International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 07 | July 2021 www.irjet.net p-ISSN: 2395-0072

Speed Limit Sign Detection using Deep Learning

Yogesh Valeja[1], Shubham Pathare[2], Dipen Patel[3], Prof. Mohandas Pawar[4]

[1]UG Student, MIT ADT University, Pune
[2]UG Student, MIT ADT University, Pune
[3]UG Student, MIT ADT University, Pune
[4]Asst. Prof., MIT ADT University, Pune

Abstract - *In 21'st century we have witnessed a vast development in the field of Autonomous Self driving cars and automation in various auto industries. It is not far when we can see fully self-driving cars traveling from destination A to destination B without any human interference, and we can have a gist of that when Elon Musk introduced the first semi-autonomous TESLA vehicle. In this ever-fast-growing autonomous industry, we have tried to contribute to this industry by integrating OpenCV with python to test our first autonomous model in the CARLA environment which will detect speed signs and control the speed of the vehicle accordingly. We will use the YOLO object detection algorithm for the purpose of Speed sign detection and train our model on the same, we have gathered various images of Speed signs from the CARLA environment which will be used as a strong dataset for the training purpose.*

Key Words*: *Self-Driving Simulator, Object Detection, Speed Limit Sign Detection, Carla, Yolo, Deep Learning

1.INTRODUCTION

There is a high hype among individuals about autonomous vehicles in the past several years and is constantly building up. There has been a lot of research on self-driving cars, the technology that enables society to remove the driver from the front seat, not to mention, it's crucial to understand how these machines on wheels work.

Self-driving cars work on five major components:

1) **Computer Vision:** the eyes of the car; how it 'sees' the road this is done through various types of sensors and not just cameras.

2) **Sensor Fusion:** the process of combining information from other sources just like how our brain combines auditory and visual data to make decisions.

3) **Localization:** figuring out EXACTLY (down to single-digit centimeters) where the car is on the road, to a very high degree of accuracy

4) **Path Planning:** taking all the information about our surroundings and making a decision about which way we need to go to get to our destination.

5) **Control:** the process of physically moving the car based on decisions made in "path planning", by constantly adjusting the steering, gas, and brakes.

In 2016, the Society of Automotive Engineers (SAE) released documentation, SAEJ3016, defining six different levels of automation for self-driving cars. In the year 2021, a detailed revision of the SAEJ3016 document was carried out that adds a few new terms clarifying the concepts and defining responsibility in the event of a failure.

Level 0: No drivers assistance or automation

Level 1: Either Auto steering OR accelerating in limited conditions and areas

Level 2: Auto steering and accelerating in limited conditions and areas

Level 3: Partial automation in limited conditions and areas

Level 4: Complete automation in limited conditions and areas.

Level 5: A level 5 vehicle can carry out all the driving tasks in all the conditions and areas. The vehicle is solely responsible for all the tasks and carrying them out safely.

For the following project, we are trying to detect the traffic speed signs and letting the vehicle take its own decision and warn the driver according to our specific application. As the implementation of this project Physically can be expensive as well as risky we have tried to create a virtual city environment with all the factors like traffic lights, speed signboards, pedestrians, and other vehicles using a virtual simulator called Carla which is built on top of Unreal 4 engine and gives a strong base to run our research and test our thesis virtually without harming any human life and is very much cost-effective.

When we talk about the evergrowing advancement in autonomous vehicles the main concern we encounter is the vehicle making its own decision according to the surrounding environment it senses. Our project Speed Sign Detection and Vehicle Control follow a similar approach, in a nutshell, it collects the surrounding environment with the camera installed on a hood and detects for speed signboards on the road, once it captures any of the max Speed signs it compares with its own Speed and takes decision accordingly which are two main components first being Warning Driver if he exceeds max Speed set by the road and Second Being

lowering Down acceleration automatically if it exceeds the max Speed set for that road.

2. LITERATURE SURVEY

To include an exhaustive examination of Speed Sign Detection, we studied papers from various journals, conferences, and acquired data. The following is how the various papers are organized:

They discussed the basic chronology leading to autonomous car production. This paper explores the historical context, recent patterns and innovations, and semi-autonomous cars' predictable future for public use [1].

They discussed the R-CNN, use the proposed Range of Interest (ROIs) methods to set possible boundary boxes and then run a classification system for each of these suggested boxes.[2]

They suggested the Faster R-CNN to achieve greater precision and higher latencies overall than making a sequential area pipeline [3]

Furthermore, the unified object detection model YOLO was developed. It can be educated directly on the full images in them. YOLO is not trained on a set of detection rules but instead trained on a set of non-negative output losses to detect scenes [4].

They proposed a method that effectively detects and monitors one or more moving objects in a variable context simultaneously. The algorithm's key benefit was that it did not rely on any previous environmental information [5].

They expressed the importance of deep learning applications in image classification, object detection and face identification. Experimental evidence shows that deep learning technology is an efficient method to move from the human-made function that relies on the drive of experience to the learning that relies on the data drive. Extensive data is the foundation of deep learning performance, large data, and the rocket's fuel for deep learning [6].

The degenerative model fed by the degraded image has been educated. The results showed that the model enhanced the overall classifications of objects. The model trained with reduced data has higher generalization ability, more significant potential, and more robustness [7].

A method for improving YOLO v2's network structure and obtained the YOLO-R network model. In pedestrian identification, they have achieved successful outcomes [8].

A process for identifying and detecting welding joints used on the production line of automobile door panels by using the YOLO algorithm. It was suggested that a detector for detecting the position of joints in solder [9].

A barrier algorithm combines YOLO and light field camera, which would categorize objects into various categories and mark them in the input image [10].

A deep fusion practical method for detecting objects in remote sensing images with high resolution. This method consisted of three main steps: the generation of candidate regions, the fine-tuning extraction of in-depth features, and the deep feature classification of SVM [11].

They proposed Tiny SSD, a single shot detection deep convolutional neural network. TINY SSD aimed to ease real-time embedded object detection. It comprises of greatly enhanced layers comprising of non-uniform Fire subnetwork and a stack of non-uniform subnetwork of SSD based auxiliary convolutional feature layers. The best feature of Tiny SSD is its size of 2.3 MB which is even smaller than Tiny YOLO.[12]

They suggested a single layer multilayer perceptron network YOLO for object detection. YOLO can predict boundaries, thanks to its ability to detect object boundaries [13].

They proposed YOLO v1 neural object detection network by modified loss function and added spatial pyramid pooling layer and initial module with coevolutionary kernels [14].

They discussed YOLOv3. YOLOv3 was one of the best modifications that had been done to an object detection system since the introduction of Darknet 53. This modified update was received very well among the critics and other industrial professionals[15]

3.PROPOSED SYSTEM

The problem statement we are working on is the detection of Speed traffic signs and accordingly control the speed of our vehicle, which will be an advancement for the automation of the cars we are expecting this decade. As told, we have two main Modules which are Detection of Speed Signs as we drive our vehicle in streets and the environment which we need to set up for running our car, as this is the first working prototype, we designed a virtual environment using **CARLA** as deploying the model physically is quite expensive and risky.

For setting up an Object detection Environment for detecting Speed Signs on the road we need few Prerequisites to complete before moving forward.

3.1 Configuring OpenCV with CUDA

CUDA is a parallel computing platform and programming model that makes using a GPU for general-purpose

computing simple and elegant. For detecting Speed signs on the road we need quite a good and powerful GPU to detect high speed and for that reason, we are going to use the YOLO object detection module for which CUDA is a requirement. We need to first of all configure Cuda for its extensive GPU computation with OpenCV. First, we downloaded the latest version for OpenCV. Then we need to download the contribute files for extra modules of OpenCV. Now to configure CUDA and OpenCV we need to download CMAKE and Visual Studio for building OpenCV. To download CUDA we need to download a specific version according to the graphics Card we are having in our System. After you get to know what graphics card(NVIDIA) we have installed. Go to the Wikipedia page and find the corresponding architecture and version of CUDA most appropriate according to our requirements. After a successful installation of CUDA and cudnn we need to make OpenCV with cmake and we have successfully configured OpenCV with CUDA.

3.2 Configuring YOLO

Now after having installed OpenCV with CUDA support in place, we need to configure the state-of-the-art object detection YOLO module with darknet. Darknet is mainly for Object Detection and has a different architecture, features than other deep learning frameworks. It is faster than many other NN architectures and approaches like FasterRCNN etc. Darknet architecture & YOLO is a specialized framework, and they are on top of their game in speed and accuracy. YOLO can run on CPU but you get 500 times more speed on GPU as it leverages CUDA and cuDNN. Now after building up darknet we need a weights file to train our speed sign detection module which will complete our object detection process.

3.3 Acquiring Speed Limit Sign Dataset

Getting Speed traffic signs for the real world is quite easy as there is a lot of traffic dataset available in public to train on, but acquiring a dataset of speed signs for specific Carla was quite a hectic job as it wasn't available in public. Therefore for the same we first screen recorded the screen by continuously driving around the Carla environment and recording each and every speed sign driving past the traffic signs 3-4 times. Having this 30min recording we used cap.get(cv2.CAP_PROP_FRAME_COUNT) to know how many frames have been recorded and can be converted to an image, if we have taken all the frames it was quite an entire video with duplicate images therefore we ran an python script to convert each(every 30th) video frame to image using Opencv (cv2.imwrite) to extract image to a specified folder having quite a huge dataset for our training model. These were all unlabelled images and were to be labeled according to their class , in here Carla city had maily 3 classes i.e 3 speed signs which were 30km/h, 60km/h, and 90km/h. For labeling these images we used labellmg tool, this tool is quite simple and provide a simple interface where we needed to draw a box around our target, which we need to detect and train our model around this resulted in getting the labeled image itself with a text file having information in the format: <object_class> <x_center> <y_center> <width> <height> Now having our labeled images in Place we are ready to train our Model.

3.4 Network Modification

The next step here was to modify our network accordingly to detect our custom speed signs from Carla accurately. For the training purpose, we have used tiny v3i.e., the latest and fastest upgrade. While using a convolutional network it simultaneously predicts the part of the image and class probabilities while in the case of YOLO it taoptimizestire simple image and trains on the full image instead of making multiple bounding and parting the image and thus optimises the performance for detection. This entire Yolo model takes up smaller 16 layers instead of traditional 22 deep Convolutional Neural Networks. This tiny v3 yolo architecture is been inspired by GoogleNet architecture. Understanding the network modification of Yolo architecture for our discussed project is illustrated below as: We have taken a batch of 64 having 16 subdivisions and the height and width of the training image dataset is 608 X 608, with 3 channels and 0.9 momentum having 0.0005 decay rate , learning rate for training is set to 0.001 with max_batches of 6000 having 4800 , 5400 steps giving exposure of 1.5 and saturation to 1.

3.5 Training Process

For faster training and efficient results instead of training on the local system's CPU, we have used Google's advanced Google Colab which allows us to use their cloud services which allows us to access their high-end resources such as we can use their Tesla K80 GPU absolutely free. Now here we just executed the darknet training command for Yolo tiny v3 which started the training and gave us a Loss graph along with the number of iteration to see how our model is doing i.e whether it is underfitting or overfitting and thus creating a backup weights file after every 1000 iterations. Below given is the table to summarize how our 6 weights file created, performed and saved after every 1000 iterations.

Weights files comparison			
Weights Files	Avg Loss	maP	IoU
1000 iterations	0.2612	38.11%	28.25%
2000 iterations	0.1559	82.80%	58.22%
3000 iterations	0.1013	89.49%	66.26%
4000 iterations	0.0618	90.93%	72.18%
5000 iterations	0.0482	91.62%	75.39%
6000 iterations	0.0521	91.56%	75.36%

Table 1: Table comparing all the weights files created after every 1000 iterations

After seeing the results from the Table 1, we can conclude that the weight file created after 5000 iterations has the highest Average Precision and Intersection over union and low Average Loss. We will test all these weights files and calculate the precision-recall matrices and F1 score and accordingly use the most suitable one.

3.6 CARLA

CARLA is an open-source simulator developed for autonomous driving research. CARLA has been developed from the ground up to support the development, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely. The simulation platform supports flexible specification of sensor suites and environmental conditions.

For the current project, we have used CARLA's 0.8.2 version which has been considered as the most stable version for windows. The best thing for choosing the Carla simulator was it can be configured by connecting an external python script and thus can be entirely customized and modified according to our requirement, in our case to implement Yolo's object detection module. Also for the research purpose, we have been supported by python's most efficient GUI i.e pygame module where the user can control his vehicle manually which can give the user a feeling of a perfect real environment where he/she is driving a car in the streets. Along with the manual feature CARLA also equipped us with a lot of sensors like a camera sensor, impact sensor, and so on. We will be using a Camera sensor to read the images captured by the camera sensor to our training model sending after every 5fps and replacing the previously saved image.

Fig 1: Images captured by Camera Sensor installed on hood of Camera on CARLA Simulator

Fig 1 shows the image captured by our camera sensor placed on the hood of the car to the training model. To run our trained Yolo model this image is paced to the training function and is checked if the received scene contains any of our trained classes. If yes then the class to which the image is detected is embedded on the screen and it is set to that speed and stays on the top of the screen. This set speed acts as a flag to perform two main tasks which can be performed which is discussed in the next section.

Fig 2: Sign Board Detected by passing each frame through Yolo module to Detect Speed Sign

3.7 Applications

The main objective of the project as discussed was to detect the speed sign on the road and accordingly our system should take decisions accordingly. After detecting the maximum speed limit allowed signboards there are mainly two main applications been deployed: One being the **Warning System** and the other is **Vehicle Speed Controlling System**. We have discussed both of the applications briefly below.

3.7.1 Warning System

If the user sets the flag as "warning" during executing the manual control file the warning system will get triggered. The simple function of warning system that can be stated is that when a speed sign on a road is detected, it is checked with the current speed of the vehicle if the current speed of the vehicle exceeds the speed limit detected by the system, it gives a warning message suggesting "Reduce your Speed" in a red indication warning the driver to reduce his speed.

Fig 3: Warning Message displayed when Speed of Vehicle exceeds the threshold Speed Detected by the vehicle

Once the driver reduces his speed and the speed of the vehicle becomes less than the speed detected before no message is displayed and the vehicle moves with no interruption with a green color stating the current speed of the vehicle.

3.7.2 Vehicle Speed Control System

As the name suggests the function of the application when the user selects control application(-app Control) while executing the manual_control.py, the task of this application is to simply reduce the speed of the vehicle if it exceeds the speed beyond the detected sign. This is done by simply, the vehicle current speed is constantly compared to detecting speed if it exceeds the threshold already detected by Yolo then it triggers a keyboard function which throttles the acceleration and disables the key which is set to acceleration until the speed is reduced till the threshold speed and a message in red is displayed stating that the "Speed is Reduced".

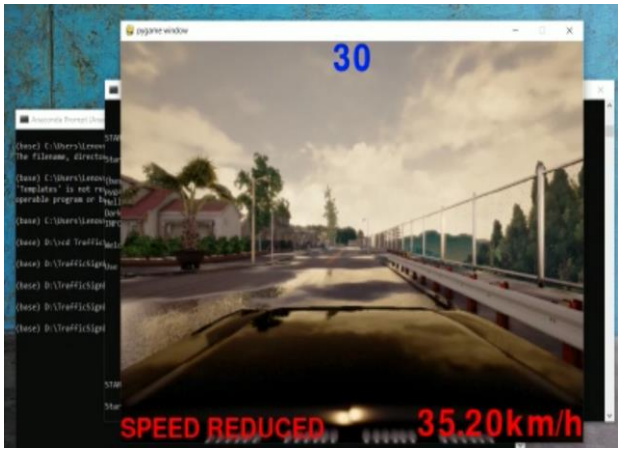

Fig 4: Vehicle Control System Reducing the Speed Automatically when Speed exceeds the Threshold Speed

4. RESULTS

A test set of 1,180 photos retrieved from CARLA in a single session was constructed to test the trained network. The test set included the same number of photos with and without tagged objects as the training set. Apart from the mAP and the IoU, three more metrics were analyzed.

In order to select the weights file that would be used within the system. Precision, recall, and the F1-Score were the three metrics. The ratio of successfully predicted positive observations to total expected positive observations is the precision measure. The recall statistic represents the percentage of positive observations that have already been properly forecasted as positive. Finally, the F1-Score is a metric that is used to find a balance between precision and recall, or the weighted average of these two metrics.

Weights Files	30 km/h	60 km/h	90 km/h
1000 Iterations	41%	34%	38%
2000 Iterations	73%	81%	88%
3000 Iterations	83%	85%	92%
4000 Iterations	85%	86%	96%
5000 Iterations	86%	88%	100%
6000 Iterations	84%	87%	98%

Table 2: Performance values on the test phase

When we compare these findings to the ones acquired during the validation test, we can observe that the weights file values on the 5000th iteration are the highest. This explains why they were chosen to be included in the object-detection system. The fact that the files with the greatest weights are not the ones acquired in the previous iterations suggests that between iterations 5000 and 6000, an overfitting phenomenon occurs.

5. CONCLUSIONS

In this study, a usable framework for training Speed Sign Detection vehicle using YOLO algorithm for detection purposes implemented on CARLA environment has been constructed. Although an effective policy was not achieved after the first round of training, many insights about how to improve these results in the future have been obtained. The learning framework created provides the opportunity to easily apply these insights in the future.

The next thing to work on would be to test more values for the hyperparameters and compare the results of each, which again could be much better done using larger hardware than the gaming laptop used in this study since training time could be drastically reduced by having more workers in parallel. Smaller values for learning rate should be explored since we saw decreasing average rewards in the early period of this training session, which could be an indication that the learning rate was too large.

ACKNOWLEDGEMENT

First of all, we would like to thank our project guide Prof. Mohandas V. Pawar for giving us the courage, guidance and suggestions for doing this major project. We also express our gratitude towards Dr. Rajneesh Kaur Sachdeo, HOD CSE and Dr. Kishore Ravande, Principal MITSOE sir for their support and guidance. We are thankful to MIT School of Engineering-MIT ADT University, Pune for providing all resources and valuable information required about data mining techniques for our project the process of analyzing and doing research on the valuable inputs helped us to explore knowledge, was a continuous source of inspiration and a unique experience.

REFERENCES

[1] Keshav Bimbraw "Autonomous Cars: Past, Present and Future"

[2] J.R.R. Uijling, K.E.A. van de Sande, T. Gever, and A.W.M. Smeulders2 "Selective Search for Object Recognition" in International Journal of Computer Vision · September 2013

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" arXiv:1506.01497v3 [cs.CV] 6 Jan 2016

[4] Joseph Redmon,Santosh Divvala,Ross Girshick,Ali Farhadi." You Only Look Once: Unified, Real-Time Object Detection" [J].2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2016:779788.

[5] Kumar S. Ray and Soma Chakraborty "An Efficient Approach for Object Detection and Tracking of Objects in a Video with Variable Background" 2017 arXiv:1706.02672

[6] Xinyi Zhou, Wei Gong, WenLong Fu, Fengtong Du "Application of Deep Learning in Object Detection" 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)

[7] Chengji Liu ,Yufan Tao ,Jiawei Liang ,Kai Li1 ,Yihang Chen," Object Detection Based on YOLO Network" 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC 2018)

[8] Wenbo Lan, Jianwu Dang, Yangping Wang and Song Wang," Pedestrian Detection Based on YOLO Network Model" 978-1-5386-60751/18/$31.00 ©2018 IEEE

[9] Zhimin Mo1, Liding Chen1, Wenjing You1" Identification and Detection of Automotive Door Panel Solder Joints based on YOLO" 978-1-72810106-4/19$31.00 ©2019 IEEE

[10] Rumin Zhang, Yifeng Yang," An Algorithm for Obstacle Detection based on YOLO and Light Filed Camera", 2018 Twelfth International Conference on Sensing Technology (ICST)

[11] Eric Ke Wang, Yueping Li, Zhe Nie, Juntao Yu, Zuodong Liang, Xun Zhang, Siu Ming Yiu "Deep Fusion Feature Based Object Detection Method for High Resolution Optical Remote Sensing Images"

[12] Womg A, Shafiee MJ, Li F, Chwyl B. Tiny SSD: a tiny singleshot detection deep convolutional neural network for real-time embedded object detection. In: 2018 15th conference on computer and robot vision (CRV). IEEE; 2018, p. 95101

[13] Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam "Real-Time Object Detection with Yolo" International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-3S, February 2019

[14] Tanvir Ahmad, Yinglong Ma, Muhammad Yahya, Belal Ahmad, Shah Nazir, and Amin ul Haq "Object Detection using Modified YOLO Neural Network" Scientific Programming, vol. 2020

[15] Huang YQ, Zheng JC, Sun SD, Yang CF, Liu J. Optimized YOLOv3 algorithm and its application in traffic flow detections. Appl Sci. 2020;10(9):3079.