CUSTOM OBJECT DETECTION, TRACKING AND WEB API WITH YOLO AND FLASK USING DARKNET NEURAL NETWORK FRAMEWORK

Jaivik Rathod[1], Dr. Kiran Trivedi[2]

[1]PG.Scholar, Dept. of Electronic and Communication, Vishwakarma Government Engineering College, Gujarat, India
[2]Professor, Dept. of Electronic and Communication, Vishwakarma Government Engineering College, Gujarat, India

---***---

Abstract - *Deep learning is widely used for advanced applications of image and video processing with high performance levels. Deep learning neural networks make use of the higher levels of accuracy in prediction and dynamic data analysis. Deep neural network has shown its extraordinary performance in different task of computer vision and machine learning tasks. These types of networks often require large sets of labeled data for training and involve high computational complexity. This poses considerable challenges for the development and deployment of deep neural networks in real-time systems. In The proposed research work we analyzes the custom object-public sector car and state government car detection and tracking system. Images frames from video sequence are used to detect moving vehicles based on Yolov3 object detection algorithm with darknet frame work to trained own custom data model. And results display on webpage using Flask API. This deep learning method showed better classification and detecting rate compare to background subtraction techniques. The percentage evolution of object detection rate is discussed in final result.*

Key Words: **Deep learning, Object detection, Darknet, Convolution Neural Network, OpenCV, Yolo**

1. INTRODUCTION

Deep learning is largely used for advanced applications of image and video processing application with high performance levels. Deep learning neural networks make use of the higher levels of accuracy in prediction and dynamic data analysis. Vehicle data recognition is a key part of Intelligent Transportation Systems [1]. Vehicle data recognition include types of vehicles, number plate recognition etc. One of the easiest techniques for the object detection is the background subtraction. Background Subtraction is commonly used in the fields of video surveillance, optical motion capture and multimedia application where it needs to be the first step to detect the moving objects in the scene. There are different methods which are for Background Subtraction [2].

These different techniques have unique strengths and their weaknesses in terms of performance and computational requirements. With the advent of artificial intelligence, the popular algorithms used for perform object detection the convolutional neural network, R-CNN (Region-Based Convolutional Neural Networks) Fast R-CNN, and YOLO (You Only Look Once). R-CNN and Fast R-CNN are slower than YOLO and it's fails to perform real time detection, YOLO is good at regression than classification and it can perform real time classification with good speed[paper]. YOLO is one of the powerful methods of real-time object detection with integration of advanced deep learning.

In The proposed research work we analyzes the custom object-public sector car and official state government car detection and tracking system. Images frames from video sequence are used to detect moving vehicles based on Yolov3 object detection algorithm with darknet frame work to trained own custom data model.

2. TECHNICAL REVIEW

There are several methods for detection and tracking some of techniques employed are discussed below.

Rahul Dutt Sharma have proposed optimized dynamic background subtraction technique for moving object detection and tracking. They have discussed the proposed technique shows accurate and wider foreground compared to existing background and two frame technique and reduced holes problem effectively [2].

Syed Mazhar Abbas designed region-based object detection and classification using Fast R-CNN they trained Faster R-CNN using custom based data set of images. Showed trained network efficiently detects objects from an image consisting of multiple objects. And also, network requires minimum GPU capability of 3.0 or higher [3].

Yongbon Koo have proposed OpenCL-Darknet: An OpenCL Implementation for Object Detection they have discussed the OpenCL-Darknet, which transforms the CUDA-based Darknet – a deep learning based object detection framework – into an open standard OpenCL backend also they evaluated the OpenCL Darknet in AMD R7-integrated APU with OpenCL 2.0 and AMD Radeon RX560 with OpenCL 1.2 using the VOC 2007 dataset [4].

Priyanka Malhotra comprised object detection techniques RCNN, Fast RCNN and YOLO are the common techniques employed for object detection. RCNN and Fast

RCNN are slower than YOLO but can detect small objects. YOLO is good at regression than classification. YOLO has difficulty in classifying small objects. Both RCNN and Fast RCNN fails to perform real time detection but YOLO can perform real time classification with good speed [5].

Ajeet Ram Pathak have proposed application of Deep learning for object detection they demystified the role of deep learning techniques based on CNN for object detection Deep learning frameworks and services available for object detection are also discussed, they also represent dataset of pascal voc [6].

Chun Ju Huangv designed chip and system for real time object detection based on ls-r-yolo network in system they adopted point cloud and image feature distribution algorithm, also classify cause of error between the final bounding box and detection frame due to algorithm takes real-time as the main consideration [7].

3. METHODOLOGY

This work trained custom object - public sector car and official state government car for detection and tracking, also analyzed result on webpage. To trained custom data certained steps followed as shown in fig 1.

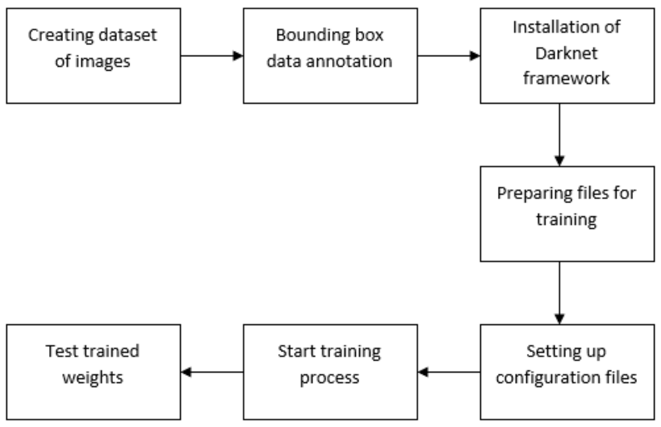

Creating dataset of images
Bounding box data annotation
Installation of Darknet framework
Preparing files for training
Setting up configuration files
Start training process
Test trained weights

Fig -1: Diagram of proposed methodology

I. CREATING DATASET OF IMAGES

For each object we have collected set of images that discover object in shape, side of object, relative size, angle of rotation, tilt, illumination. We have collected individually 500 images for public sector car as well as official state government car. For official state government car, we analyze some specification to classify the difference between those cars for example some national emblem, national flag, Ashoka sign etc.

Table -1: Number of training images

Classes	Number of images
Public Sector Car	500
State Government Car	500
Total	1000

II. BOUNDING BOX DATA NNOTATION/DATA LABELING IN YOLO FORMAT

Data labelling is an essential step in a supervised machine learning task. Annotations used for data labelling Bounding boxes are the most commonly used type of annotation in computer vision.

There are different types of data annotation techniques like bunding box, semantic, contour annotation and Bounding boxes - are rectangular boxes used to define the location of the target object. They can be determined by the *x* and *y* axis coordinates in the upper-left corner and the *x* and *y* axis coordinates in the lower-right corner of the rectangle. Bounding boxes are usually represented by either two coordinates (x1, y1) and (x2, y2) or by one co-ordinate (x1, y1) and width (w) and height (h) of the bounding box using **lABELIMG** tool, it is used a Python Qt5 library. We drawn the bounding box in images that locate the object.

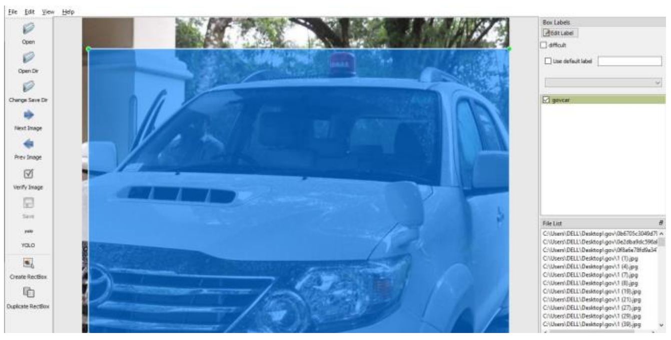
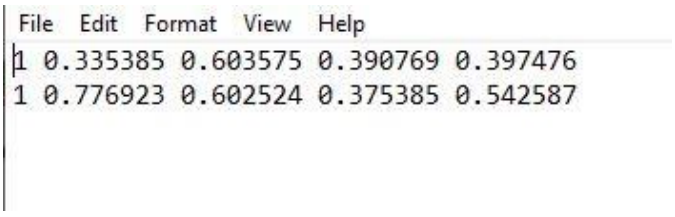


Fig -2: Window of LableImg with object bounding box

As the program window in fig-2 the file list box shows the input data and the class that we define will appear above the box. For each image we have drawn.

In YOLO labeling format, a .txt file with the same name is created for each image file in the same directory. Each .txt file contains the annotations for the corresponding image file, that is object class, object coordinates, height and width. <object-class> <x> <y> <width> <height> For each object, a new line is created.



```
1 0.335385 0.603575 0.390769 0.397476
1 0.776923 0.602524 0.375385 0.542587
```

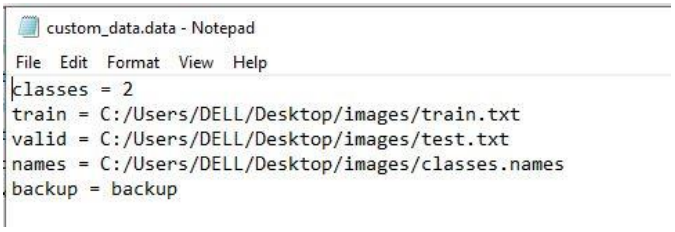
Fig -3: The bounding box coordinate with a class

III. INSTALLATION OF DARKNET FRAMEWORK

Darknet framework have different architecture, features than other deep learning framework and it's supports CPU and GPU computation [8]. Using the windows command prompt (cmd) as a command line to run darknet.

IV. PREPARING FILES FOR TRAINING

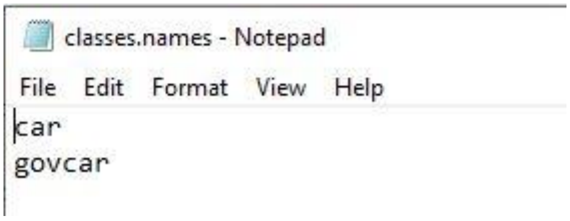
For custom dataset we created four files – custom_data.data, classes.names, train.txt, test.txt . Custom data. data file consist of following five lines.



```
classes = 2
train = C:/Users/DELL/Desktop/images/train.txt
valid = C:/Users/DELL/Desktop/images/test.txt
names = C:/Users/DELL/Desktop/images/classes.names
backup = backup
```

Fig -4: Custom_data.data file

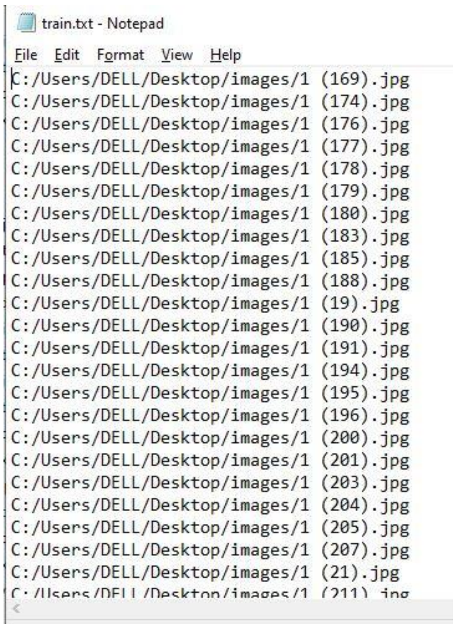
1st line contain number of classes (classes=2,car,govcar) , 2nd , 3rd and 4th line contain full path of train.txt, test.txt and classes.names files 5th line contain folder name where trained weights are saved after every 1000 iteration .Classes.names file consists of object names for training.



```
car
govcar
```

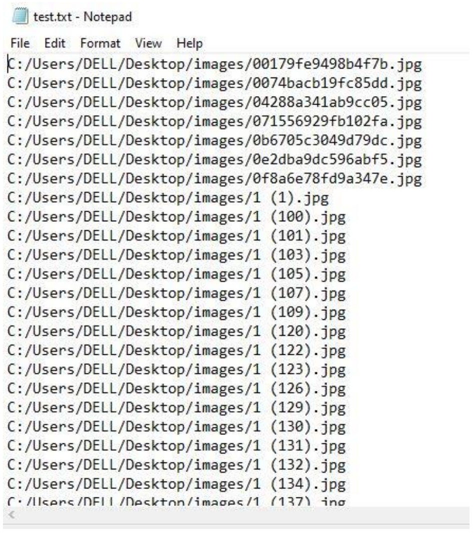
Fig -5: classes.names file with classes names

Train.txt and test.txt contain full path of all apropeate images.



```
C:/Users/DELL/Desktop/images/1 (169).jpg
C:/Users/DELL/Desktop/images/1 (174).jpg
C:/Users/DELL/Desktop/images/1 (176).jpg
C:/Users/DELL/Desktop/images/1 (177).jpg
C:/Users/DELL/Desktop/images/1 (178).jpg
C:/Users/DELL/Desktop/images/1 (179).jpg
C:/Users/DELL/Desktop/images/1 (180).jpg
C:/Users/DELL/Desktop/images/1 (183).jpg
C:/Users/DELL/Desktop/images/1 (185).jpg
C:/Users/DELL/Desktop/images/1 (188).jpg
C:/Users/DELL/Desktop/images/1 (19).jpg
C:/Users/DELL/Desktop/images/1 (190).jpg
C:/Users/DELL/Desktop/images/1 (191).jpg
C:/Users/DELL/Desktop/images/1 (194).jpg
C:/Users/DELL/Desktop/images/1 (195).jpg
C:/Users/DELL/Desktop/images/1 (196).jpg
C:/Users/DELL/Desktop/images/1 (200).jpg
C:/Users/DELL/Desktop/images/1 (201).jpg
C:/Users/DELL/Desktop/images/1 (203).jpg
C:/Users/DELL/Desktop/images/1 (204).jpg
C:/Users/DELL/Desktop/images/1 (205).jpg
C:/Users/DELL/Desktop/images/1 (207).jpg
C:/Users/DELL/Desktop/images/1 (21).jpg
C:/Users/DELL/Desktop/images/1 (211).jpg
```

Fig -6: train.txt with full path of images



```
C:/Users/DELL/Desktop/images/00179fe9498b4f7b.jpg
C:/Users/DELL/Desktop/images/0074bacb19fc85dd.jpg
C:/Users/DELL/Desktop/images/04288a341ab9cc05.jpg
C:/Users/DELL/Desktop/images/071556929fb102fa.jpg
C:/Users/DELL/Desktop/images/0b6705c3049d79dc.jpg
C:/Users/DELL/Desktop/images/0e2dba9dc596abf5.jpg
C:/Users/DELL/Desktop/images/0f8a6e78fd9a347e.jpg
C:/Users/DELL/Desktop/images/1 (1).jpg
C:/Users/DELL/Desktop/images/1 (100).jpg
C:/Users/DELL/Desktop/images/1 (101).jpg
C:/Users/DELL/Desktop/images/1 (103).jpg
C:/Users/DELL/Desktop/images/1 (105).jpg
C:/Users/DELL/Desktop/images/1 (107).jpg
C:/Users/DELL/Desktop/images/1 (109).jpg
C:/Users/DELL/Desktop/images/1 (120).jpg
C:/Users/DELL/Desktop/images/1 (122).jpg
C:/Users/DELL/Desktop/images/1 (123).jpg
C:/Users/DELL/Desktop/images/1 (126).jpg
C:/Users/DELL/Desktop/images/1 (129).jpg
C:/Users/DELL/Desktop/images/1 (130).jpg
C:/Users/DELL/Desktop/images/1 (131).jpg
C:/Users/DELL/Desktop/images/1 (132).jpg
C:/Users/DELL/Desktop/images/1 (134).jpg
C:/Users/DELL/Desktop/images/1 (137).jpg
```

Fig -7: train.txt with full path of images

V. SETTING UP CONFIGURATION FILE:

Configuration file consist of specific parameter that are used for testing and training such parameter includes Learning rate, Angle, Saturation, Exposure, Hue along with last three layer that describe the architecture of YOLO. for training in darknet framework we used batch='32' and subdivision = '16'. If there is memory overload error, then it is needed to decrease batch number and increase subdivisions.

a) UPDATE NUMBER OF ITERATIONS FOR TRAINING

For number of iteration we, needed to change *max_batches* and *steps* that are used for updating learning rate. *max_batches* is updated according to the number of classes.

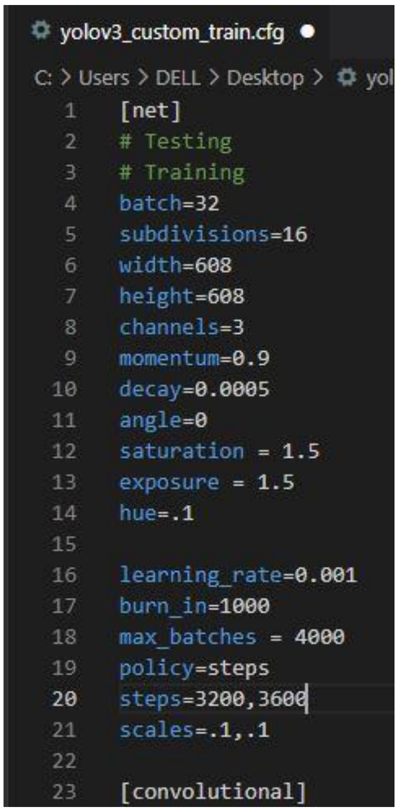
General equation is as following:

*max_batches = classes * 2000* (but not less than 4000)

max_batches=2*2000=4000

steps are calculated as 80% and 90% from *max_batches*.

Max_batches=320,360



```
yolov3_custom_train.cfg ●
C: > Users > DELL > Desktop > yol
[net]
# Testing
# Training
batch=32
subdivisions=16
width=608
height=608
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 4000
policy=steps
steps=3200,3600
scales=.1,.1

[convolutional]
```

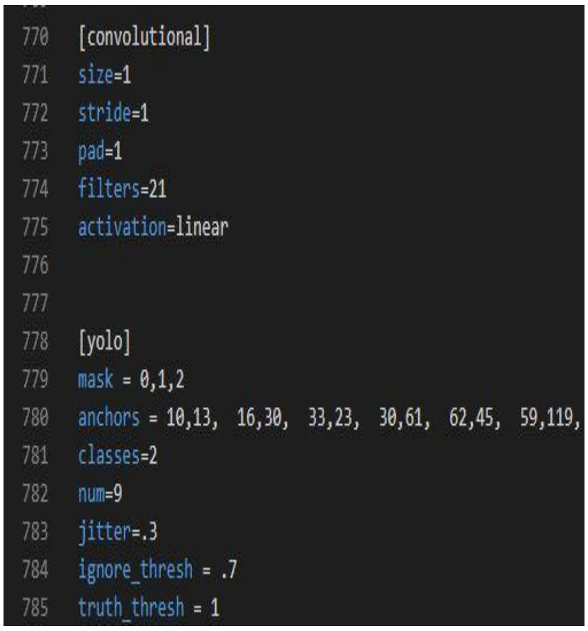
Fig -8: Updated max_batches and steps in cfg file

b) UPDATE NUMBER OF CLASSES IN 3 [YOLO] LAYERS AND FILTERS IN 3 [CONVOLUTIONAL] LAYERS

To set yolo architecture we needed to update number of *classes and filter* in every of three *[yolo]* layers in the end of the configuration files.

General equation to calculate number of filters as following:
filters = (classes + coordinates + 1) * masks

filters=(2+5)*3=21



```
[convolutional]
size=1
stride=1
pad=1
filters=21
activation=linear


[yolo]
mask = 0,1,2
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,
classes=2
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
```

Fig -9: Updated number of classes and filter for YOLO layer in cfg file

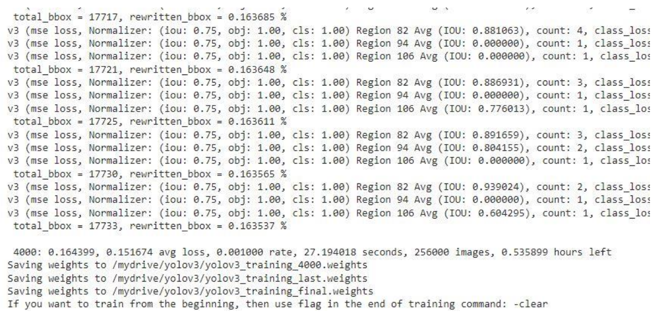
VI. START TRAINING PROCESS

Copy files created in section a in to darknet root directory, and start training by writing following commands in command prompt

./darknet detector train cfg/custom_data.data cfg/yolov3_custom_train.cfg weights/darknet53.conv.74

It is possible to stop training after 1000 iteration and continue later by using already saved weights. to continue training just specify at the end of command location of needed weights to continue training from.

./darknet detector train cfg/ts_data.data cfg/yolov3_ts_train.cfg backup/yolo-obj_1000.weights

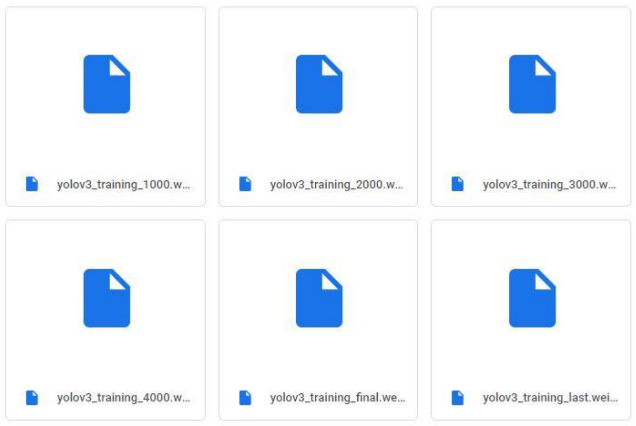


```
 total_bbox = 17717, rewritten_bbox = 0.163685 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.881063), count: 4, class_loss
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_los
 total_bbox = 17721, rewritten_bbox = 0.163648 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.886931), count: 3, class_loss
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.776013), count: 1, class_los
 total_bbox = 17725, rewritten_bbox = 0.163611 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.891659), count: 3, class_loss
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.804155), count: 2, class_loss
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_los
 total_bbox = 17730, rewritten_bbox = 0.163565 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.939024), count: 2, class_loss
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.604295), count: 1, class_los
 total_bbox = 17733, rewritten_bbox = 0.163537 %

 4000: 0.164399, 0.151674 avg loss, 0.001000 rate, 27.194018 seconds, 256000 images, 0.535899 hours left
Saving weights to /mydrive/yolov3/yolov3_training_4000.weights
Saving weights to /mydrive/yolov3/yolov3_training_last.weights
Saving weights to /mydrive/yolov3/yolov3_training_final.weights
If you want to train from the beginning, then use flag in the end of training command: -clear
```

Fig -10: Completion of training process

When should we stop training? our training image is used to take some time to train dataset output, but we could know where the best time is to stop training by average loss, the lower is better. It either can stop at around below 1% loss (red line) for the big dataset or it reaches the maximum iterations. After every 1000 iteration training weights are saved in backup folder as showed in below fig-11.



yolov3_training_1000.w...
yolov3_training_2000.w...
yolov3_training_3000.w...
yolov3_training_4000.w...
yolov3_training_final.we...
yolov3_training_last.wei...

Fig -11: Saved weight backup folder

VII. TEST TRAINED WEIGHTS

DNN (deep neural network) module is a module of OpenCV. In this section, we created a Python script that can be used OpenCV and Flask library. Moreover, we can see resulted videos on webpage.

Fig-12 is result of the public sector and official government sector car detecting in one frame with the 98% and 99% accuracy that means the system strongly confirms the custom object.

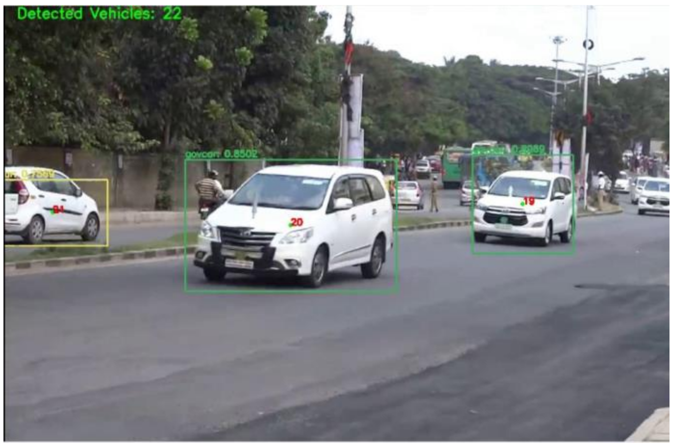


Fig -12: Vehicle classification of public sector car and state government car

Above fig-12 shows vehicle classification of public sector car and state government car with the accuracy of 75% and 85%.

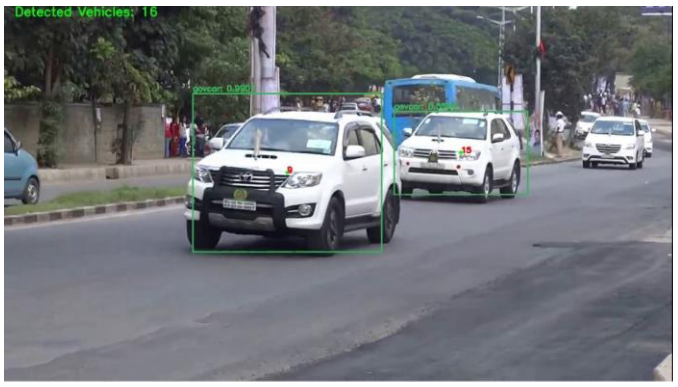


Fig -13: Image with state government car with 99% and 95% accuracy

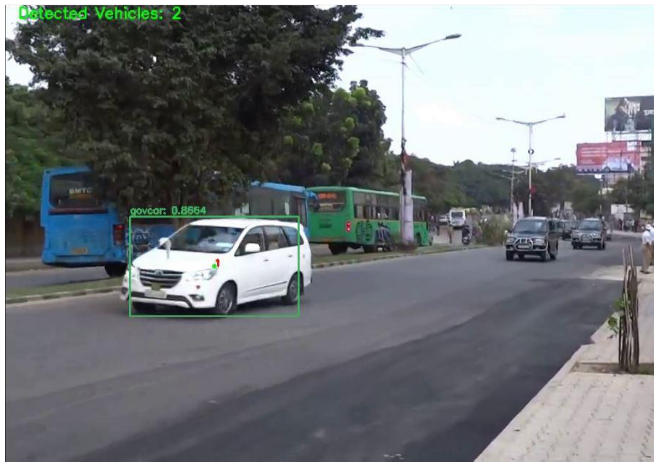


Fig -14: Detect state government car with 85% accuracy

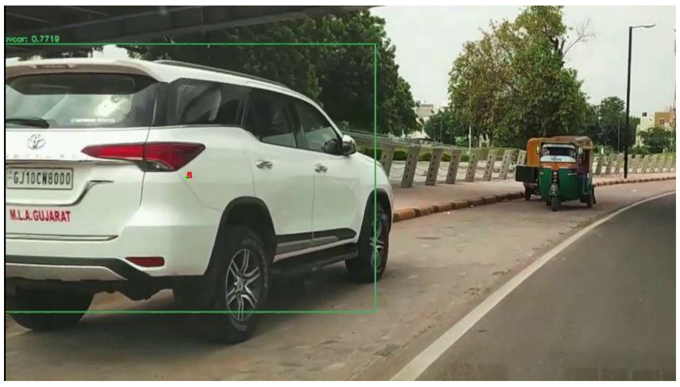


Fig -15: Detect state government car with 77% accuracy

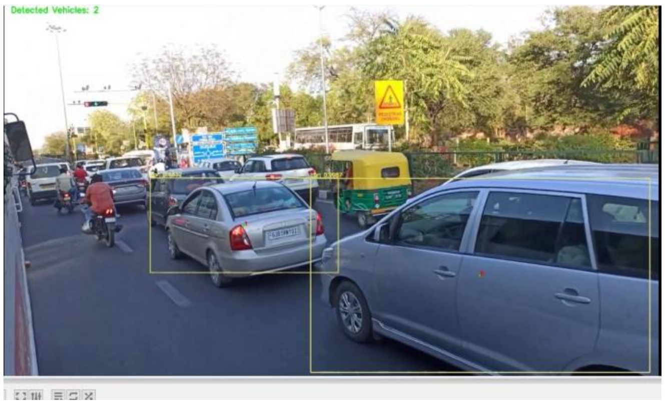


Fig -16: Detect public sector car 86% and 99% accuracy

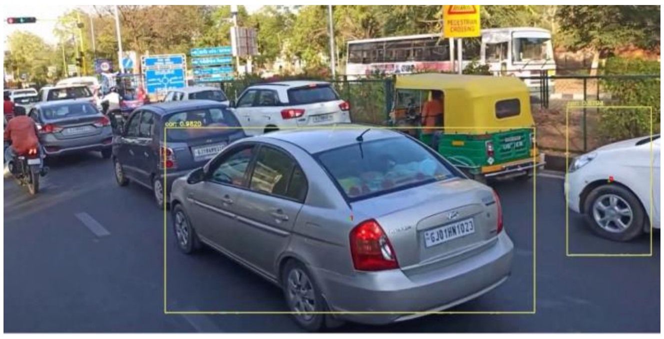


Fig -17: Detect public sector car 98% and 87% accuracy

The accuracy of the system also depends on the camera is placed. From fig-10 we got average loss 0.151674 after completing trained process.

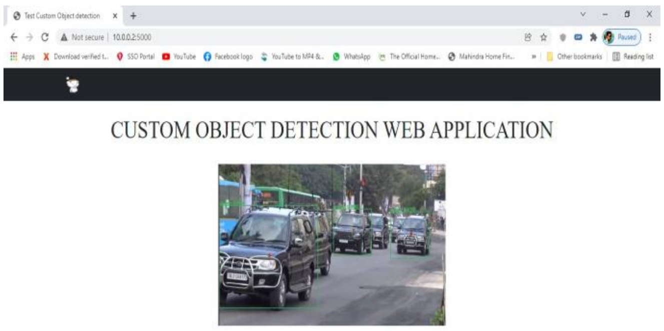


Fig -18: Object detection video streaming on web page

Fig -19: Object detection video streaming on web page

4. CONCLUSION AND FUTURE WORK

Our custom dataset can detect the public sector car and official state government car correctly with high accuracy. Also, we got 0.151674 average loss of trained process. Accuracy can decrease in case might happen from the test frame that has a low resolution than 416x416 or unsuitable size such of the picture. Therefore, all images in the dataset need to have more quality. The accuracy of object detection may also depend on which angle camera is fixed. This paper shows a complex scenario with analysis of each classification for typical Indian roads with better accuracy from yolov3 custom trained model.

As a part of future work, it can be further extended to classify official state government car using numberplate, color, speed calculation and direction of movement for each vehicle. The accuracy can be improved by considering the all-possible vehicle category.

REFERENCES

[1] Sergios Theodoridis, Neural Networks and Deep Learning 2015.

[2] Rahul Dutt Sharma, Shubh Lakshmi Agrwal, Sandeep K. Gupta, Anil Prajapati, " optimized dynamic background substraction technique for moving object detection and tracking" 2017 2nd International Conference on Telecommunication and Networks (TEL-NET 2017)R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[3] Syed Mazhar Abbas, Dr. Shailendra Narayan Singh, "region-based object detection and classification using fast r-cnn" International Conference on "Computational Intelligence and Communication Technology" (CICT 2018)

[4] Yongbon Koov, Chayoung You, SungHoon Kim, "OpenCL-Darknet: An OpenCL Implementation for Object Detection" 2018 IEEE International Conference on Big Data and Smart Computing

[5] Priyanka Malhotra, Ekansh Garg, " object detection techniques: a comparison" IEEE 7th International Conference on Smart Structures and Systems ICSSS 2020

[6] Ajeet Ram Pathak, Manjusha pandey, siddhahrth rautaray. "application of deep learning for object detection" International Conference on Computational Intelligence and Data Science (ICCIDS 2018)

[7] Chun Ju Huang, Ting-Wei Chen,Yu-Cheng Fan. " Chip and System Design of Real Time Object Detection Based on LS-R-YOLO Network" 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)

[8] https://pjreddie.com/darknet/install/

[9] Sushmitha.S, Neelima Satheesh, Kanchana .V ,"Multiple Car Detection, Recognition and Tracking in Traffic". 2020 IEEE International Conference for Emerging Technology (INCET*)*

[10] Mirthubashini J, Santhi V, "Video Based Vehicle Counting Using Deep Learning Algorithms", IEEE 2020 6th International Conference on Advanced Computing & Communication Systems (ICACCS)

[11] Ahmet Ali Süzen , Burhan Duman , Betül Şen , "Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN ", IEEE August 02,2020

[12] Eduardo Jr Piedad, Tuan-Tang Le, Kimberly Aying, Fhenyl Kristel Pama, Ianny Tabale." Vehicle Count System based on Time Interval Image Capture Method and Deep Learning Mask R-CNN", ", 2019 IEEE Region 10 Conference (TENCON 2019)

BIOGRAPHIES

Jaivik D. Rathod pursued Bachelors of Engineering in specialization of Electronics and Communication from Vishwakarma Government Engineering College in 2020. He is currently pursuing his Master of Engineering in specialization of VLSI and signal processing from VGEC. His main research work focus on Artificial Intelligence - Deep Neural Network based on custom object detection technique.

Dr. Kiran Trivedi currently working as professor in the department of Electronics and Communication at Vishwakarma Government Engineering College. He has 22 years work experience in respective field and also working in the area of Artificial Intelligence and Wireless Communication.