Self-driving Cars Using CNN and Q-learning

Syed Owais Ali Chishti
Dept. of Computer Science
FAST National University
Peshawar, Pakistan
p146011@nu.edu.pk

Sana Riaz
Dept. of Computer Science
FAST National University
Peshawar, Pakistan
p146114@nu.edu.pk

Muhammad Bilal Zaib
Dept. of Computer Science
FAST National University
Peshawar, Pakistan
p146099@nu.edu.pk

Mohammad Nauman
Dept. of Computer Science
FAST National University
Peshawar, Pakistan
mohammad.nauman@nu.edu.pk

***Abstract*—DrivingMatter is an experiment carried out to understand the deeper side of an autonomous car. In 1900s, idea was to drive car on Moon from Earth. This was initial motivation which grew from there and now expanding to complex system of roads in the real world.**

A book-sized Raspberry Pi based autonomous car is built to carry out the experiment on hardware. Software side was accomplished by developing a Python based library for controlling and communicating with car over a network or locally within the car.

For environment learning two methodologies are practiced; *Supervised learning:* Drove the car on an environment/road and collected 3,000+ data-points. Based on this a CNN model was trained which achieved 73% test 89% train accuracy. *Reinforcement learning:* Car is trained for three different road signs; Stop, No left, and Traffic light using DQN with existing CNN model. These road signs are detected in the environment using OpenCV cascade classifiers.

***Index Terms*—self-driving car, deep learning, deep Q-Network, robotics**

I. Introduction

An autonomous or self-driving car is a vehicle that must be able to navigate, to certify as autonomous, without human input to a pre-established destination over a path that have not been adapted for its use. It makes its own driving decisions, able to cope with all situations and continuously keep learning from its environment and decisions on that environment to maximize its output over time. A fully automated car is able to handle all driving tasks in all driving mode and under all environmental conditions, just like human drivers.

Self-driving cars development is uprising rapidly in automotive industry. They are increasingly catching attention worldwide because prospective of this technology is clear as it will dramatically change transportation by minimizing traffic jam, increasing efficiency and allowing faster speed. Autonomous cars are predicted to increase traffic flow and lower fuel consumption which will reduce contamination in urban areas by improving driving and significantly reduce needs for parking space. Self-driving cars also have potential for minimizing road crimes by recording everything that happens in and around them. In addition, autonomous cars will speed up people and freight transportation, as well as increase the security, specifically a significant reduction in traffic collisions by reducing the human error.

Since governments, industries, and research institutions are investing vast amounts of both human-time and money, we can make an idea of the interest that autonomous driving is raising and pushing its limits. Fully autonomous vehicles have become a reality, as for example by November 2017, it has revealed that Waymo's self-driving cars had driven more than 4 million miles on public roads [1]. The Institute of Electrical and Electronics Engineers (IEEE) predicts that driver-less cars will account for up to 75% of vehicles on the road by the year 2040 [2].

Despite the various benefits to increased vehicle automation, there is a lot more work to be done before self-driving cars are ready for the mainstream. As the self-driving car is a collection of networked computers wirelessly connected to the outside world, the most daunting challenge is keeping the systems safe from intruders i.e. Cyber Security. Similarly disputes concerning liability, driving safely despite unclear lane markings or recognize traffic lights that are not working. Other obstacles could be missing driver experience in potentially dangerous situations, ethical problems in situations where an autonomous car's software is forced during an unavoidable crash to choose between multiple harmful courses of action, and possibly insufficient adaptation to respond on spoken commands or hand signals from pedestrians or highway safety employees. Autonomous cars today are intelligent enough to drive around the city with very low chances of crashing, that is great but there are some limitation which they are bounded with, is that they are limited to GPS and ground truth of the Earth i.e. Google Maps or similar. If we removed that which is common in mountainous, forest or new area then car is zero and then you have to drive yourself.

In this paper, the main focus is on the learning process of a self-driving car irrespective of how the hardware is established. We have specified a road environment and will train the car using deep learning by extracting road features with some real world obstacle/constraints and let the car learn whole state by itself from zero to pro i.e. instead of extracting the useful features beforehand and training a model on those features, the model will extract features itself from the whole state and will learn on what is useful from those feature. Training of the car is also done on basis of rewards associated with a given state using the same model to maximize those rewards.

FAST National University, Peshawar, Pakistan

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

II. Background

The history of self-driving cars goes beyond early 1920s when experiments were started being conducting to turn the fantasy of self-driving cars concept into reality. Promising trials took place in 1950s; first self-sufficient and truly autonomous car appeared in 1980s. There has been a strong uprising tendency in the autonomous driving technologies since in 2004 DARPA introduced its first grand challenge [3]. Currently, there are many projects on autonomous car are going on industry level that include Google self-driving car (now Waymo), Tesla, General Motors, Ford, Volkswagen, Audi and Volvo etc., who have begun testing autonomous/self-driving vehicles.

There are significant number of projects going on self-driving car. For instance, Udacity is building open source self-driving car which provides with learning models for example many different neural networks are trained to predict steering angle and Robot OS is used as a middle-ware to interact with these models. They have also collected over 10 hours of driving data from different real-time traffic scenarios and labeled it to make annotated datasets for training of autonomous vehicles. For detecting cars, pedestrians and traffic lights Yolo models are trained against the annotated Udacity datasets [4].

III. Literature Review

Over the past decades, many researches generally focus on road lane detection as they contain simple geometric features and can be extracted with ease from background [5].

Following the revolutionized CNNs pattern recognition [6], there is a substantial amount of research on the self-driving car using deep learning approach. Numerous research efforts are being done recently in applying deep learning techniques in all low-level/mid-level computer vision tasks of robot navigation trends in which the low computational cost of implemented CNN makes real-time processing easily achievable [7]. Huval et al. [8] proposes a method to predict two end points of a local lane segment by regression in a sliding window using CNN and combines it with RNN to detect lane boundaries. Similarly, research of Hadsell et al. [9], which propose a deep hierarchical network to extract important and meaningful features from an input image, and the features are used to train a real-time classifier to predict traversability. They incorporated a self-supervised learning framework capable of accurately classifying complex terrains at distances from 5 to over 100m away from the platform, thus remarkably outstripping path-planning.

Chen et al. [10] present an approach which instead of learning the mapping from pixels, first estimates a several human key perception indicators which are directly related to affordance measures such as the distance to surrounding cars.

IV. System Application and Environment

Training the car in the real-time environment like motorway or a road with heavy traffic is a very risky and cost-driven task. To tackle such situation a small Raspberry Pi based car, and a custom environment was established, so that training can be carried out feasibly. The environment consist of a map track. Roads are based on two kinds, one simply rounded road and one with the forks, with the direction markers, traffic lights and other signs which contributes to the decision of action which car takes. This Raspberry Pi based small car and defined specific environment makes our own system. Once the car is trained on this environment, it can easily be scaled to any other bigger environment with an expensive car.

V. Dataset Acquisition

In our case, the external feedback discussed, is the dataset consist of the knowledge of car's environment state on each action taken by a real driver (camera's images with the action taken on that state), this gives knowledge to the learning function or model to predict the best action for the new state in future. Dataset is collected by driving car over track. The dataset consists of images, taken from the three cameras of the car which are stored in a folder and dataset.csv file which contains action taken on a state, path to the three images of left, right and center camera stored in /images folder, and the time action was taken. So, the basic structure of the dataset is as following:

```
/Dataset-2017-12-01
├── dataset.csv
└── /images
    ├── 1.jpg
    ├── ..
    ├── ..
    └── *.jpg
```

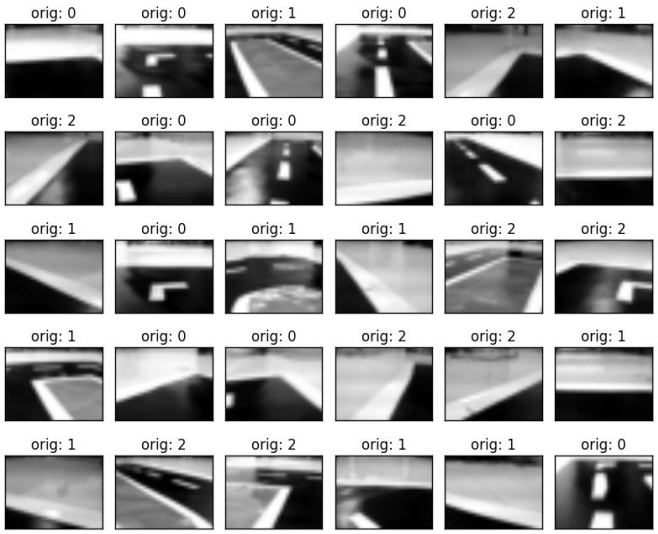


Fig. 1: Random images from dataset

Fig. 1 shows some example of images from our dataset. Actual actions are placed on top of images, 0 is for forward, 1 for left and 2 for right action.

Table I show the structure of dataset.csv with actions against the images of center camera. The actions, 0 for forward, 1 for left and 2 for right, are stored against images of three cameras

(left, right and center) of each state at certain time along with the images of next state.

TABLE I: Dataset File

state0_camera_c	action	state1_camera_c	time
images/img11.jpg	0	images/img10.jpg	53:43.9
images/img13.jpg	0	images/img12.jpg	53:48.5
images/img17.jpg	2	images/img16.jpg	53:49.6
images/img23.jpg	1	images/img22.jpg	53:51.8
images/img29.jpg	1	images/img28.jpg	53:52.7

VI. Methodology

To make the car autonomous, Image Processing, Artificial Intelligence and Machine learning techniques are used. The two approaches, supervised and reinforcement learning are used for the learning of car to make it autonomous which are discussed in detail in coming section.

A. Supervised learning

In supervised learning, the external feedbacks are used by learning functions to map input to output. This external feedback acts as a "teacher" for the learning function. Convolutional Neural Network (CNN); one of the deep learning architecture [11], is used for the supervised learning. The greatest thing of any deep learning model is extracting feature representations through back-propagation. Unlike other classifiers which need hand-engineered features, CNN knows which information like color or shape is important to do the task. The motivation of using the CNN over other neural network will be discussed later.

1) Image Preprocessing: Dataset is consist of total 3025 samples from which training data has shape (2420, 24, 32) because it has 2420 training sample images and test data has shape (605, 24, 32) because it has testing 605 sample images. The length of output class in 3 which are the actions the car can take that includes forward, left and right. The original images gathered are in RGB form containing three layer and the size of the images are 320x240.

a) Resize: Each 320x240 image of train and test sets is converted into matrix of size 1x24x32 which is fed to neural network. This resize is done to minimize the learning time of hyper parameters.

b) Rescaling: Data is converted to type float32 before feeding to the network. Also, image pixel values are re-scaled to 0-1 range.

c) Image Equalization: Images are gathered by driving the car in real track which could have low contrast places or different illumination conditions which can affect the results. To normalize under different illumination conditions, equalization of image histogram is used. After that, image is smoothed to reduce noise introduced by histogram equalization.

d) Image labeling: The Fig. 2 shows the image labeling scheme of for each action over three images. When the car takes the forward action, the center image is labeled as forward, left image is labeled as right and right image is labeled as left (Fig. 2a). The described pattern of image labeling is followed because prediction is made on the basis of center camera image, so when a car reaches at a state, where the center camera image is like left camera image in Fig. 2, the appropriate action at that stage would be left.

e) Class labeling: Since data is categorical, which needs to be converted it to a vector of numbers. Using One hot encoding, a boolean row vector of 1x3 for each image which consists of all zero values expect for the class it represents. For example, for forward class, it will be [1 0 0] and for left it will be [0 1 0].

f) Class weights: For each class, there are different number of samples in dataset e.g. forward class could have 500 while right and left have 400 and 300 etc. In such scenario, class_weights are assigned to each class and data is standardize so that each class has equal distribution of samples while training.

2) Deep Learning for Autonomous Driving: As the goal at this point is to predict the best actions, on a given state of the car in environment, so to classify the state information i.e. camera's images with respect to actions. There are different classification algorithms but Convolutional Neural Networks (CNN) model is used for the described problem.

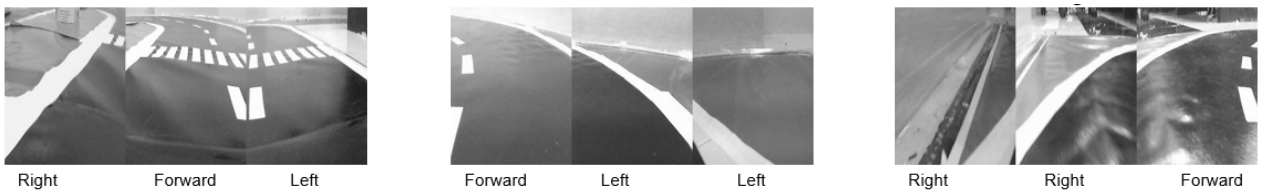
Why CNN?

The motivation of using the CNN over other classification techniques or neural networks is that CNN make efficient use of patterns and structural information in an image. As for instance, in RNN, output dependency on all previous values results very bad for images. Furthermore, in recurrent models such as in LSTMs, the output becomes more biased towards recent historical states as the sequence becomes too long [10]. As the model is targeted the the Raspberry pi based car with low computing power, CNN is the best bet that can be made. So, in result, the model have only 20,000 parameters to fit.

a) CNN Architecture: The architecture proposed has 15 layers, of which 4 are CNN layers. It has around 20,000 parameters and resulted with 73% test accuracy and 89% train accuracy. The accuracy graph is shown in figure 4. At each training stage, some neurons are ignored during a particular forward or backward pass using Dropout of 0.1 to prevent over-fitting. As input layer is benefited from normalization, same is done for hidden layers by using Batch Normalization along with Dropout, which provides regularization of hidden layers. (Figure 3)

B. Deep Reinforcement learning

Deep reinforcement learning is a mixture of deep learning and reinforcement learning (RL). In RL, there is no external feedback or answer key to our problem but agent (the car in our case) still has to decide how to act in a certain situation or task. So, to perform it's task, agent learns from its experience. Still, some knowledge is provided about the goodness or badness of an action on a state, known as reward (or punishment in case of negative reward). So unlike supervised learning, reinforcement models employs different dynamics



(a) Forward (b) Left (c) Right

Fig. 2: Image Labeling

i.e. rewards or punishment to "reinforce" unprecedented types of knowledge.

So the car has a current state s of the environment E, the car can perform an action a which will transition it to next state s'. Each action is evaluated by a reward r. The set of action car takes, defines the policy Π and reward in returns, defines the value v.

The goal is to select the correct policy Π to maximize the rewards. This just defines our solution in reinforcement learning scenario in Markov Decision Process framework [12], So we have to maximize

$$E(r_t|\Pi_t, s_t) \tag{1}$$

for all possible states s at time t.

The rationale for introducing the deep reinforcement learning architecture in the autonomous car is that apart from using the knowledge of dataset samples from training, the car should also be able to continuously learn and leverage from every experience by getting the feedback on each decision it make which will enable it to take the befitting actions in future.

1) Deep Q-learning: In Q-Learning Algorithm, a Q-Function $Q(s, a)$ is used to approximate the reward on a state, where Q is a function which calculates the expected future value from state s' and action a. The Q-Function can be described by Bellman Equation [13] as follow:

$$Q(s_t, A) = \alpha[r_{t+1} + \gamma maxQ(s_{t+1}, A)] \tag{2}$$

We are using Q-learning, the policy-based model-free reinforcement learning with a neural network to approximate the reward based on the state, which is also known as Deep Q-Network (DQN) [14].

a) Neural Network to Predict Rewards: Same CNN model as Fig. 3 is used for the simplicity. The training process of model makes the neural net to predict the reward value of each action from a certain state. The action with maximum reward is performed. We are setting rewards based on environment information such as specific reward if the car takes appropriate actions on a specific traffic signs, negative reward if car remain stop when there's no stop sign or hurdle, a huge positive award in case of goal state and small positive reward when on each correct step car take. These rewards values will be soon explained in detail.

b) Exploration - Exploitation: The car might find; let's say, a stop sign and continue to take stop action and spend all the time exploiting that discovery by racking up small reward it's getting and never go further to find a larger reward of goal state. To overcome this middle exploration-exploitation trade-off is done. This is where epsilon comes in; epsilon is the percent of the time the car will takes a random action rather than the action that is most likely to maximize reward. In the beginning, car takes many random because of higher value of epsilon (0.8) and explores the track. At each step, a decay of 0.99 take place in epsilon which decreases the exploration rate over time and start to exploit the reward information got during exploration.

c) Discounted Reward: To keep the future reward in mind, such that the end result from prediction model should specify that taking a stop action has the best reward in current state especially when it can obtain additional reward, if there is a goal state next to stop if car keep moving forward. A loss function shows the difference between target and prediction, it is defined as following:

$$loss = (r + \gamma maxQ'(s', a) - Q(s, a))^2 \tag{3}$$

where the term $r+\gamma maxQ'(s, a')$ represents the target value and $Q(s, a)$ is predicted value.

γ is gamma the defines discount rate (0.9) which help to converge discounted future to the best reward for the given state.

At first, car performs action a and get the reward r and resulting new state s'. Using the new state we compute its maximum target Q with a discount. This introduce the little variation in the reward for better exploration. Current reward is then added to the resulting discounted future to achieve the target value. Then the loss is computed by taking difference of current prediction from the target.

d) Memory: The memory of the experiences is kept and at each state, model is trained on these previous experiences in memory. These experience contain information of state, action taken, reward and next state.

e) Traffic Sign Recognition for rewards: A self-driving car should be able to obey the traffic signs so the car needs to recognize the signs. Each traffic sign proposes a different action to perform with different reward. For example, on a custom stop sign, there should be a positive reward if car takes a stop action and a negative reward if car keeps driving. Traffic signs show a wide range of variations between classes in terms of color, shape, and the presence of pictogram or text. There are thousands types of traffic sign present

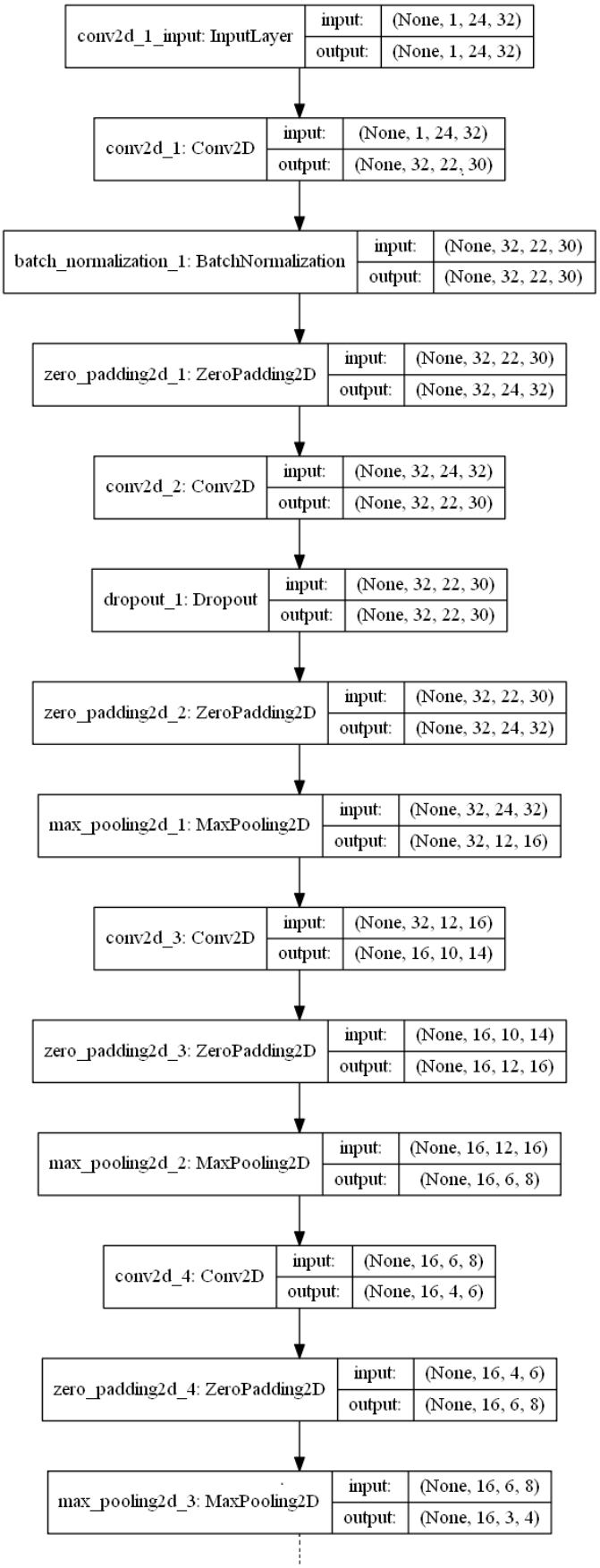


Fig. 3: CNN model architecture

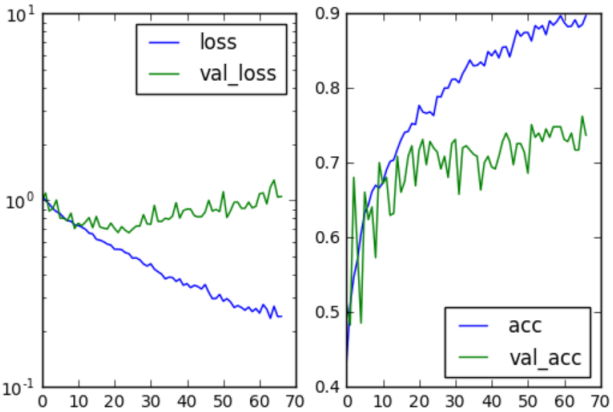


Fig. 4: Accuracy and Loss on Training and Testing Data Points

today, for example even the German Traffic Sign Detection Benchmark (GTSDB) comprises of dataset of over 1200 types of traffic signs [15]. The Fig. 5 shows four main categories in GTSDB based on the shape of the sign. Sign detection and recognition of each and every sign is a very long task, we are considering three main sign which are stop, no-left turn and traffic light. For detection of these signs. OpenCV provides many efficient and effective methods of detection such as cascade classifiers which we have used. These classifiers are based on voila-jones cascade classifiers based on HAAR features [16]. The main advantage of using cascade classifiers over other OpenCV detection algorithm is it works efficiently in embedded systems with low resources which was very beneficial in our case. Moreover, HAAR feature considers adjacent rectangular features in image instead of computationally expensive image intensities (RGB pixel value at every pixel of image) like in LBP. Since, cascade classifiers are binary classifiers, three classifier for each classes (stop, no-left, traffic-light) are trained. For each classifier training, 1,800 of positive images (contain the signs in image) are

gathered for three classes for positive dataset and a vector file is made from these which contains the path to the image and the coordinates of sign in image, 900 non-sign images are used as negative dataset and *.dat* file is made from it which contains the negative images path. Finally, using information from *.vec* and *.dat* files three cascade classifiers are trained with `opencv_traincascade` function consist of 10 stages with $minHitRatio$ = 0.998, $maxFalseAlarm$ = 0.3 and sampleheight and samplewidth $(-w, -h)$ of 20. On unseen dataset, `stop_classifier`, `noleft_classsifier` and `trafficlight_classifier` gives 98% , 96% and 91% accuracy respectively.

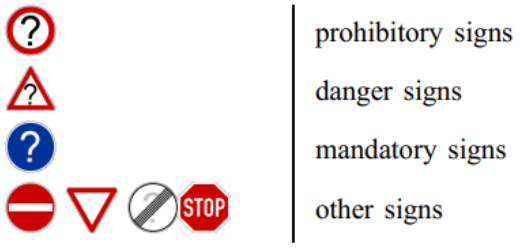


Fig. 5: GTSDB: Three categories are assigned to the sign on the basis of sign pattern and a fourth category to relevant signs

(a) no-left turn (b) stop sign (c) traffic light

Fig. 6: Sign detection with OpenCV cascade classifiers

For the recognition of which traffic light in blinking, OpenCV color detection technique is used. Color boundaries for red, green and yellow in BGR are set, the color in boundary is found in image and the light is recognized by the location of the color found. Fig. 6 shows a demo of sign detection. The car can detect the sign from a long distance but it must take decision according to sign from a specific distance say 30cm away from sign in our case. So the distance from camera to sign is calculated using Triangular similarity $F = P * D/W$ and $D' = F * W/P'$ where P in sign size in pixel, W is actual width of sign at distance D, F is focal length, P' perceived width and D' is calculated distance from camera to sign.

f) Reward values: Reward for each state is set according to the action taken in presence or absence of different environment factors. For example, Tables II and III show the reward values for action forward and stop in presence or absence of a stop and traffic light sign. If there is a stop sign detected within a specified distance, and car takes a forward action, it should be punished by negative reward say -0.5 while in case if car stops, it did a good job and should be rewarded with +2. In a scenario, where there is no stop sign, car should keep moving forward. A similar but a bit complex scheme is used for setting the rewards for the traffic-light sign. If there a traffic sign detected within defined distance, there can be different action on different possible states of the traffic light i.e forward action for green and yellow light, stop for red and keeping the previous action in case no light is on and reward value is set for each possibility is set accordingly.

TABLE II: Reward Table for Stop Sign

Stop Sign Detected	Action Taken	Reward Value
True	forward	-0.5
	stop	+2
False	forward	+0.05
	stop	-0.5

TABLE III: Reward Table for Traffic Light

Traffic light State	Action Taken	Reward Value
Red Light	forward	-1
	stop	+0.01
Green/Yellow Light	forward	+0.01
	stop	-0.1
No Light	forward	+0.01
	stop	-0.1

VII. Conclusion

There are many projects related to self-driving car, in our case we have created a customized car and environment from scratch which helped in understanding all the back-fall one can face, such as car on map with single light perform different while when four light above it i.e. a brighter environment. We also found out that there can not be a single technique to solve the whole problem, its the combination of multiple algorithms together which sums up to give good results.

For our case Supervised learning was done using Convolutional Neural Network and we achieved 73% test and 89% train accuracy.

In addition to this we trained car with reinforcement learning for three different sign boards; Stop, No left and Traffic light using deep Q-learning technique with CNN model which we used earlier.

VIII. Future Work

Currently, the training is done in a constrained environment, many factors of real environment can sway the prediction of model. Lightening effects and weather conditions influences the images we get from cameras which can have a great impact on model prediction. We look forward for further improvements so that car can have as good results in real environment as we have in our custom environment. For that decision technique could be done using object and line detection. Moreover, introducing SLAM for decision making on roads with fork will have noticeable impact.

REFERENCES

[1] Nov 28, 2017, THE VERGE, Vlad Savov, "Waymo's self-driving cars rack up 4 million miles on public roads," https://www.theverge.com/2017/11/28/16709104/waymo-self-driving-autonomous-cars-public-roads-milestone/.

[2] Sep 20, 2012, ILSTV Auto & Trucking, Trends, "Self-driving cars could account for 75% of all vehicles on the roads by 2040," http://co.water.usgs.gov/trace/pubs/wrir-99-4279/.

[3] Oct 16, 2017, Argo AI, Bryan Salesky, "A Decade after DARPA: Our View on the State of the Art in Self-Driving Cars,"https://medium.com/self-driven/.

[4] 2011-2018, Udacity, Inc, https://www.udacity.com/course/self-driving-car-engineer-nanodegree–nd013/.

[5] J. McCall, M. Trivedi, "Video based lane estimation and tracking for driver assistance: Survey system and evaluation, " in IEEE Transaction on Intelligent Transportation Systems, vol. 7, no. 1, pp. 20-37, 2006.

[6] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Proc. 25th Int. Conf. Neural Inf. Process. Syst., 2012.

[7] L. Ran and Y. Zhang, "Convolutional Neural Network-Based Robot Navigation Using Uncalibrated Spherical Images," .

[8] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates and A. Y. Ng, "An Empirical Evaluation of Deep Learning on Highway Driving, " in Computer Vision and Pattern Recognition, 2012.

[9] R. Hadsell, P. Sermanet,J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller and Y. LeCun, "Learning long-range vision for autonomous off-road driving.," in J. Field Robot. 2009.

[10] C. Chen, A. Seff, A. Kornhauser, J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," Proc. IEEE Int. Conf. Comput. Vis., pp. 2722-2730, 2015.

[11] Y. LeCun, Y. Bengio & G. Hinton, "Deep learning" in Nature, 2015.

[12] Dec 19, 2015, Tambet Matiisen, "DEMYSTIFYING DEEP REINFORCEMENT LEARNING," http://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/.

[13] Francisco S. Melo, Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, PORTUGAL, "Convergence of Q-learning: a simple proof".

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis," Human-level control through deep reinforcement learning. "Nature, vol. 518, 2015.

[15] J. Stallkamp, M. Schlipsing and J. Salmen, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in The 2011 International Joint Conference on Neural Networks, 2011.

[16] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1. IEEE, 2001.