Autonomous vehicle using Artificial Intelligence

Tejoram.V, Jenisha Priscilla.J, Swetha, B.Bhuvaneshwari and Dhanalakshmi.S
Coimbatore Institute of Technology, Coimbatore, India.

ABSTRACT

Millions of accidents take place every year due to drunk and distracted driving. The cause of these accidents is human negligence. To avoid this and to make roads safer there is a need to make vehicles autonomous. This will also aid the physically impaired and the elderly. These intelligent vehicles should be able to take decisions and navigate on road precisely without the aid of human assistance. Several multinational companies like Google and Tesla have been investing efforts into developing such autonomous vehicle systems with robust algorithms. In this project, a prototype of an intelligent self-driving vehicle was developed with a variety of machine learning algorithms. This prototype was developed on a Raspberry Pi and has been able to predict the direction and control the vehicle based on these decisions. The prototype has also been trained to detect and obey traffic signs and navigate by avoiding obstacles. To achieve this, the images of a track collected from a Pi camera were used to train different models of neural networks and the performance of each model was tested. An optimal model of neural network was then used to process images collected by the Pi to control the wheels of a motor car. Ultrasonic sensor module aids the vehicle to detect and halt on the occurrence of any obstacle. Haar cascade classifier based stop sign detection signals the vehicle to stop. The results from this project will provide an insight on the various learning algorithms best suited for this application which can also be extended to other applications in the areas of industrial and home automation.

INTRODUCTION

Ever since the development of cars, autonomous driving has been of great interest and an even greater challenge to achieve in the field of Intelligent Transportation Systems [8]. With the evolution of technology, driverless vehicles are no longer a distant reality. Companies like Ford, Tesla and Mercedes [26] are already testing their prototypes on road. A major portion of their success is attributed to the boom in the field of machine learning and artificial intelligence. The development of autonomous driving dates back to the 1500’s with Leonardo Da Vinci’s Self-propelled cart, a cart which could move on its own through a known path. This was achieved through high tension springs.

Over the years, the level of autonomy in vehicles has increased progressively. The National Highway Traffic Safety Administration (NHTSA) of the United States has come up with six levels of autonomy, starting from fully manual to fully autonomous [17]. The zeroth level is the no autonomy level wherein the driver performs all the manoeuvring tasks. The first level is the Driver Assistance where few features are provided to assist the driver. The second level is the Partial Automation, where acceleration and steering are partially automated. However, the driver must remain engaged with the steering as well as monitoring the environment. The next stage is Condition Automation. In this level, the driver need not monitor the environment; however with notice or alarms, he must be able to take in-charge of the steering. The fourth level is the High Automation Level in which the driver can choose to manually or autonomously control the vehicle. The fifth and the final level is the fully autonomous level, in which the vehicle can be driven with no assistance in all conditions of road, weather and environment.

The major cause of such an increased interest in autonomy is because of the number of accidents that happen every year due to drunk and distracted driving. Autonomy increases safety and ensures reduced rates of crimes. This also enhances the mobility for children, disabled and the elderly. It reduces the amount of time and energy spent on transportation, thereby increasing the efficiency of work and quality of life in individuals. They could serve as wonderful highway goods carrying trucks where there is less traffic and the routes are known in prior. Mobility as a service could also utilize the benefits of autonomous cars thereby increasing public transportation. This reduces the number of vehicles owned by individuals which in turn produces better flow of traffic, less congestion, lesser need of parking spaces and lesser pollution. In short, the boom of autonomous driving can provide a multitude of benefits to the world.

It is also to be remembered that there are several obstacles in achieving autonomous driving. One of the major obstacles is the potential hacking of the driving systems. Security must thus be given an important priority in designing these vehicles. Another major obstacle is maintenance cost of the sensors and peripherals which could potentially be damaged due to environmental changes. In countries like India, there is a major need to change the infrastructure of the roads for these vehicles to navigate smoothly. However companies like Mahindra are making use of the structured environments in farming to introduce autonomous driving functions. [3]

While the major application of self-driving vehicles is in the field of road transportation, the same machine learning technology can be altered to meet different automation needs in industries. Robots can be made to sense their environment and navigate in factories and can even control the machineries based on the needs. The potentiality of this technology is vast and is expected to bring a huge impact in the future by making a majority of physically and mentally intensive tasks easier. If full autonomy is not desired, intelligent assistive features can be provided.

Fig 1 A Google Waymo car

Companies like Tesla and Google have invested millions into research and development of self-driving cars. In 2009, Google started the development of their self-driving car, ‘Waymo’. By 2013, Waymo had travelled around 2 million miles in the US with only one reported accident. By this time other leading car companies like Nissan, Ford, Benz, BMW had launched themselves into the self-driving technology. The first major setback which speculated the safety and ethicality of the autonomous vehicles was brought by a fatality during the testing of Tesla's Autopilot. However, companies are still confident with the potentiality of these vehicles and claim that such incidents can't be compared to the enormous benefits such vehicles bring. [15] Audi is claimed to be the first company which will provide Level 3 autonomy in its A8 model. Fig 1 shows a Google Waymo car which is currently under testing phase.

LITERATURE SURVEY

Driver assistance systems have been implemented in vehicles to provide safe, easy and improved driving experience. Automated driving systems have emerged as the result of rapid advancements in drive assistance technologies and machine vision algorithms over the past few years [11]. Kichun Jo et al [6] have provided a detailed description of the components and comprehensive instructions for the design and real time implementation of autonomous vehicle. Their work also addresses the several advantages of using a distributed system over the centralized architecture. Their proposed system has achieved significant improvements in system safety, computational load distribution and flexibility of the system against to changes or extensions. However, their work fell short of its capability to work with software platforms other than windows.

The intelligent vehicle takes into consideration of several inputs before making any decision. This methodology explained by Hyunggi Cho et al [2] throws light on multi sensor fusion system which integrates a multitude of sensors like LIDAR, radar, Camera etc. The results of their work shows that the model was able to predict and track better by responding to the threshold value set by several inputs instead of one input. The cameras in the smart vehicles function just like the eyes of a human driver to facilitate the continual monitoring of the environment during navigation. B.S Khan et al [5] has proposed a simple algorithm in which the vehicle navigates by detecting the lane markings. Graph cut segmentation technique along with CLAHE is used to process the lane markings in input images the captured by camera. This technique is robust to camera orientations and illuminations. However, navigation on roads without the road markings is not feasible in this method and also using only the visual information as an input is potentially dangerous as cameras fail to see through adverse environmental conditions like fog, mist, storms. Several advancements have been made to improve the visual capability of the vehicle. In recent times, FIR thermal cameras along with LIDAR sensors have been implemented to reinforce the visual based driving system [16].

A prototype of autonomous car developed by Mohammad Rubayat Tanvir Hossain et al [4], features a robotic car controlled by raspberry pi and Arduino which works in the created environment. Their model is also equipped with Haar feature based cascade classifier to detect traffic lights and obstacle avoidance algorithm using ultrasonic sensor. The car navigated flawlessly on a designed track by using OpenCV based lane detection techniques like Canny edge detection and Hough Transform to process images captured by the Picamera. This method has shown to provide an automated system which is safe and cost effective as well. The dependence on lane markings on the roads is the major setback of this paper. Also real time implementation requires a more powerful computing machine to support multitude of sensors inputs and machine vision algorithms. Concepts from papers [9], [12], [7] have been combined to implement stop sign detection using haar cascade classifier.

The complex and unpredictable driving environment have made simple machine vision algorithms incapable because of the model's dependency on the complex training data for enhanced performance. Also the processing speed of the networks degrades as more input elements are added. This has led to development of deep learning algorithms like Convolutional neural networks for vehicular automations. The end to end learning approach as explained in the literature [1] has proven to be faster and more powerful as it eliminated the need for preprocessing of the input data and is also able to work on any roads irrespective of the lane markings. With only very minimal training data from humans, deep learning methods facilitate efficient working of vehicles in complex environments.

From the literature survey, it is evident that there is very less study on the algorithms that can be used for such automated systems. This project aims to throw some light on some of the possible methods and technologies automated vehicular systems can utilize.

PROPOSED SYSTEM

Block diagram

The proposed system consists of a processing unit composed of a Raspberry Pi and an auxiliary processing unit consisting of a PC. The processing unit is interfaced to the input and output units. Fig 2 shows the block diagram of this intelligent autonomous vehicle system.

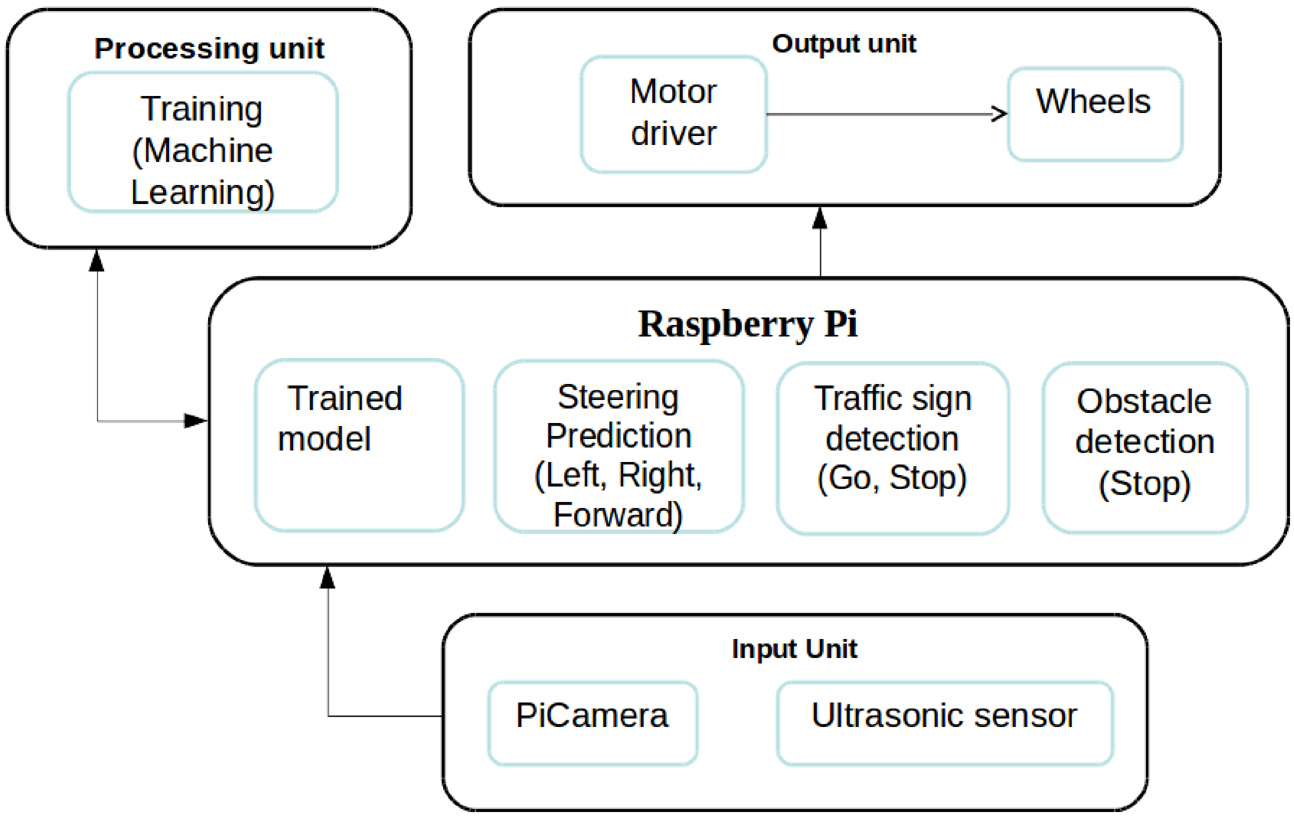

Processing unit
Training (Machine Learning)
Output unit
Motor driver
Wheels
Raspberry Pi
Trained model
Steering Prediction (Left, Right, Forward)
Traffic sign detection (Go, Stop)
Obstacle detection (Stop)
Input Unit
PiCamera
Ultrasonic sensor

Fig 2 Block Diagram

Input unit

The input unit consists of a Picamera and an ultrasonic sensor. The picamera provides images to the Raspberry Pi which it utilizes for both the training phase and the final autonomous driving phase. The camera essentially supplies with images of the track in our case. The ultrasonic sensor on the other hand is used to detect any obstacles in the path of the vehicle. It can also be used to find the distance between the vehicle and the obstacle in front. This information helps the vehicle to stop if there are objects or other vehicles in the track to avoid collision.

Processing unit

The processing unit is divided into a main unit and an auxiliary unit. The main unit is the Raspberry Pi which is mounted directly on the vehicle. In the training phase the RPi acts as a data collection unit to provide images and corresponding labels. Once the images have been collected, they are ported to the auxiliary unit through a suitable File Transfer Protocol (FTP) server-client system. The auxiliary unit is used to train a neural network model with the collected images and labels. The need for the auxiliary unit arises as the RPi cannot efficiently handle extremely computationally intensive tasks such as training of network models. Neural network training also requires quite an amount of experimentation and the auxiliary unit consisting of a PC can be used to achieve this effectively. Once the network is trained, the trained model weights are sent back to the Rpi. The Rpi can now be used in the testing and autonomous driving phase. In these phases, for the captured images, the network weights are used to predict the direction of motion in which the vehicle must proceed. The predicted value is then used to drive the wheels of the vehicle. If a stop sign is detected, the vehicle is halted. The Rpi also uses the information from the ultrasonic sensor to detect obstacles.

Output unit

The output unit is essentially composed of the motors attached to the wheels of the vehicle along with their driver. In the training phase, the motors are controlled by user input. In the testing and autonomous driving phase, the motors are controlled by the output of the neural network and require no manual intervention. The driver is used to supply necessary current to the motors and also helps in controlling the speed and direction of the motors.

MACHINE LEARNING

In a broad sense, Artificial Intelligence describes the intelligent computers which think and act like humans. Machine learning is a type of artificial intelligence where machines can learn from data without explicitly programmed instructions. Machine learning requires a huge amount of data. With the development in data management techniques like Big data management, cloud computing, machine learning has had an enormous growth in last few years. AI has brought revolutionary improvements in the computing field in terms of cost and efficiency leading to its application in all walks of life. It has been used in all search engines, automation, driver assistance technologies, SIRI, ALEXA, CORTANA etc. Artificial Neural Network is another form of Artificial Intelligence which is used in this project to make the vehicle learn to run on the track. Dr. Robert Hecht-Nielsen, who is the inventor of the first neuro computer, defines a neural network as “a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs”

ACTIVATION FUNCTIONS

Activation functions decide whether or not to activate a node, thereby mapping the signals from the input to the output nodes. The output signal from one node could be used as an input to nodes of other layers in the case of Multilayer neural networks. Without Activation function, a neural network would simply be a Linear regression Model**,** which does not perform well in the case of complicated datasets consisting of images, videos, audios which require complex non-linear mappings from inputs to outputs [25].

Since the activation functions are differentiable, they are used in Backpropagation network mechanism which computes and optimizes the error function by comparing the output values and the desired values. The main goal is to make a complex neural network learn better by introducing non-linear properties thereby making the network more powerful.

Activation functions vary in their range, threshold values, gradients and their performance vary with the neural networks. The proper choice of activation function is very essential as it has major impact on the training process.

TYPES OF ACTIVATION FUNCTIONS

By varying the activation functions, various neural network models were designed, trained and each of their performance was studied to determine the apt model for our application. The various activation functions are described as follows.

Sigmoid function

A sigmoid function represented by its characteristic S-shaped curve is differentiable, real and well defined for all input values. It maps the interval (-∞, ∞) onto (0,1) range. The sigmoid function is represented by

$$F(x) = \frac{1}{1+exp(-x)} \tag{3.1}$$

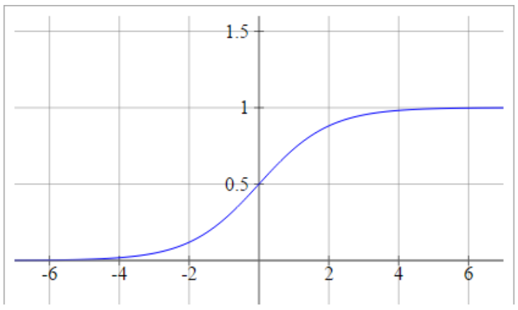


Fig 3 Sigmoid activation function

It is very advantageous to use this function in the networks with the backpropagation models because of its non-negative first derivative. One of the major drawbacks of the Sigmoid function is the vanishing gradient problem (i,e the gradients gets saturated and die) . The output is not zero-centered which makes the optimization difficult.

Hyperbolic tangent function

A Tanh function is the ratio of hyperbolic sine and cosine function having the range of (-1, 1). It is represented by

$$F(x) = \frac{1-exp\,(-2x)}{1+exp\,(-2x)} \tag{3.2}$$

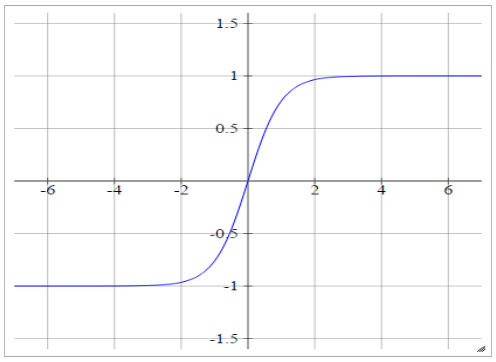


Fig 4 Tanh activation function

The outputs of Tanh functions are zero-centered which makes the optimization easier. Thus they are preferred over sigmoid functions. However, Tanh functions also suffer from vanishing gradient problem.

Rectified linear unit

ReLU is the most commonly used activation function in the neural networks [10]. It is defined as the positive part of the argument and it is expressed as

$$F(x) = \max(0, x) \tag{3.3}$$

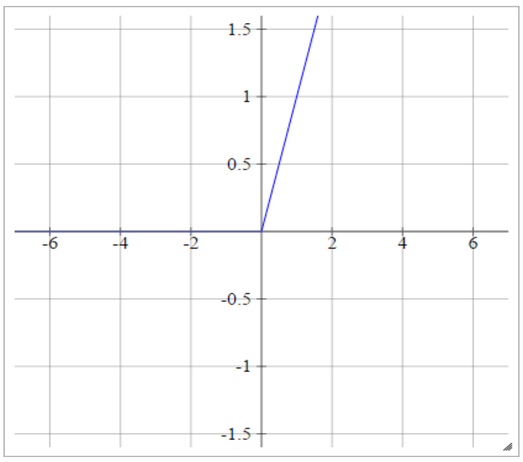


Fig 5 ReLU activation function

The ReLU activation is thresholded at zero and has a range from 0 to ∞. Since expensive functions like exponentials are avoided, ReLU function does not have vanishing gradient problem. It is proved that the convergence of gradient descent in ReLU is six times faster when compared to Tanh and sigmoid functions [13].

The major drawback in this activation function is that some of the neurons become fragile during the training, thus resulting in dead neurons. This problem could be minimized by adjusting the learning rates. However, ReLU function is restricted to the hidden layers in the neural network which explains the usage of Softmax functions along with the ReLU functions in most of the neural networks.

Softplus

Softplus activation function is a smooth approximation to the ReLU activation function. It is described as

$$F(x) = log(1 + exp(x)) \tag{3.4}$$

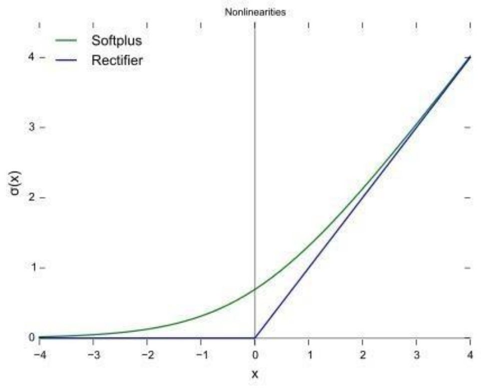


Fig 6 Softplus activation function

Softplus functions are smooth and easily differentiable at zero. However, ReLU is preferred to Softplus function because of its simple structure which enables fast and easy computations.

Leaky ReLu

Leaky ReLU activation function is obtained by modifying the ReLU function such that it has small negative value instead of zero for negative inputs. This eliminates the problem of dead neurons in the hidden layers. This function is expressed as

$$F(x) = \mathrm{x}, (\mathrm{x} \geq 0) \tag{3.5}$$

$$F(x) = \alpha\mathrm{x}, (\mathrm{x} < 0) \tag{3.6}$$

where α is a constant, (α<1).

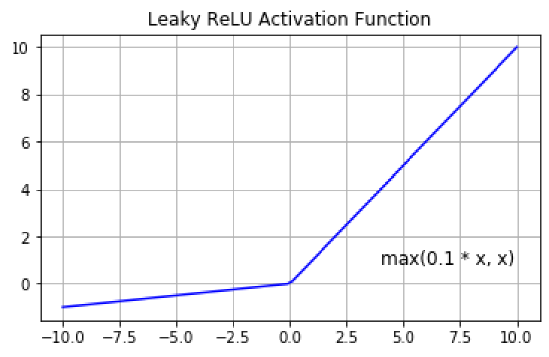


Fig 7 Leaky reLU activation function

The result of this activation function not being consistent at all times is its major setback. Learning rate has to be chosen carefully to prevent the neurons from being stuck at dead zones.

Softmax function

It is widely used in the outer layer of the neural network. It differs from all the other activation functions in terms of the number of classification of the output classes (i,e it can be used to classify even more than hundred different classes). The Softmax function is defined as ratio of the exponential of the input units to the sum of all exponentials of the input units [14]. The expression is given as

$$F(X_i) = \frac{exp\ (X_i)}{\sum_{j=0}^{k} exp\ (X_j)} \tag{3.2}$$

It calculates the probabilities of every output class and selects the target output based on the class with highest probability. The softmax function maps the inputs to the range of (0,1) similar to the sigmoid function and in addition to that it also divides the outputs such that the sum of all output probabilities are one thereby ensuring only one output is selected (i,e only one output class will have high probability).Thus the softmax functions are mostly used when there is the need for non-binary classifications

HARDWARE

The hardware components used for this system consists mainly of a Raspberry pi, a camera module, an ultrasonic sensor and a motor driver.

Raspberry PI

Raspberry Pi is a single board computer manufactured by the Raspberry Pi Foundation. The compact board, with dimensions 85.60 mm × 56.5 mm × 17 mm and og 46g weight, is very powerful and has proved to be of great use in small scale projects, home and industrial automation products, and in education. Several models have been release since the launch of the first generation model in 2013. The model used in this project is Rpi 3B. Launched in February 2016, this model is built on an ARMv8-A (64/32-bit) architecture with a Broadcom BCM2837 system on chip. The Central Processing Unit is a 64-bit quad-core ARM Cortex-A53 with a clocking frequency of 1.2 Ghz. The built in memory is about 1GB which is shared with the Graphical Processing Unit. The board also consists of a MicroSDHC slot in which SD cards can be mounted for extra storage. [18]

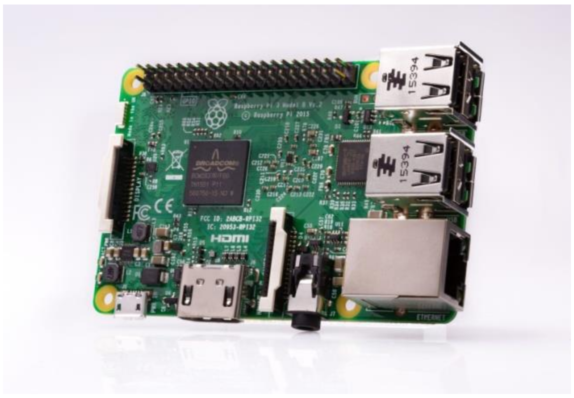


Fig 8 Raspberry Pi 3 Model B. Source: [18]

The board features four USB 2.0 ports which can be used to interface multiple USB devices at once. A HDMI port is provided to visualize the GPU. A 15-pin MIPI camera interface connector is also provided. This can be used along with the Raspberry Pi camera module for image and video capturing. The power rating of the board is around 1.5 W when idle and a maximum of 6.7 W under stress. The 3B board also has built provisions for 2.4 GHz WiFi 802.11n (150 Mbit/s) and Bluetooth 4.1. The entire board is to be supplied from a power source providing 2.1 A current at 5V. For remote uses, power banks can be used.

The board has a 40 GPIO pins which can be used to obtain input and provide output signals to other components and devices. Fig 9 shows the pin layout of the RPi 3B model.

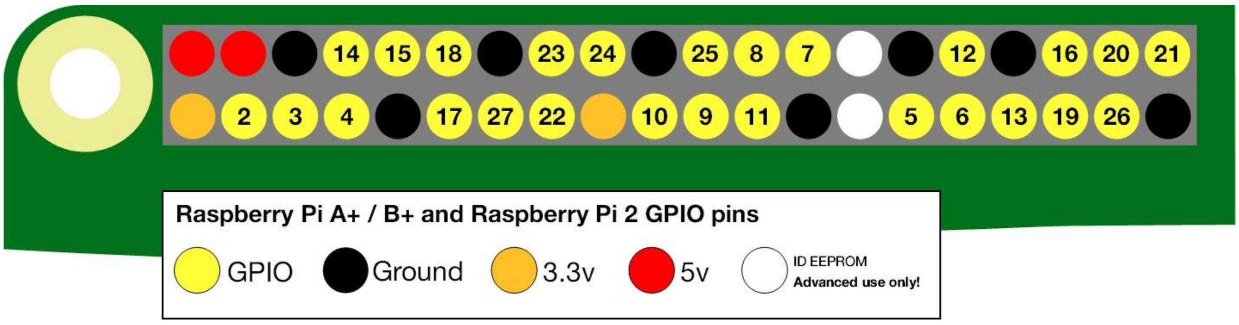


Fig 9 Pin layout of the RPi 3B model. Source: [19]

Camera module

The camera module used in this system is a Raspberry Pi Camera Module v2. The camera has an 8 Megapixel sensor [20] which captures high definition video and photos. The camera is attached via a 15cm ribbon cable to the CSI port on the Raspberry Pi. Fig 10 shows a Pi camera module attached to the RPi board.

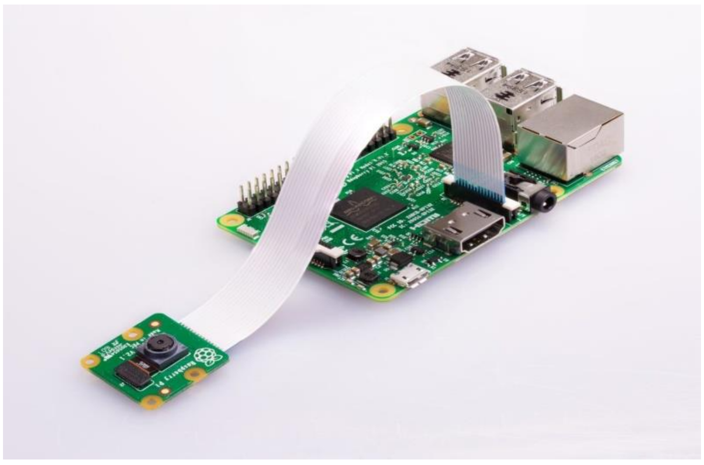


Fig 10 Pi camera module attached to the RPi board. Source [20]

The parameters of the Pi camera can be controlled using various libraries built for it. The python Picamera module has been used in this project.

Ultrasonic sensor

Ultrasonic sensors are used for ranging purposes. They consist of a transmitter and a receiver. The transmitter sends a high frequency ultrasonic sound, which on encountering obstacles, will be reflected back. The reflected pulse is detected by the receiver. The time difference between the sending of trigger pulse and the reception of the echo pulse can be used to determine the distance at which the obstacle is present. Fig 11 shows HC-SR04 which is a popular ultrasonic sensor module.

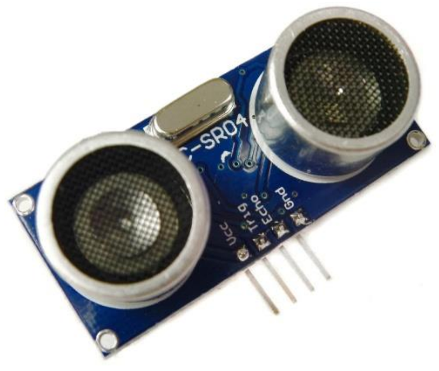
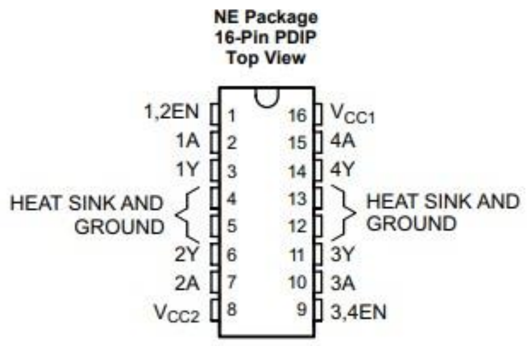
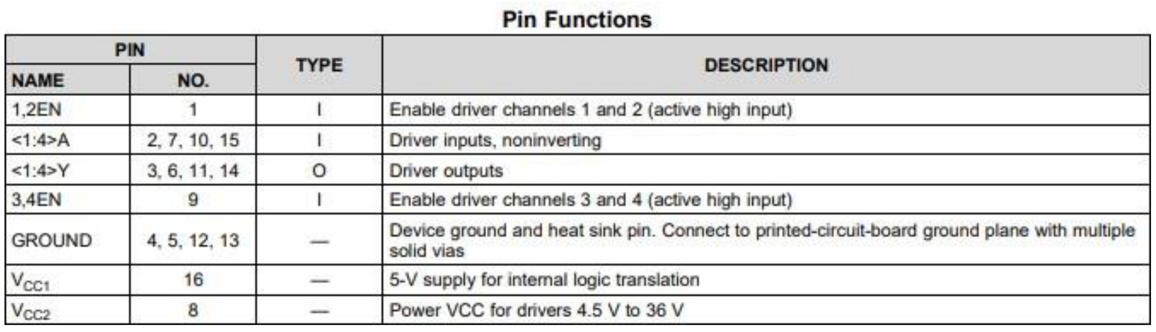


Fig 11 HC-SR04

Motor driver

Motors need to be driven with appropriate current. Generally the control signals from the RPi are not sufficient to drive the motors. There is also a need to alternate the voltage directions and the power to control direction and speed. For this purpose, a motor driver IC is used. These motor drivers consists of a H-bridge circuit which allows to alternate the voltage across the load. The motor driver takes input from a controller (RPi in this case) and uses the voltage from a battery to provide the necessary driving current to the motors. L293D is the motor driver IC which has been used in this system. This IC can control two motors separately. For each motor, the IC has an enable pin and two control inputs. The schematic of the IC is given in Fig 12.




Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

Fig 12 Schematic of L293D (Source:[23])

SOFTWARE

The entire project is programmed with Python language along with other supporting softwares and libraries that includes putty, VNC viewer, opencv, tensorflow.

Python and its packages

Python is a high level general purpose programming language developed by Python software foundation in 1991. Python supports a range of paradigms like Object Oriented, Functional, Structural programming paradigms wholly and several other paradigms with extensions. Python is used in various Operating Systems like UNIX, Linux, Windows and Mac OS. The language is easy to understand with simple features and elegant syntaxes that offer dynamic typing thereby making it very useful software for backend web development, data analysis, artificial intelligence, app development, graphic design applications, etc., [24]. The Python interpreter and the extensive standard libraries are available for free which adds to its popularity. Python 3.5 version is used in this project. This version features significant changes in standard library and syntaxes, with addition to new built features including some improvements in windows.

Tensorflow

Tensorflow is an open source library created by Google in November 2015 exclusively for machine learning and deep learning applications. This software library basically uses data flow graphs for fast numerical computations which help us build, train and optimize the neural network in an efficient way. The flexible architecture of tensorflow allows the computations to be deployed in CPUs, GPUs and also in smartphones. It works with C++, Python 2.7, Python 3.3+ versions. Google uses tensorflow in its wide range of applications like Gmail, maps, Google play, Google translate etc. The data is represented in the form of tensors that are nothing but multidimensional arrays.

The process includes creating the computational graphs in which inputs and outputs are Tensors and the nodes are called “operations,” or “ops.” Each node represents a mathematical operation that is performed on the tensors [24].

Primarily, all the values are to be initialized either as constants or variables. Constants are used when the values do not change and variables are used when the parameters demands updated values. Placeholder is used to define these variables. After the data is initialized, it is subjected to evaluation. The Tensorflow then runs a session in which all the nodes in computational graph are evaluated and the results are obtained.

With the recent release from TensorBoard, a data visualization toolkit that enables a better understanding of all the data flow graph operations has been made possible. Tensorboard provides summary of node operations and various parameters apart from the graphic visualizations of the tensorflow programs. All these features of the tensorboard make it easy to understand, debug and optimize complex multi-layer neural networks.

Putty

PUTTY is an open-source client program which is used to remotely connect to systems or servers. It supports various network protocols like SSH, telnet, Rlogin. Putty is mainly used in the Windows and Unix platform since it doesnot have an SSH client on its own. In our project, Raspberry Pi is accessed from a windows system with the help of Putty and SSH which provides a secure connection on the internet. The computers that desires an access to the Raspberry Pi has to be connected to the same network as that of the Pi [21].

Once the Putty software is installed in a windows machine, a configuration window appears in which hostname (i,e name of the server that should be accessed) or its IP address and the type of the protocol has to be specified. The Raspberry has to be selected as a host and SSH protocol is used for logging in after which Pi can be accessed remotely.

VNC Viewer

Virtual Network Computing refers to the desktop sharing facility that is based on Remote Frame Buffer protocol which offers remote access and control over other systems. This open source software is employed in many applications that require remote access of files, home computer networks, remote troubleshooting and system support for customers and employees. VNC is a client server protocol where VNC server must be installed in the machine which is to be accessed and VNC client must be installed in the machine that would control the server.

VNC Viewer is client server software based on VNC protocol developed by the REAL VNC company which works on Windows, Unix, Mac systems. Raspberry Pi is accessed and controlled on a Windows system using VNC Viewer over a network (i,e the raspberry pi desktop screen in replicated in our Pc). This offers various advantages as it eliminates the need for separate keyboard and monitor for the Raspberry Pi [22].

The first step in establishing the connection requires downloading and installation of VNC client in Windows PC. Then the VNC server, installed on the Pi has to be configured in such a way that the server automatically starts to boot up. This can be done with the configuration settings by enabling the VNC given under the interfacing options menu.

Once these initial setups have been completed, the IP address of the Raspberry Pi, username and password has to be entered in the VNC Viewer which makes client PC to serve as the Raspberry Pi's monitor.

RESULTS AND DISCUSSION

PHASE ONE

In the phase one of the project, neural network was trained to classify the images fed into it as left and right. All the processes were done using a computer in Python. The process and the results are covered in the subsequent sections.

Data collection

Modelling of any neural network starts with the collection of data and its labels. Dataset was created with the images collected from an online car simulator. Every image shows a graphical car taking either left or right turn. This dataset consists of 52 images in which 40 images were used for training the network and 12 were test images.

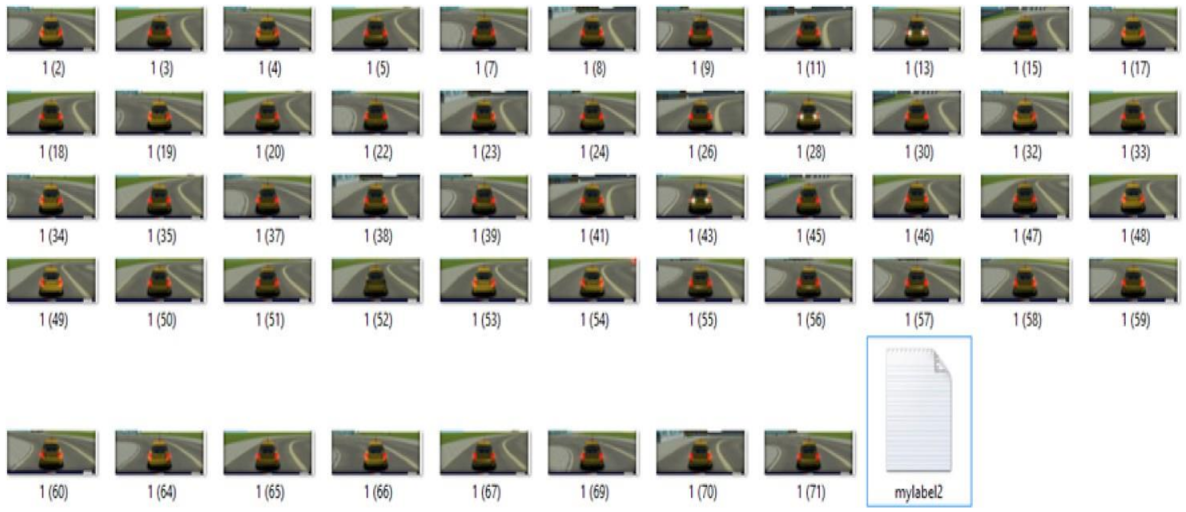


Fig 13 Dataset and label file

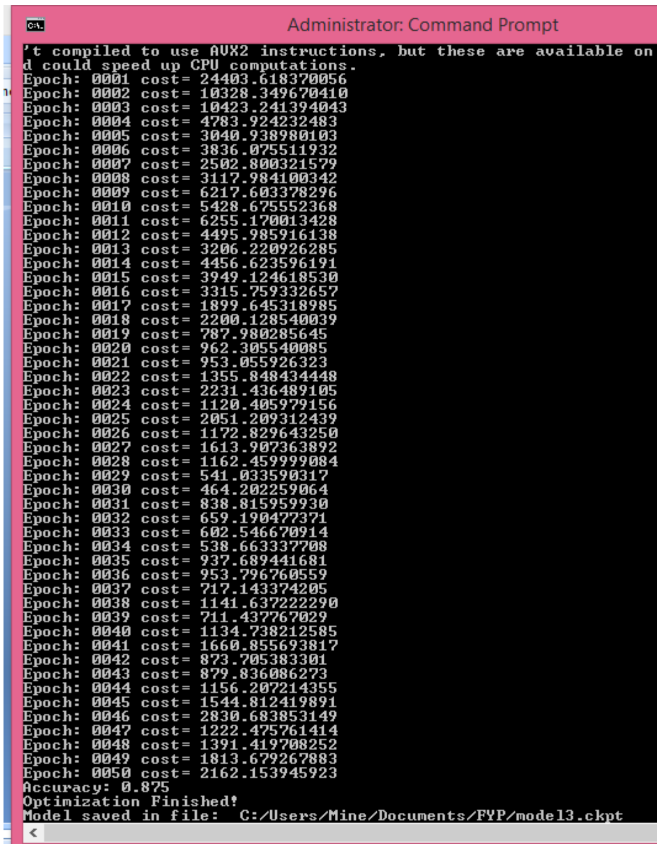
Fig 13 shows the dataset consisting of 52 images and the label file which is used to train the neural network. This neural network predicts two directions, left and right namely and hence the labels are assigned as ‘0’ for left direction and ‘1’ for the right direction. All the images were labelled accordingly in a text document. This file is the label file for this model.

Training and testing

The images and its associated labels were first converted into numpy format before it were fed to the neural network. The images were resized and normalized to make the network perform well under different illuminations.

The specifications of the neural network model are given below.

- Input nodes= 75840 (No. of pixels)
- Hidden Layer 1 nodes= 256
- Hidden Layer 2 nodes= 256
- Output nodes= 2 (No. of labels)
- Learning Rate= 0.01
- Batch Size= 5
- Activation= RELU
- Optimizer= Adam’s



```
't compiled to use AVX2 instructions, but these are available on
d could speed up CPU computations.
Epoch: 0001 cost= 24403.618370056
Epoch: 0002 cost= 10328.349670410
Epoch: 0003 cost= 10423.241394043
Epoch: 0004 cost= 4783.924232483
Epoch: 0005 cost= 3040.938980103
Epoch: 0006 cost= 3836.075511932
Epoch: 0007 cost= 2502.800321579
Epoch: 0008 cost= 3117.984100342
Epoch: 0009 cost= 6217.603378296
Epoch: 0010 cost= 5428.675552368
Epoch: 0011 cost= 6255.170013428
Epoch: 0012 cost= 4495.985916138
Epoch: 0013 cost= 3206.220926285
Epoch: 0014 cost= 4456.623596191
Epoch: 0015 cost= 3949.124618530
Epoch: 0016 cost= 3315.759332657
Epoch: 0017 cost= 1899.645318985
Epoch: 0018 cost= 2200.128540039
Epoch: 0019 cost= 787.980285645
Epoch: 0020 cost= 962.305540085
Epoch: 0021 cost= 953.055926323
Epoch: 0022 cost= 1355.848434448
Epoch: 0023 cost= 2231.436489105
Epoch: 0024 cost= 1120.405979156
Epoch: 0025 cost= 2051.209312439
Epoch: 0026 cost= 1172.829643250
Epoch: 0027 cost= 1613.907363892
Epoch: 0028 cost= 1162.459999084
Epoch: 0029 cost= 541.033590317
Epoch: 0030 cost= 464.202259064
Epoch: 0031 cost= 838.815959930
Epoch: 0032 cost= 659.190477371
Epoch: 0033 cost= 602.546670914
Epoch: 0034 cost= 538.663337708
Epoch: 0035 cost= 937.689441681
Epoch: 0036 cost= 953.796760559
Epoch: 0037 cost= 717.143374205
Epoch: 0038 cost= 1141.637222290
Epoch: 0039 cost= 711.437767029
Epoch: 0040 cost= 1134.738212585
Epoch: 0041 cost= 1660.855693817
Epoch: 0042 cost= 873.705383301
Epoch: 0043 cost= 879.836086273
Epoch: 0044 cost= 1156.207214355
Epoch: 0045 cost= 1544.812419891
Epoch: 0046 cost= 2830.683853149
Epoch: 0047 cost= 1222.475761414
Epoch: 0048 cost= 1391.419708252
Epoch: 0049 cost= 1813.679267883
Epoch: 0050 cost= 2162.153945923
Accuracy: 0.875
Optimization Finished!
Model saved in file:  C:/Users/Mine/Documents/FYP/model3.ckpt
```

Fig 14 Training process

The training processes were repeated until an optimal accuracy was achieved. Fig 14 shows the training process of the network. Its noted that,as the neural network learns, the cost function (i,e error function) keeps decreasing as it reassigns the weights based on the error function. This process was repeated for the specified number of epochs and a model with 87.5% accuracy was obtained.

Prediction

Prediction results provide an insight into the performance of the neural networks. In the prediction program, the trained model was incorporated into a network having the same specifications as the training program has had. A new set of images (i.e images that were not used in the training process) were used to test the trained model.

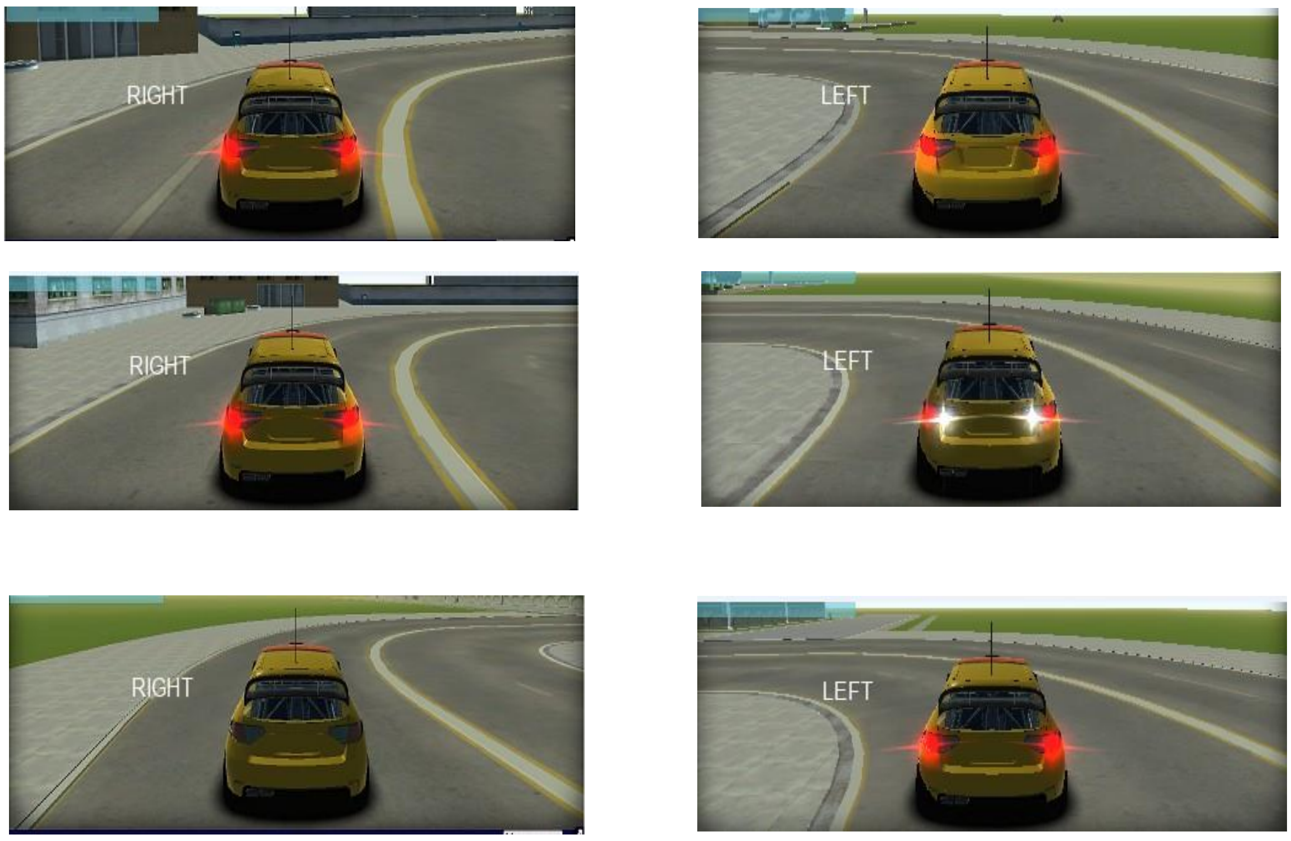


Fig 15 Correct predictions

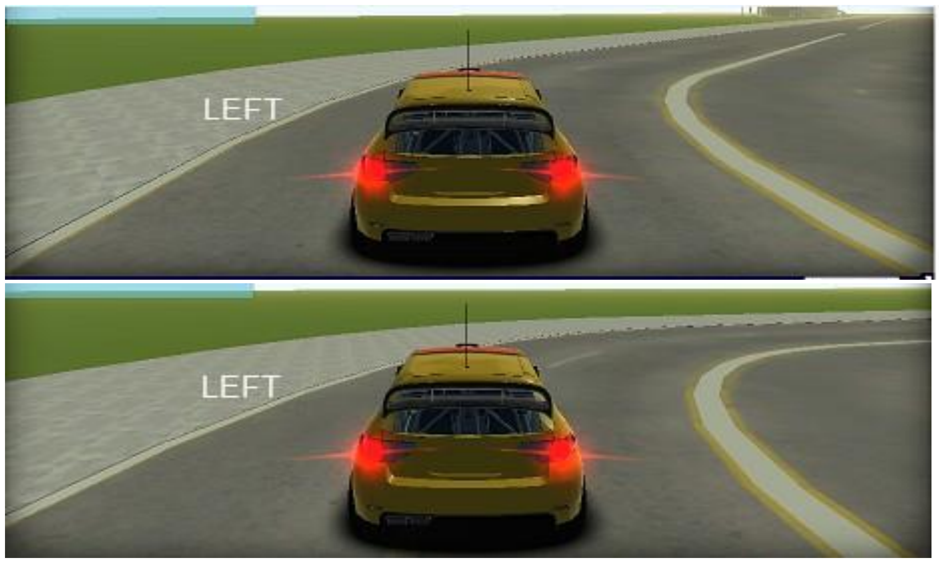


Fig 16 Wrong predictions

Fig 15 and Fig 16 shows the prediction results of the eight test images. The performance of the network is found to be good with six correct predictions and only two wrong predictions. This model can be further optimized and trained better with more number of training images.

PHASE TWO

In the second phase of the project, a neural network was been trained to classify the images taken by a robotic car. The trained network model was then used to drive the car autonomously. The robotic car was set up by connecting the Raspberry pi to the motors mounted in a chassis through L293D. The Input and enable pins were connected to GPIO pins in the raspberry pi and where programmed to turn left, right and forward on command. A Pi camera was also attached to the raspberry pi and sturdily mounted on top of it. The HC-SR04 module was also interfaced to Rpi through a voltage divider circuit. A power bank is used to power the Rpi and a 9V battery is used in the interface of Pi with L293D. Fig 17 shows images of the robotic car. The various processes performed and the results are covered in the subsequent sections.

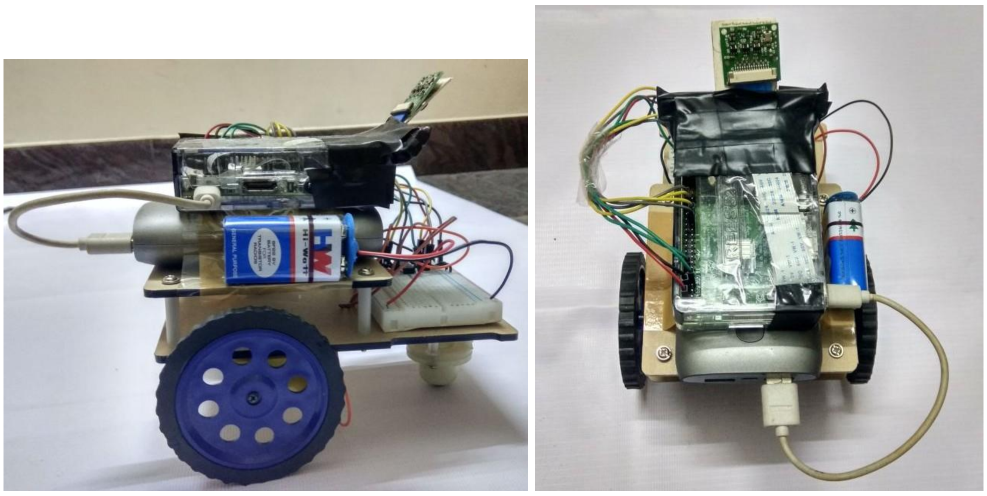


Fig 17 The hardware set-up

Data collection

A new portable track was designed on a flex sheet on which the car was made to learn to drive by it. Two parallel strips of two centimeters each represent the boundary of the S shaped curve having left and right turns.

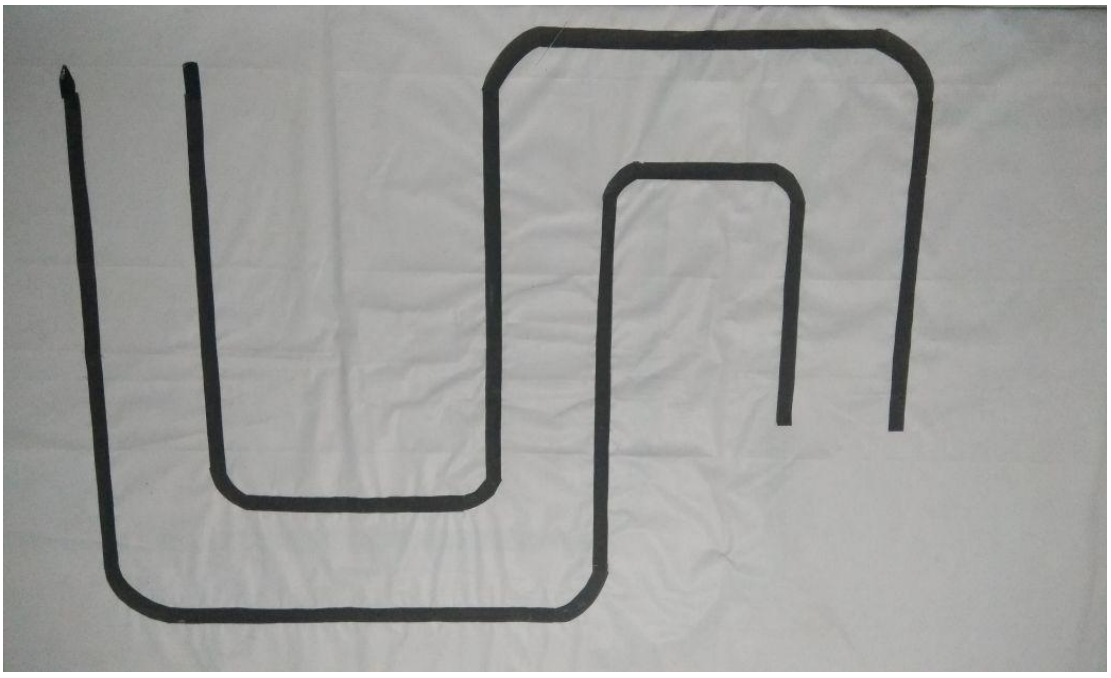
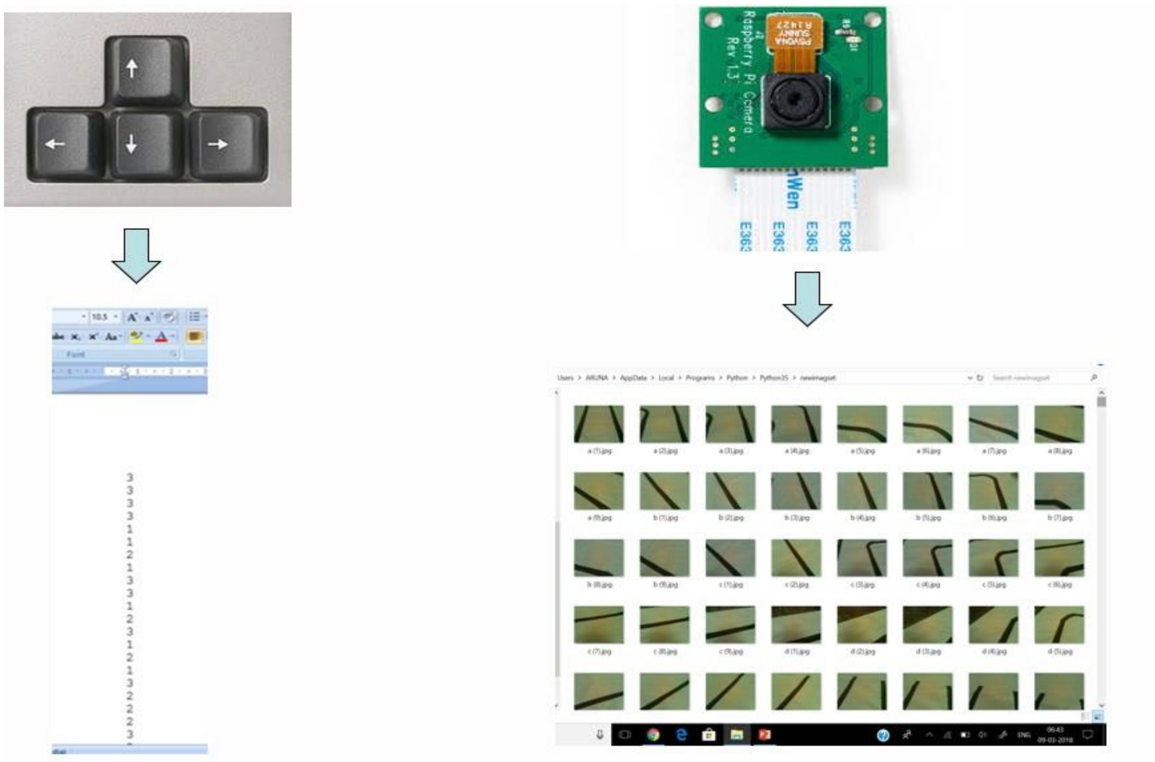


Fig 18 Image of the track

Images of the track were collected by manually driving the car on the track. The data collection process is shown in the Fig 18.



3
3
3
3
1
1
2
1
3
3
1
2
3
1
2
1
3
2
2
2
3

Fig 19 Training data collection

The car was made to move on a key press. For every step, an image of the track was captured by the Picamera and was saved in a folder along with its appropriate label which was saved in a text file. Three labels namely 0, 1 and 2 for forward, left and right directions respectively were used for this network.

About 600 images were collected by driving the car several times on the track under various illuminations. By using a data augmentation technique called flipping, another 600 images of the track were obtained.

The dataset was then balanced by assigning an equal number of images for all the three output classes (i,e left, right and forward). Dataset consisted of 1392 images in which a total of 1248 images were used for training the neural network with each output class consisting of 416 images and the remaining 144 images were used for testing the network.

Comparison and optimisation of neural network models

A variety of neural network models were designed using the different activation functions described in the chapter 3. The models were also optimized by reducing the number of input nodes and hidden layers. The performance of each of the models was studied in order to choose the best model for this application.

The specifications for the neural network models with different activation functions are given below.

Hidden Layers=2 Optimizer= Adam's Learning Rate= 0.01
Input nodes= 786 (No. of pixels) Output nodes= 3 (No. of labels)

The activation functions were varied only the hidden layer whereas the output layer had softmax activation. The neural networks were first trained and then the trained models were tested for its performances using new sets of images called prediction test images. A set of 21 images captured under different illuminations were chosen for testing all the models.

Table 1 Comparison of two layered neural networks with different activation functions

HIDDEN LAYER NODES	**ACTIVATION FN**	**ACCURACY**
512	ReLU	66
256	ReLU	67.77
256	LeakyreLU	77
256	LeakyreLU	71.42
256	Tanh	85.71
256	Sigmoid	90.47
256	Softmax	95.21

Table 1 shows the following. Network with reLU activation function had the lowest accuracy of 66% for test image set of 144 images. It shows that the model is biased towards two of the three outputs (i,e model predicted only forward and right directions correctly). The results were similar when number of hidden layer nodes was reduced from 516 to 256. Hence for all other models 256 hidden layer nodes were used in order to reduce the prediction time. The model with the leaky reLU function provided slightly better results than the reLU model. With an accuracy of 77%, the model predicted most of the forward and right directions and very few left directions. Models with activation functions like tanh and sigmoid provided a very good accuracy of 85% and 90% respectively. Both these models predicted 19 of the 21 test images correctly.

Maximum accuracy of 95.2% with extremely good prediction results was achieved in the neural network with softmax function.

Table 2 Comparison of neural networks with reduced hidden layer nodes

HIDDEN LAYER NODES	**ACTIVATION FN**	**ACCURACY**
32	Tanh	84.71
32	Sigmoid	90.47
32	Softmax	95

The models were further optimized by reducing the hidden layer nodes to minimize the processing time. For this the models from the table 1 having accuracy more than 75% were chosen. Table 2 shows that the accuracy of the models having 32 nodes was almost equal to the models with 256 nodes. Further reduction in hidden layer nodes provided poorer results as the models were able to predict only one of the directions correctly. From the above discussion the optimal two layered neural network model is chosen to be the one with softmax function having 32 hidden layer nodes.

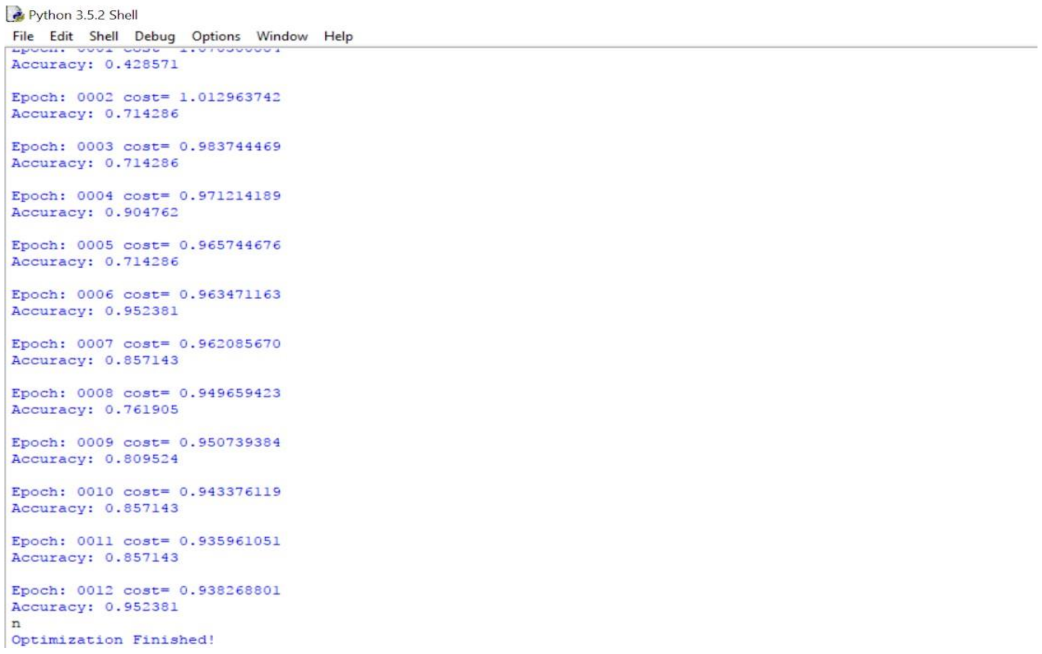
The next level of optimization was done in the hidden layers. The neural networks were implemented with a single hidden layer and their performances were compared with the networks having two hidden layers to analyze if better results could be achieved with lesser complexity of the neural networks.

Table 3 Comparison of single layered neural networks with different activation functions

HIDDEN LAYER NODES	**ACTIVATION FN**	**ACCURACY**
32	softplus	69
32	relu	71.42
32	leakyreLU	85.71
32	tanh	90.23
32	sigmoid	95.23

Starting with 512 hidden layer nodes, several single layered models were trained and the prediction results were subnormal. The training and optimization processes were repeated until certain accuracy is achieved.

The comparison of single layered networks with 32 hidden layer nodes is given in the table 3. With an accuracy of only 30% Softplus model performed the worst as it was able to predict only one direction (i,e forward direction). The reLU model was yet again an average accuracy model with prediction results being biased to two of the directions. Neural networks with leaky reLU and tanh provided higher accuracy with better prediction results as well. Applying sigmoid in the hidden layer resulted in a neural network with desired accuracy of over 90%. The training of the sigmoid model is shown in the Fig 20.

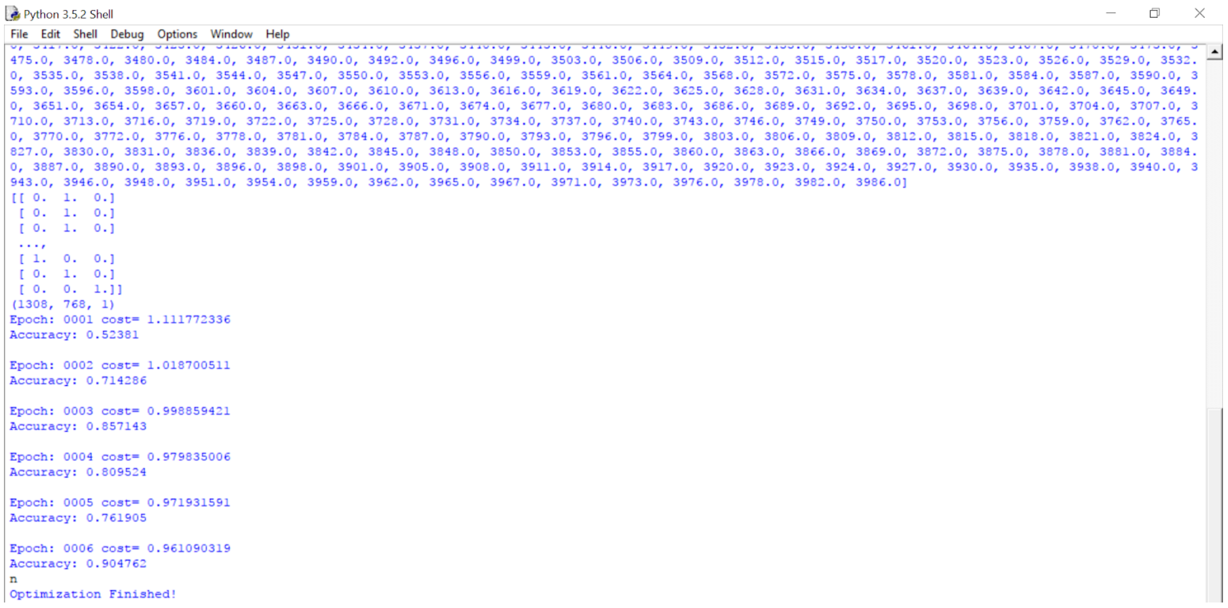


```
Accuracy: 0.428571

Epoch: 0002 cost= 1.012963742
Accuracy: 0.714286

Epoch: 0003 cost= 0.983744469
Accuracy: 0.714286

Epoch: 0004 cost= 0.971214189
Accuracy: 0.904762

Epoch: 0005 cost= 0.965744676
Accuracy: 0.714286

Epoch: 0006 cost= 0.963471163
Accuracy: 0.952381

Epoch: 0007 cost= 0.962085670
Accuracy: 0.857143

Epoch: 0008 cost= 0.949659423
Accuracy: 0.761905

Epoch: 0009 cost= 0.950739384
Accuracy: 0.809524

Epoch: 0010 cost= 0.943376119
Accuracy: 0.857143

Epoch: 0011 cost= 0.935961051
Accuracy: 0.857143

Epoch: 0012 cost= 0.938268801
Accuracy: 0.952381
n
Optimization Finished!
```

Fig 20 Training process of two layered sigmoid model

Prediction

Before implementing the trained neural network to drive the car autonomously on the track, its performance was tested on the computer using a prediction algorithm. The chosen model with sigmoid function provided extremely good results by predicted all the test images correctly. Fig 21 shows the model's correct predictions of all the directions.



```
475.0, 3478.0, 3480.0, 3484.0, 3487.0, 3490.0, 3492.0, 3496.0, 3499.0, 3503.0, 3506.0, 3509.0, 3512.0, 3515.0, 3517.0, 3520.0, 3523.0, 3526.0, 3529.0, 3532.
0, 3535.0, 3538.0, 3541.0, 3544.0, 3547.0, 3550.0, 3553.0, 3556.0, 3559.0, 3561.0, 3564.0, 3568.0, 3572.0, 3575.0, 3578.0, 3581.0, 3584.0, 3587.0, 3590.0, 3
593.0, 3596.0, 3598.0, 3601.0, 3604.0, 3607.0, 3610.0, 3613.0, 3616.0, 3619.0, 3622.0, 3625.0, 3628.0, 3631.0, 3634.0, 3637.0, 3639.0, 3642.0, 3645.0, 3649.
0, 3651.0, 3654.0, 3657.0, 3660.0, 3663.0, 3666.0, 3671.0, 3674.0, 3677.0, 3680.0, 3683.0, 3686.0, 3689.0, 3692.0, 3695.0, 3698.0, 3701.0, 3704.0, 3707.0, 3
710.0, 3713.0, 3716.0, 3719.0, 3722.0, 3725.0, 3728.0, 3731.0, 3734.0, 3737.0, 3740.0, 3743.0, 3746.0, 3749.0, 3750.0, 3753.0, 3756.0, 3759.0, 3762.0, 3765.
0, 3770.0, 3772.0, 3776.0, 3778.0, 3781.0, 3784.0, 3787.0, 3790.0, 3793.0, 3796.0, 3799.0, 3803.0, 3806.0, 3809.0, 3812.0, 3815.0, 3818.0, 3821.0, 3824.0, 3
827.0, 3830.0, 3831.0, 3836.0, 3839.0, 3842.0, 3845.0, 3848.0, 3850.0, 3853.0, 3855.0, 3860.0, 3863.0, 3866.0, 3869.0, 3872.0, 3875.0, 3878.0, 3881.0, 3884.
0, 3887.0, 3890.0, 3893.0, 3896.0, 3898.0, 3901.0, 3905.0, 3908.0, 3911.0, 3914.0, 3917.0, 3920.0, 3923.0, 3924.0, 3927.0, 3930.0, 3935.0, 3938.0, 3940.0, 3
943.0, 3946.0, 3948.0, 3951.0, 3954.0, 3959.0, 3962.0, 3965.0, 3967.0, 3971.0, 3973.0, 3976.0, 3978.0, 3982.0, 3986.0]
[[ 0.  1.  0.]
 [ 0.  1.  0.]
 [ 0.  1.  0.]
 ...,
 [ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
(1308, 768, 1)
Epoch: 0001 cost= 1.111772336
Accuracy: 0.52381

Epoch: 0002 cost= 1.018700511
Accuracy: 0.714286

Epoch: 0003 cost= 0.998859421
Accuracy: 0.857143

Epoch: 0004 cost= 0.979835006
Accuracy: 0.809524

Epoch: 0005 cost= 0.971931591
Accuracy: 0.761905

Epoch: 0006 cost= 0.961090319
Accuracy: 0.904762
n
Optimization Finished!
```

Fig 21 Training process of single layered sigmoid model

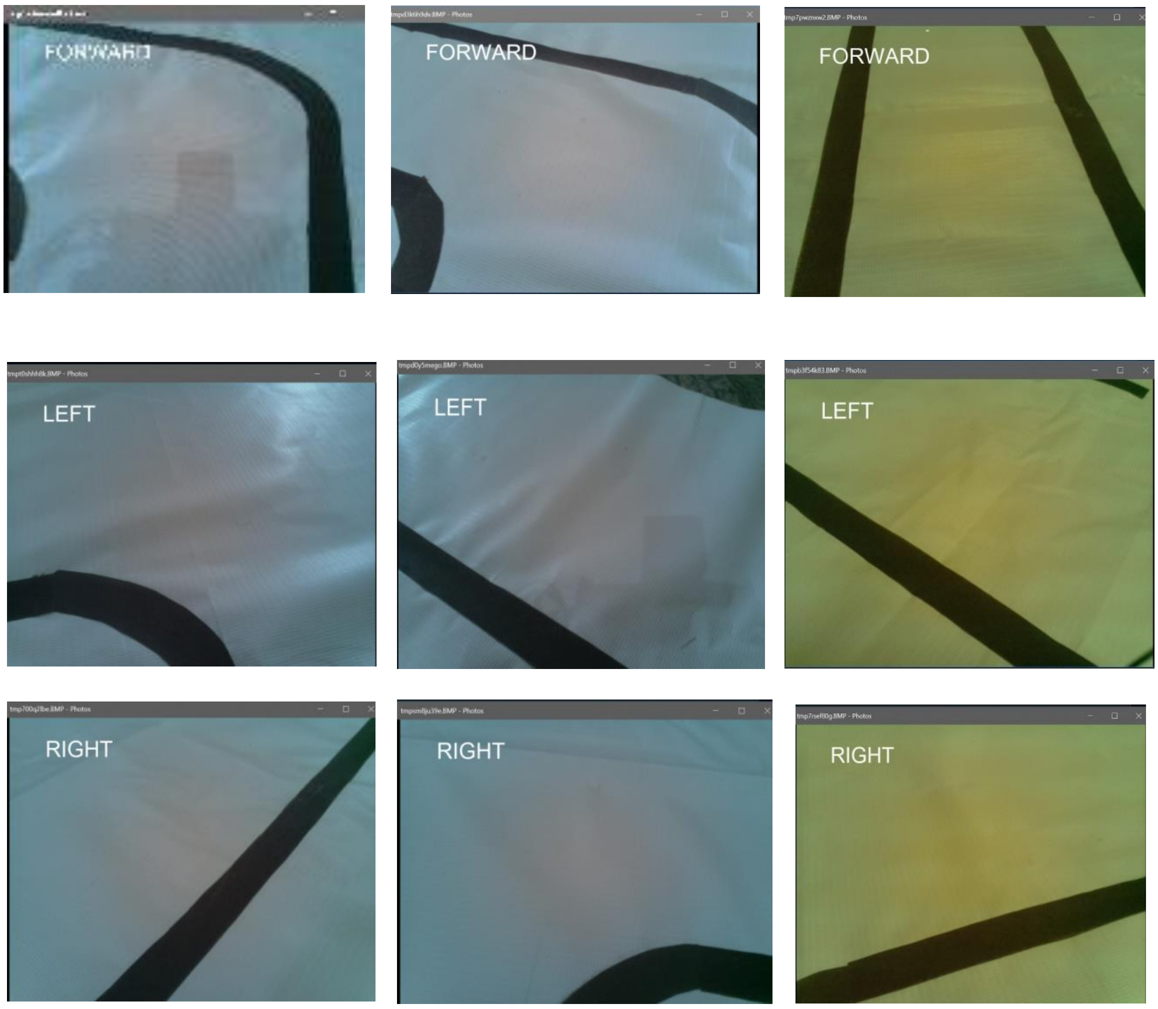


Fig 22 Direction prediction using sigmoid model

From the above discussions it is concluded that the model with the sigmoid activation function which performed exceptionally well in both the cases of single and double layered networks would is the apt model to be used to test the car on the track given in the Fig 17.

CONCLUSION AND FUTURE ENHANCEMENTS

In this project, a prototype of an intelligent autonomous vehicle was developed. The vehicle has been able to navigate in the trained track and detect stop signs and obstacles. As a future extension, the speed of the processing could be optimized by better choice of processor and also through multithreading techniques. Deep learning algorithms can be used for better accuracy on smaller dataset. The training can also be made diverse and more robust by adding images from various tracks under different circumstances. Finally, a multitude of other sensors can be integrated through sensor fusion to improve the robustness of the decisions taken by the vehicle.

REFERENCES

[1] Bojarski, Mariusz, et al. (2016) ‘End to end learning for self-driving cars’, NVDIA Corporation, arXiv:1604.07316v1.

[2] Cho, H., Seo, Y. W., Kumar, B. V., & Rajkumar, R. R. (2014) ‘A multi-sensor fusion system for moving object detection and tracking in urban driving environments’, 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 1836-1843.

[3] Daily, M., Medasani, S., Behringer, R., & Trivedi, M. (2017) ‘Self-Driving Cars’, Computer, Vol 50(12), pp.18-23.

[4] Hossai, M. R. T., Shahjalal, M. A., & Nuri, N. F. (2017) ‘Design of an IoT based autonomous vehicle with the aid of computer vision’, International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 752-756.

[5] Khan, B. S., Hanafi, M., & Mashohor, S. (2015) ‘Automated road marking detection system for autonomous car’, 2015 IEEE Student Conference on Research and Development (SCOReD), pp. 398-401.

[6] Kichun Jo, Junsoo Kim, Dongchul Kim, Chulhoon Jang, and Myoungho Sunwoo. (2015) ‘Development of Autonomous Car—Part II: A Case Study on the Implementation of an Autonomous Driving System Based on Distributed Architecture’, IEEE Transactions On Industrial Electronics, Vol. 62(8), pp.5**1** 9-5132.

[7] Lorsakul, A. and Suthakorn, J., (2007) ‘Traffic sign recognition using neural network on OpenCV: Toward intelligent vehicle/driver assistance system’, 4th International Conference on Ubiquitous Robots and Ambient Intelligence, pp. 22-24.

[8] Milanés, V., Llorca, D.F., Vinagre, B.M., González, C. and Sotelo, M.A., (2010) ‘Clavileño: Evolution of an autonomous car’, Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference, pp. 1129-1134.

[9] Mogelmose, A., Trivedi, M. M., & Moeslund, T. B. (2012) ‘Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey’, IEEE Transactions on Intelligent Transportation Systems, Vol.13(4), pp.1484-1497.

[10] Ramachandran, P., Zoph, B., & Le, Q. V. (2018), ‘Searching for activation functions’, arXiv:1710.05941.

[11] Ranft, B., & Stiller, C. (2016) 'The role of machine vision for intelligent vehicles', IEEE Transactions on Intelligent Vehicles, Vol.1(1), pp.8-19.

[12] Shopa, P., Sumitha, N., & Patra, P.S. K. (2014) 'Traffic sign detection and recognition using OpenCV', 2014 International Conference on Information Communication and Embedded Systems (ICICES), pp. 1-6.
[13] http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

[14] http://dataaspirant.com/2017/03/07/difference-between-softmax-function
-and-sigmoid-function

[15] https://www.digitaltrends.com/cars/history-of-self-driving-cars-milestones/

[16] https://www.forbes.com/sites/jenniferhicks/2017/09/11/how-to-make- autonomous-cars-see-better/#436b998c14fe
[17] https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[18] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[19] https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/RE ADME.md

[20] https://www.raspberrypi.org/products/camera-module-v2/

[21] https://www.raspberrypi.org/documentation/remote-access/ssh/windows.md

[22] https://www.raspberrypi.org/magpi/vnc-raspberry-pi/

[23] http://www.ti.com/lit/ds/symlink/l293.pdf

[24] https://www.toptal.com/machine-learning/tensorflow-machine-learning-tut orial

[25] https://towardsdatascience.com/activation-functions-and-its-types-which-is

-better-a9a5310cc8f

[26] https://www.wired.com/brandlab/2016/03/a-brief-history-of-autonomous- vehicle-technology/