RESEARCHERID
THOMSON REUTERS
[Nagda, 5(10): October 2018] ISSN 2348 – 8034
DOI: 10.5281/zenodo.1476487 Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

AUTONOMOUS RC CAR USING OPTIMIZED NEURAL NETWORKS

Malay Nagda[*1], SiddheshKaranjkar[2] & Sudhanva Vaidya[3]
[*1,2&3]K.J Somaiya College of Engineering, Department of Electronics Engineering, Mumbai, India

Abstract

Due to rapid developments in the field of autonomous driving the roads are about to get a lot safer and a lot of our traffic woes will be solved in the near future. The key contributions of this paper include improving the quality of collected data and optimization of neural network parameters for achieving high accuracy in predicting the directions for the RC car to drive autonomously. To ensure this, we made use of methods like Isolation forests for removing erroneous training data, carried out regularization using dropout to avoid overfitting on certain specific training dataand varied parameters like momentum and decay value. Also, front collision avoidance was implemented using ultrasonic sensor. Another task that we realised was traffic sign identification using Haar Cascade Classifiers. Monocular distance measurement method was used to calculate the distance to the traffic signs from the Pi Camera attached to Raspberry Pi on the RC car after calibrating the Pi camera to calculate the intrinsic parameters in order to eliminate the distortion caused by camera lens which might reduce the accuracy of the distance measured.

Keywords- *Autonomous Vehicle, Neural Network, Haar Cascade, OpenCV, Dropout, Isolation Forest, Gradient Descent, Momentum.*

I. INTRODUCTION

An autonomous car is a vehicle that is capable of sensing its environment and navigating through it without human input. From all the recorded traffic accidents, 90% of accidents are caused by human error. Autonomous car promises to tackle this problem of vehicular accidents. The fundamental technology used by industry leaders is Computer Vision. In conjunction with Computer Vision, the advent of machine learning has boosted the development process of autonomous cars.

A major part of autonomous car is measuring the distance of the objects in and around the car. During our research we came across industry grade components and techniques used to achieve this goal. Autonomous cars use LiDAR sensor, LiDAR [1] (Light Detection and Ranging) is active remote sensing. This means the LiDAR system sends a pulse of light and it waits for the pulse to return. We did not use it due to its cost and bulk and instead used a method involving Pi Camera.

Chu, Ji, Guo, Li and Wang (2004) [2] have devised a monocular vision system whose primary area of interest is found by the lane borderline and a likelihood target vehicle is searched by the grey difference between the target vehicle and the background; second, a target vehicle is affirmed by a symmetry character and a position of the vehicle symmetrical axis is ascertained; third, the object vehicle is tracked by Kalman forecast principle; fourth, a method of detecting distance in a frame of image is introduced.

The implementation of Haar Cascade Classifiers in the OpenCV library utilizes a technique described in a paper by Viola and Michael Jones [3]. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. We used this technique for identifying different traffic signs.

[Nagda, 5(10): October 2018] ISSN 2348 – 8034
DOI: 10.5281/zenodo.1476487 Impact Factor- 5.070

We also utilize the paper by Liu, Ting and Zhou [4] which describes the isolation forest and how they can be used for differentiating anomaly from normal instances by random partitioning of data points after a selecting a feature arbitrarily. The anomalies require less number of splittings to be uniquely identified than normal instances.

The next section describes the hardware components, design of the RC Car system and working of the RC car model. Section 3 describes configuration and process for achieving sign detection. Section 4 illustrates data collection, training and optimization of neural network for autonomous driving with optimal accuracy and the section after this shows the effects of different parameters on the accuracy and training time of the neural network. The entire paper is concluded in Section 6 along with how certain aspects of the proposed RC car model can be improved and new ones added as future scope.

II. PROPOSED METHODOLOGY

In this paper we have used the results from few of the papers mentioned in the previous section for developing an autonomous car, which fulfils the objectives mentioned in Fig 1. This section defines the configuration of the system, the setup required and the training process carried out in order to achieve the three objectives of the system.

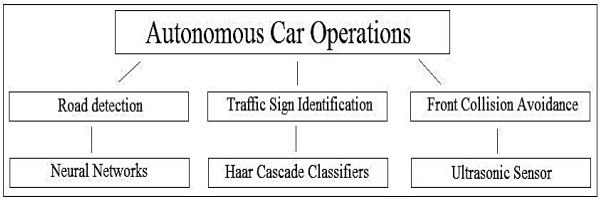
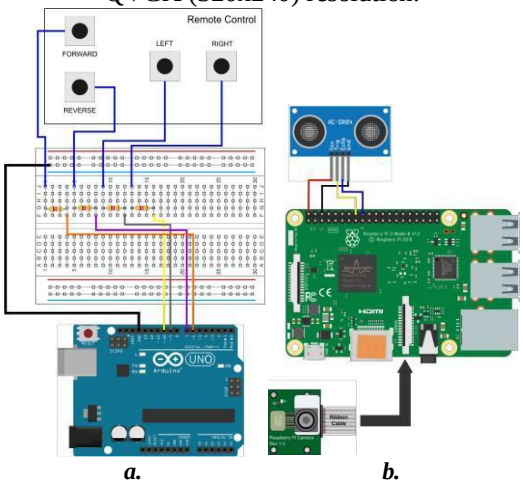


Fig 1. SystemObjectives

2.1. System Design-
The system is subdivided into three parts: input unit, consisting of camera and ultrasonic sensor, processing unit i.e. computer and RC car forming the output unit.

2.1.1) Input Unit
A Raspberry Pi board (model B+), attached with a Pi Camera module and an HC-SR04 ultrasonic sensor is used to collect input data. Two client programs run on Raspberry Pi for streaming colour video and ultrasonic sensor data to the computer via local Wi-Fi connection. In order to achieve low latency video streaming, video is scaled down to QVGA (320x240) resolution.



a. ***b.***

Fig 2. a. Hardware connections to RC Car b.Hardware connections to server (laptop)

[Nagda, 5(10): October 2018] ISSN 2348 – 8034
DOI: 10.5281/zenodo.1476487 Impact Factor- 5.070

2.1.2) Processing Unit

The processing unit (computer) handles multiple tasks: receiving data from Raspberry Pi, neural network training and prediction(steering), object detection(stop sign and traffic light), distance measurement(monocular vision), and sending instructions to Arduino through USB connection.

2.1.3) Output Unit

The RC car which is the output unit is controlled by the processing unit via Arduino which has RC car remote control attached to it. Whenever a low signal is givenby Arduino to the RC remote, a button press action is simulated as the resistance between ground and the pin on the controller goes to zero and the car moves in the direction in accordance with controller pin manipulated. The resistance remains as it is between the controller pin and ground.

2.2. System Configuration

The system consists of two TCP servers, one for transmitting the images from the Pi Camera to the computer and the other for transmitting the distances calculated with the help of ultrasonic sensor. Both the servers transmit information simultaneously and in real time.

Images are transmitted from Pi Camera via raspberry pi through a TCP connection to the laptop and are converted to grayscale to reduce computation power. The lower-half of the grayscale image is used for autonomous driving. The second TCP connection continuously sends the distance of the farthest obstacle that can be detected.

The system consists of two TCP servers, one for transmitting the images from the Pi Camera to the computer and the other for transmitting the distances calculated with the help of ultrasonic sensor. Both the servers transmit information simultaneously and in real time.

[Nagda, 5(10): October 2018]
DOI: 10.5281/zenodo.1476487
ISSN 2348 – 8034
Impact Factor- 5.070

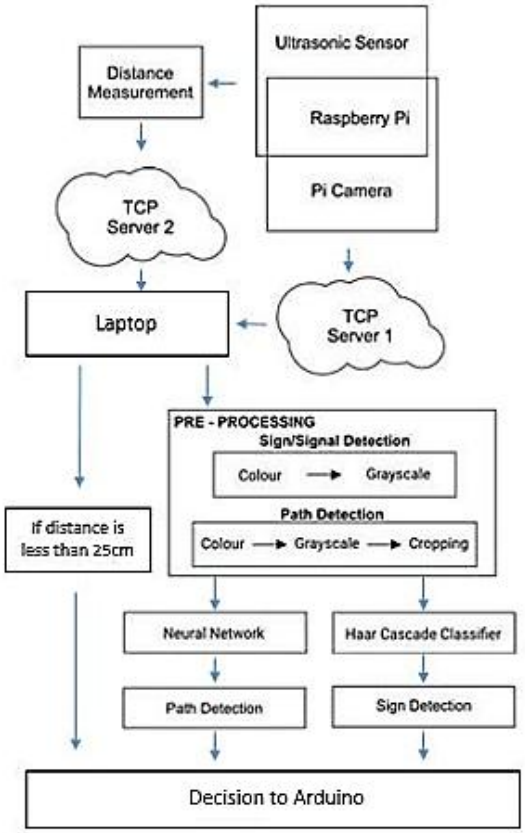


Fig 3. Proposed working of RC Car system

The system consists of two TCP servers, one for transmitting the images from the Pi Camera to the computer and the other for transmitting the distances calculated with the help of ultrasonic sensor. Both the servers transmit information simultaneously and in real time.

Images are transmitted from Pi Camera via raspberry pi through a TCP connection to the laptop and are converted to grayscale to reduce computation power. The lower-half of the grayscale image is used for autonomous driving. The second TCP connection continuously sends the distance of the farthest obstacle that can be detected.

The laptop shows the real-time video feed from the Pi Camera. The server program utilizes the weights in the xml file which represent the trained model to predict the direction in which the RC car is supposed to move. The predicted direction is printed on the screen and the car moves in that direction using the serial connection to Arduino as was described in the data collection process. Xml files associated with the each traffic sign to be detected are also loaded in the same program. When a traffic sign is detected it is marked in the real time in the video stream with a rectangle, the name of the traffic sign along with the distance to the traffic sign from the instantaneous position of the RC car and corresponding action is taken by the program. Let's say, a stop sign has been detected; the motion of the RC car will stop for 5 seconds and then continue moving in the predicted direction. In case of traffic light detection, the colour of the traffic light also has to be determined. For this, a Gaussian filter is applied on the area where the traffic light is detected. The brightest spot is estimated based on the Gaussian blur and based on the position of the brightest spot in the area where traffic light is present, we determine the colour of the traffic light and appropriate action is taken.

[Nagda, 5(10): October 2018] ISSN 2348 – 8034
DOI: 10.5281/zenodo.1476487 Impact Factor- 5.070

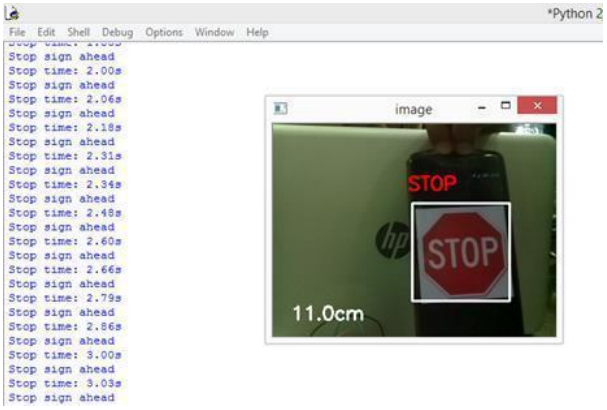


Fig 4. Stop sign detection with distance from thePi Camera.

Whenever the distance of the obstacle as calculated by ultrasonic sensor reduces below a threshold, the RC car immediately stops by sending a logic high to all the pins on the RC remote, irrespective of the traffic signs detected or the directions predicted. The RC car begins to drive autonomously as the program starts predicting the directions once the obstacle distance is above the threshold.

In the next section we specify how the objectives of sign detection and autonomous driving were achieved using Haar Cascade Classifiers and Neural Networks.

III. HAAR CASCADE CLASSIFIERS

Here, the initial configuration of Pi Camera to accurately calculate the distance to the traffic signs and the training process of Haar Cascade Classifier to detect those signs is described.

3.1) Camera calibration- For calculating thedistances we require the intrinsic parameters [8] of the Pi Camera which are focal length and optical centre of the Pi Camera and are found using camera calibration. These intrinsic parameters have minor variations between every other Pi Camera due to slight difference in conditions during manufacturing of each Pi Camera.

OpenCV has APIs for calculating the intrinsic parameters of the Pi Camera. We used a modified chessboard to do so. Pi Camera calibration requires 3D physical world coordinates called as object points (defined in the images taken from the Pi Camera at different positions and orientations) and 2D image points (defined where two black squares touch other) as input data. Each object point will have coordinates as (x,y,and z). It is assumed that z=0 for simplicity and the Pi Camera was moved accordingly. These points were taken as (0,0),(1,0),....,(8,5) as we intended to find 9x6 grid pattern using the function cv2.findChessboardCorners().

Fig 5. a. Image of chess grid taken with raspberry pi. b.9x6 grid detected on the same image.

It returns the position of corner points i.e. the image points. These image points and object points are then to be used in the function cv2.calibrateCamera to find the intrinsic parameters.

3.2) Distance Measurement-Since Raspberry Pisupports only one Pi Camera, we utilised the distance measurement by monocular vision method given by Ji,Chu, Li, Guo and Wang (2004).

P is the point to which distance is to be measured i.e. d units. h is the height at which the Pi Camera is attached from the surface level and α is the angle from point P to the optical centre. In our case, it is the angle at which the traffic signs are to be detected.

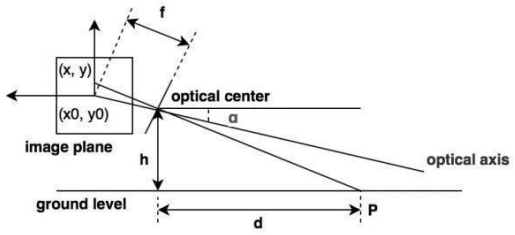


Fig 6. Distance measurement diagram [5]

We took the value of α at 8° as the traffic signs will be slightly below the image plane.
The formula for calculating the distance is

$$d = \frac{h}{\tan\left(\alpha + \arctan\left(\frac{v - v_0}{a_y}\right)\right)}, \left(a_y = \frac{f}{dy}\right) (1)$$

Here, v_0 and a_y are values we get from the camera calibration process.

3.3) Haar Cascade Classifier training process
We used Haar classifiers as it has higher success with precision and recall than LBP and CNN [6], are comparatively easier to train and easily manage scaling objects due to strong invariance. One drawback of this technique being higher time for processing. We used them to detect stop, school ahead and traffic signal signs.

The steps involved in the training of a Haar Cascade Classifier [7] are as shown in Fig 7.

ISSN 2348 – 8034
Impact Factor- 5.070

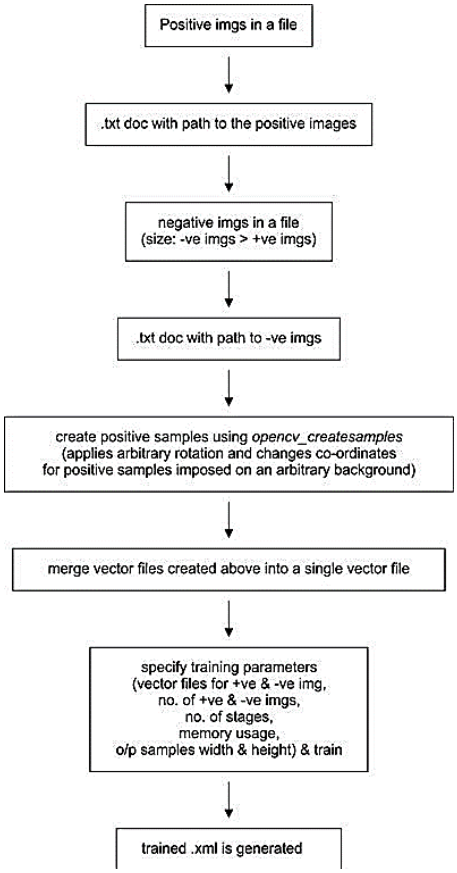


Fig 7. Flowchart of Haar Cascade training process

IV. NEURAL NETWORKS

Here, we describe the process of collecting data and the training of a neural network using that data using which the car will be able to drive autonomously.

4.1) Data collection -Training data is the collection of image frames of the road on which the RC car is to drive autonomously. These frames were tagged with direction data using manual input from the keyboard (arrow keys) for supervised learning of the neural network.
Two main libraries involvedin this process were OpenCV and Pygame and the process for training data collection was as follows-

1. A TCP connection was formed using Wi-Fi connection between the Raspberry Pi and server (laptop).

2. Video feed captured by the Pi Camera was sent to the laptop using client-server model with Raspberry Pi as client and laptop as the server. The resolution of the video feed was reduced to QVGA (320x240) for low-latency video streaming.

3. Pygame is used to register an arrow key press and move RC car on the defined road in the corresponding direction. With each key press a unique number is sent via serial port to Arduino.

4. Arduino sends a low signal to the pins (connected to RC Remote) associated with the number received and the RC car moves in that direction.

ISSN 2348 – 8034
Impact Factor- 5.070

5. Image frame in the form of a numpy array is paired with the arrow-key press data is saved in npz format.

6. We collected 11000 frames to train the neural network.

4.2) Cleaning of data- While collecting data for the training of neural network, at certain times the wrong arrow key might have been pressed; this incorrect key press will correspond to the image frame that does not represent the expected direction the car is supposed to move. As it is not feasible to look at all the saved npz files for detecting the anomaly, we used the anomaly detection technique which incorporates the use of an isolation forest. We assume that the maximum image frames are correctly tagged with the key-press data. Using this method, the algorithm isolates the image frame with anomaly as this image frame will have a numpy array that is not similar to all the numpy arrays of the image frames that are tagged correctly with the key-press.

4.3) Neural Network Training process- Only the lower-half of the image frame is used for training the neural network as the path that the RC car is supposed to traverse will not appear in the upper half of the image, the height of the image frame used intraining process will reduce by half to 120 pixels from 240 pixels and the number of pixels along the width ofthe image frame remain unchanged to 320 pixels. Hence, total number of pixels of the image frame which correspond to the total number of nodes in the input layer of the neural network will be 320x120= 38400. The number of nodes in the hidden layer was chosen to be 32. It can be taken as any arbitrary value. The output layer of the neural network indicates the four direction in which the RC car can traverse: forward,reverse,left and right.

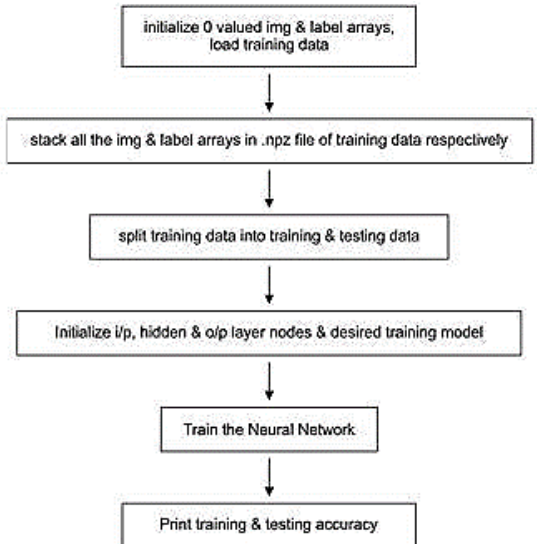


Fig 8. Flowchart of Neural Network training process.

The algorithm used for training of neural network was backpropagation algorithm which uses the concept of gradient descent for optimizing the weights towards a minimum value of error function.

4.3.1) Training Optimization- The process of neural network training was optimized by varying
1. Learning rate
2. Decay value
3. Momentum value
4. Number of hidden layers

[Nagda, 5(10): October 2018] ISSN 2348 – 8034
DOI: 10.5281/zenodo.1476487 Impact Factor- 5.070

The results of these variations are as shown in table.

To avoid getting stuck in one of the multiple local minim as due to a complex error function, we used the concept of momentum in the objective error function .Momentum has a value between 0 and 1. It reduces the time required for training the neural network as it increases the size of the steps taken to reach the global minima. If the value of momentum is too big, it is possible to miss the global minima andif it is too small the training process will be very slow. The effects of momentum on the training performance is shown in section

4.3.2) Avoiding overfitting using dropout- In Dropouttechnique random neurons are deactivated so that the neurons which are connected to higher weight values do not dominate the network and thus the network avoids overfitting. Dropout can be used on the input layer as well as the hidden layers but is avoided on the output layer.

The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data. This can be seen from the Fig 11.

V. RESULTS AND ANALYSIS

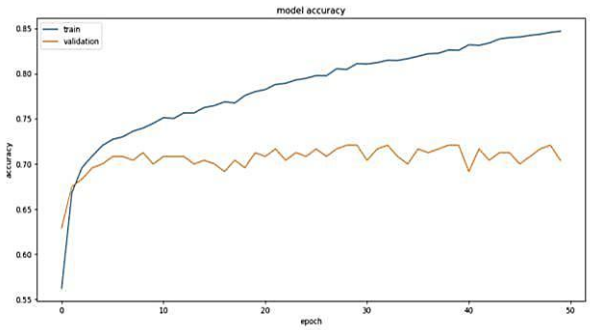


Fig 9. Accuracy plot without dropout

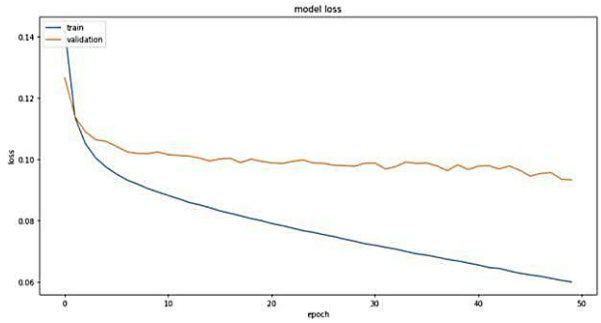


Fig 10. Model losses plot without dropout

For this model, the accuracy on training data was observed to be 84.3% from Fig 9. and the accuracy on validation data was seen to be around 70.65% in Fig. 9 The difference in accuracy of the training and validation data is very high so we can conclude that he model is overfitting on the data if we do not include a Dropout layer.

ISSN 2348 – 8034
Impact Factor- 5.070

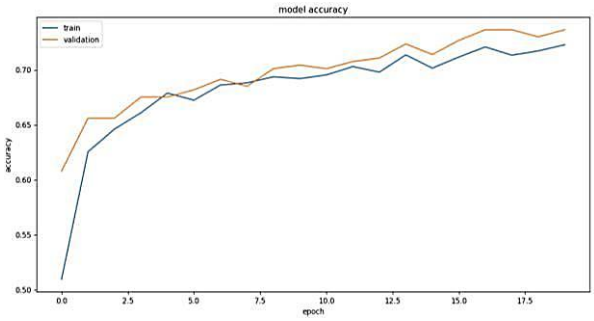


Fig 11. Accuracy plot with dropout

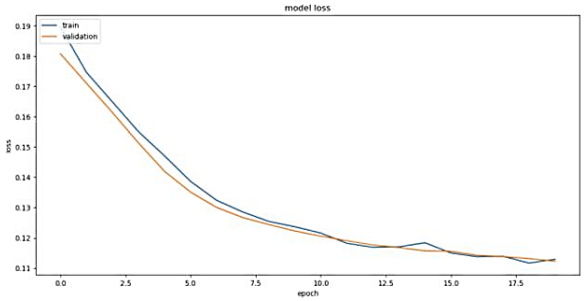


Fig 12. Model losses plot with dropout

The model accuracy as observed from the above graph is 72.56% on the training set and 73.67% on the validation set. The loss is for training set is 11.47% and for validation set is 12.83%. For this model, the learning rate was set at 0.001, momentum at 0.9 and a single hidden layer. We can observe that after 50 epochs the model accuracy and losses are better as compared to the other models. As momentum variable is added in the model the model converges faster.

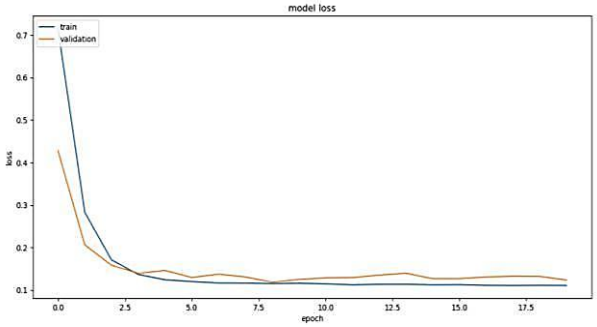


FIg 13. Model losses plot with high momentum

For the above implemented model the learning rate was set at 0.001, decay as 10e-6, momentum at 0.9 and the training model involved two hidden layers. The minimum loss value is achieved in the initial epochs itself. There is no significant change in the loss value over the entire training process.

ISSN 2348 – 8034
Impact Factor- 5.070

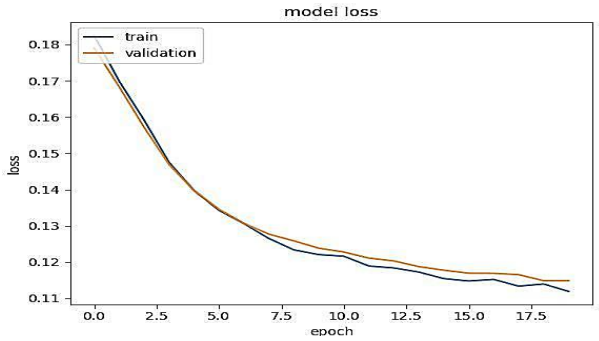


Fig 14. Model losses plot with low momentum

For this model the learning rate was set at 0.001, momentum at 0.5 and the model was trained with twohidden layers. It is seen that after reducing the momentum from 0.9 (used in the previous training model) to 0.5, the model takes time to converge after the same number of epochs. Hence a higher value of momentum can be used for faster convergence of the model.

Table 1. Performance of the neural network with differentoptimization parameters.

Learning Rate	Decay	Mome-ntum	Number f hidden ayers	Accuracy (training et)	Accuracy validati on set)
.001	0e-6	.9	(with ropout)	8.75%	7.64%
.001	0e-6	.5	(with ropout)	6.76%	7.38%
.01			(with ropout)	5.45%	9.78%
.001	0e-6	.9	1(without ropout)	4.3%	0.65%
.001	0e-6	.9	(with ropout)	2.56%	3.67%

VI. CONCLUSION AND FUTURE SCOPE

We successfully achieved the three main objectives- autonomous driving, traffic sign detection and front collision avoidance with reasonable accuracy. This was possible due to removal of improper data and optimizing various parameters during the training of neural network training resulting in direction prediction with high accuracy.

Calibration of Pi camera gave more accurate distances to the traffic signs as the error due to minor defects in the lens was accounted for. The distances were calculated using a single rather than two lens successfully.

In order reduce the complexity of the system, we can eliminate certain components like the radio controller by interfacing the motors of the RC car with raspberry pi. The control of these motors can take place with a third TCP connection. This will also result in elimination of Arduino.

While implementing the system we realised that, the car remains in motion after detecting a traffic sign or an obstacle in front even if the motors have turnedoff. This happens due to the inertia of the car. This can be solved by introducing a braking system.

The motors used in the car are DC motors. Hence, implementing a speed control mechanism is not practical. With stepper or servo motor, speed control can be implemented which will enable incorporation of even more traffic signs. Speed control can be utilised for taking left-right turns.

Lane changing can also be incorporated, where the car detects different lanes on a particular road and changes the lane to overtake another vehicle.

A major future scope is to integrate this system in multiple cars and develop communication between those cars. This will make a whole network of autonomous cars which will communicate with each to navigate through the city.

VII. ACKNOWLEDGEMENTS

The support and guidance of Dr. Sudha Gupta, Associate Professor and Dean of Student Affairs at K.J Somaiya College of Engineering has been immensely valuable at each and every step right from the implementation of the objectives to the writing of this paper.

REFERENCES

1. *GISGeography. (2016). A Complete Guide to LiDAR: Light Detection and Ranging. [online] Available: https://gisgeography.com/LiDAR-light-detection-and-ranging/*
2. *C Jiangwei, J Lisheng, G Lie, Libibing, W Rongben, "Study on method of detecting preceding vehicle based on monocular camera," for IEEE Intelligent Vehicles Symposium, 2004*
3. *P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*
4. *F Tony Liu, K Ming Ting, Z Zhou, "Isolation Forest," for 2008 Eighth IEEE International Conference on Data Mining, 2008*
5. *Orr, G. (2013). Momentum and Learning Rate Adaptation. [online] Willamette.edu. Available: https://www.willamette.edu/~gorr/classes/cs449/momrate.html*
6. *Ivan O. (2017). Convolutional Neural Networks vs. Cascade Classifiers for Object Detection. [online]. Available:https://dzone.com/articles/cnn-vs-cascade-classifiers-for-object-detection*
7. *Wang, Z. (2015). Self Driving RC Car. [Blog] Zheng Wang. Available: https://zhengludwig.wordpress.com/projects/self-driving-rc-car/*
8. *Camera Calibration. [online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html*