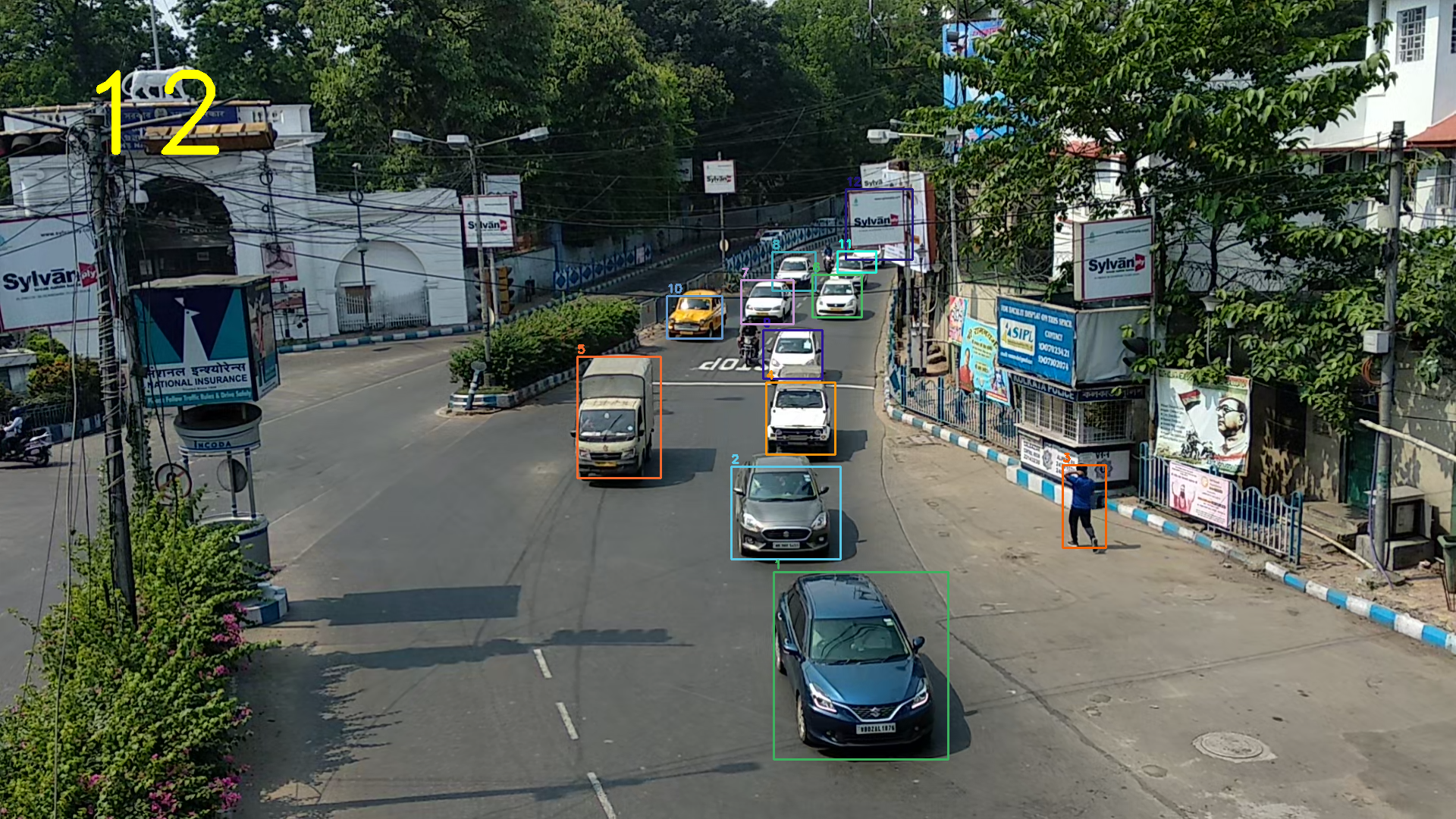
wget <https://pjreddie.com/media/files/yolov3.weights>

**For counting cars in a single Frame:**

1. Go to [***https://github.com/guillelopez/python-traffic-counter-with-yolo-and-sort***](https://github.com/guillelopez/python-traffic-counter-with-yolo-and-sort)and clone it.
2. Go to Specified Folder named (TRAFFIC\_COUNTER) and there in the inputs folder put the image for detection.
3. Now put the pre-trained weight file (*yolov3.weights*) in the *yolo-coco* folder
4. Run the command->> ***python mod.py --input input/car.bmp --output output/cars.bmp --yolo yolo-coco***
5. Now we get the classified objects in the image with no. of counts of the car

**Photo Taken from Alipore Foot Bridge:**





<https://pjreddie.com/darknet/yolo/>

<https://github.com/guillelopez/python-traffic-counter-with-yolo-and-sort>