A Project Report

On

SELF DRIVING CAR

BY

CHANUMOLU HIMAJA (15XJ1A0211)

KONDAPANENI LOKESH (15XJ1A0224)

MADADI NITHIN KODAND REDDY (15XJ1A0225)

JAHNAVI VIKRAMA (15XJ1A0520)

Under the supervision of

Dr. GOPINATH GR

SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF

SE 422: PROJECT TYPE COURSE

Mahindra École Centrale
COLLEGE OF ENGINEERING

LEADER ■ ENTREPRENEUR ■ INNOVATOR

MAHINDRA ECOLE CENTRALE

COLLEGE OF ENGINEERING

HYDERABAD

(MAY 2019)

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to our supervisor Dr. Gopinath GR for providing his valuable guidance, comments and suggestions throughout the course of the project.

We specially thank Dr. Bharghava Rajaram for his valuable guidance and encouragement.

Mahindra
École Centrale
COLLEGE OF ENGINEERING

Mahindra Ecole Centrale

Hyderabad

Certificate

This is to certify that the project report entitled "**SELF DRIVING CAR**" submitted by Ms. Chanumolu Himaja (ID No. 15XJ1A0211), Mr. Lokesh Kondapaneni (ID No. 15XJ1A0224), Mr. Nithin Kumar Madadi (ID No. 15XJ1A0225), Ms. Jahnavi Vikrama (ID No. 15XJ1A0520) in partial fulfillment of the requirements of the course SE421/SE422, Project Course, embodies the work done by him/her under my supervision and guidance.

Dr. Gopinath GR	**(EXTERNAL NAME & Signature)**
Mahindra Ecole Centrale, Hyderabad.	Mahindra Ecole Centrale, Hyderabad.
Date:	Date :

ABSTRACT

In the modern era, the vehicles are focused to be automated to give human driver relaxed driving. In the field of automobile various aspects have been considered which makes a vehicle automated. Google, the biggest network has started working on the self-driving cars since 2010 and still developing new changes to give a whole new level to the automated vehicles. In our project we are going to build a laboratory prototype of self driving car. It is controlled by raspberry pi micro controller. The best path between the source and the destination is chosen from Google maps. Several points are taken on the path and the self driving car moves form one point to another point, finally reaching the destination. Google maps provide the geographical coordinates of the points on the path chosen. These geographical coordinates are converted into Cartesian coordinates. This report presents an algorithm for self driving car movement from source point to the destination point using Cartesian coordiantes of the points on the path. In addition to this we implemented a CNN inspired from the NVIDIA model that performs deep neural networks feature extraction with convolutional neural networks as well as continuous regression to predict the angle of turning.

CONTENTS

CHAPTER 1: INTRODUCTION

Human Beings are imperfect, we make mistakes while driving thus in order to solve problems such as accidents we can automate a car without human intervention. A self-driving car, also known as a robot car, autonomous car, auto, or driverless car is a vehicle that is capable of sensing its environment and moving with little or no human input.

A robot is a programmable machine, able to extract information from its surroundings using different kinds of sensors to plan and avoid collisions without human interventions. One of the crucial parts of the design is the navigation. The navigation system generates control commands based on the code given to the microcontroller. The robot uses sensors to take appropriate actions like changing the direction of motion. There are many ways a self-driving car can perceive their surroundings such as computer vision, Lidar, GPS etc. Driverless vehicles require some form of machine vision for the purpose of visual object recognition. Automated cars are being developed with deep neural networks, a type of deep learning architecture with many computational stages, or levels, in which neurons are simulated from the environment that activates the network. The neural network depends on an extensive amount of data extracted from real-life driving scenarios, enabling the neural network to "learn" how to execute the best course of action.

The challenge for driverless car designers is to produce control systems capable of analyzing sensory data in order to provide accurate detection of other vehicles and the road ahead. There are many more benefits for implementing them which include reduced costs, increased safety, increased mobility, increased customer satisfaction, and reduced crime.

Alongside the many technical challenges that autonomous cars face, there exist many human and social factors that may impede upon the wider uptake of the technology. As things become more automated, the human users need to have trust in the automation, which can be a challenge in itself. Weather may also affect the car sensors, which may affect the auto driving mode. When heavy rainfall, heavy storms, heavy snowfall affect or damage the car sensors. Because of this, the performance of the car decreases gradually and the chance of accidents increases.

CHAPTER 2: BACKGROUND AND RELATED WORK

The most deeply involved player in the autonomous automobile market from outside of the automobile industry is Google. Google has worked to develop autonomous cars for the past ten years.

Both existing players and new entrants rely on a number of distance measurement and satellite positioning technologies. These are summarized in Table 1.

Significant investments are ongoing to develop solutions to address current limitations of these technologies. For example, Continental is evaluating the use of artificial intelligence to increase the car's ability to extrapolate a reaction from current conditions. Governments are investing in digital GPS (DGPS), and Google continues to increase digital mapping around the world. LIDAR is the cornerstone of autonomous driving technology: it uses sensors to measure the distance of objects and create 3D maps in real time, so the car knows when to stop, start and react. The biggest challenge is reducing noise caused from weather. Like human eyes, blinding sunlight, snow and sleet impair the vision of LIDAR equipment. Hella, Google and Siemens are all working to tackle this problem2. Another challenge is the price tag, which is about $70,000 a unit. This cost is expected to drop as volume increases.

Table 1 – Technologies for Autonomous Vehicles [4]

S.No	Technology	What it does ?	Limitations & Opportunities
1	Self Steering	Steering systems that use cameras that watch road markings and radar and laser sensors that track other objects	Machines do not yet do a good enough job of extrapolation – artificial intelligence can improve this capability
2	LIDAR	Optical remote sensing technology to measure distance to target by illuminating with light	Noise removal, including weather. Interpolation to fixed point spacing. Triangulation issues
3	GPS	Space based satellite navigation system that provides time and location information anywhere	Accuracy of a GPS receiver is ±10 meters, not practical for locating a car which is 3 meters
4	DGPS	Enhancement to GPS to improve location accuracy to 10 cm	Shadowing from buildings, foliage causes temporary losses of signal
5	Digital Maps	Process by which a collection of data is compiled and formatted into a virtual image	Only parts of the world mapped; need critical mass to enter and cross validate data in order to achieve required accuracy

CHAPTER 3: NAVIGATION OF THE SELF DRIVING BOT

Our bot is given a set of predetermined coordinates from the present location to the destination which comes from the gps module. These points are then mapped to the 2-d cartesian coordinates. So the bot has all the intermediate points it has to traverse before reaching the destination. Initially the bot is made to orient along the North direction with the help of a digital compass.

Inputs : coordinates

Output : angle to be turned and the distance to be travelled

- *Calculate the path angle and accordingly turn the bot towards left or right*

  Path angle = $\tan^{-1}\left(\frac{y2-y1}{x2-x1}\right)$

  Where, (x1,y1) = present location coordinates and (x2,y2) = target location Coordinates

```
if path angle == previous angle:
        go straight
elif path angle < previous angle:
        turn right by angle (previous angle – path angle)
elif path angle < previous angle + 180:
        turn left by angle (path angle – previous angle)
else:
        turn right by angle (previous angle+360-path angle)
```

- *Calculate the path distance*

  The path distance is the distance between present location point and the next point on the path. It is calculated using the below mentioned formula.

$$\text{path distance} = \sqrt{}((x2 - x1)^2 + (y2 - y1)^2)$$

CHAPTER 4: DESIGN

Proper selection of a motor for the equipment is key to ensuring performance of the equipment. The motors have been chosen according to the calculations as below:

4.1: Calculations of motor ratings:

4.1.1: Calculation of torque required for the bot

- Torque = Total Force × Radius

  Radius = 3.25×10^{-2} m

  Total Force = Mass × Acceleration + Frictional Force

  Frictional Force = Frictional Coefficient × N

  N = Mass × Gravitational Force

  Frictional Coefficient = 0.5

  Mass = 220×10^{-3} kg

  Acceleration = 0.375 m/s^2

  Total Force = $(220 \times 10^{-3} \times 0.375) + (0.5 \times 220 \times 10^{-3} \times 9.8)$

  = 1.1605 N

  Therefore Torque = $1.1605 \times 3.25 \times 10^{-2} = 0.037$ Nm (Newton-metre)

  The bot has 3 wheels. So the torque on each motor has to be 0.037/3 = 0.0107 Nm

  Therefore torque of each motor = 0.10968 kg-cm

4.1.2: Calculation of speed in rpm

- $$\text{Rpm} = \frac{\textbf{Speed}}{\mathbf{2\times\pi\times Radius\ of\ wheel}} \times \mathbf{60}$$

  Maximum attainable Speed = 0.7 m/s

  Radius of the wheel = 3.25 cm

  $$\text{Therefore Rpm} = \frac{0.7}{2\times\pi\times3.25\times10^{-2}} \times 60$$

  = 210 rpm

4.2: Components:

1. 2 Geared DC Motors with encoder

Based on the requirements of our project, we bought 2 geared dc motors with encoder and are chosen as per the calculations above.

A geared DC Motor has a gear assembly attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute and is termed as RPM. **The gear assembly helps in increasing the torque and reducing the speed.** Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction. This motor comes with a built-in encoder which measures the motor's speed in real time. The average output number of pulses can reach up to 7*2*50 pulses per revolution.

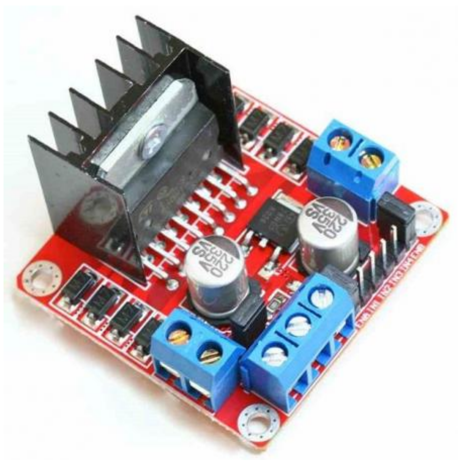
Ratings:

- Rated Voltage: 6.0 V
- Motor Speed: 15000 RPM
- Gear Reduction Ratio: 50:1
- Reducer Length: 9.0 mm
- No-Load Speed: 310 rpm@6v
- No-Load Current: 60 mA
- Rated Torque: 0.35 kg.cm
- Rated Speed: 180 rpm@6V
- Current Rating: 170 mA
- Instant Torque: 0.8 kg.cm
- Hall Feedback Resolution: 700
- Weight: 18g

2. Motor Driver – L298

The L298 Driver is a high voltage, high current dual ful bridge driver designed to accept standard TTL logic levels and drive inductive loads such relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals.

Drive Voltage: 5V-35V



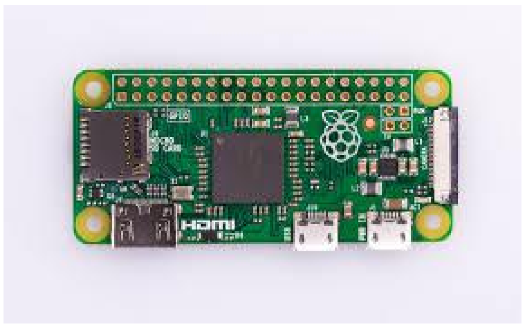
3. Raspberry Pi

Why raspberry pi over arduino?

An Arduino is a microcontroller motherboard. A microcontroller is a simple computer that can run one program at a time, over and over again. A Raspberry Pi is a general-purpose computer, usually with a Linux operating system and the ability to run multiple programs.
An Arduino board is best used for simple repetitive tasks: opening and closing a garage door, reading the outside temperature and reporting it to Twitter, driving a simple robot.
Raspberry Pi is best used when you need a full-fledged computer: driving a more complicated robot, performing multiple tasks, doing intense calculations (as for Bitcoin or encryption). This is the reason why we chose raspberry pi as our micro controller.

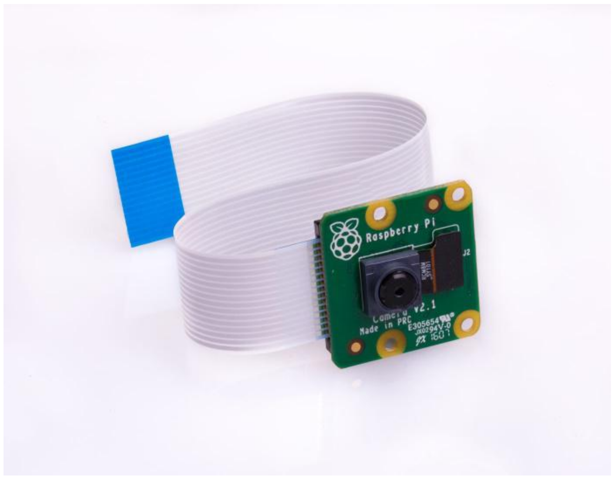
Why raspberry pi zero?

The Raspberry Pi Zero is an incredibly cheap and small computer. Its size is 65mm x 30mm. The Zero is smaller than a credit card, and it’s only 5mm thick to boot. It also weighs just nine grams.



4. Raspberry pi camera

The Raspberry Pi Camera Board v2 is a high quality 5 megapixel image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It attaches to the Pi by way of one of the small sockets on the board's upper surface and uses the dedicated CSi interface, designed especially for interfacing to cameras.



5. Battery Li- ion – 3.3V

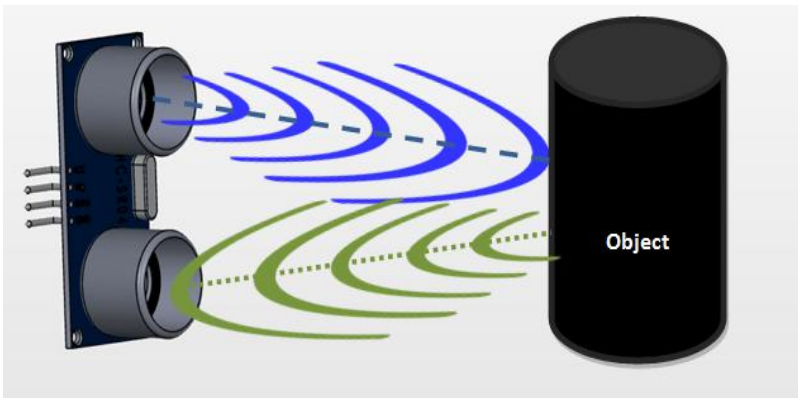
Two 3.3V Lithium-ion batteries are used to power the bot.

6. Ultra Sonic Sensor

The **HC-SR04 Ultrasonic sensor** is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple formula

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back towards the sensor. This reflected wave is observed by the Ultrasonic receiver module as shown in the picture below



Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a micro controller or microprocessor.

7. Wheels

We are using three wheels. One wheel is the ball bearing caster wheel and the other two are wheels with wheel diameter 6.5 cm.

CHAPTER 5: NVIDIA MODEL

To predict the angle, we have used convolutional neural networks (CNNs) to map the raw pixels from a front-facing pi-camera to the steering commands for a self-driving car.
The system automatically learns internal representations of the necessary processing steps such as detecting useful road features with only the human steering angle as the training signal. We never explicitly trained it to detect, for example, the outline of roads. We have used Google Colab for training the model. Before road-testing a trained CNN, we first evaluated the networks performance in simulation. For this, we used Self-driving car simulator, Open-Source by Udacity. [*Fig1,Fig2]*

5.1: Convolutional Neural Network :

CNNs have revolutionized pattern recognition. Prior to the widespread adoption of CNNs, most pattern recognition tasks were performed using an initial stage of hand-crafted feature extraction followed by a classifier. The breakthrough of CNNs is that features are learned automatically from training examples. The CNN approach is especially powerful in image recognition tasks because the convolution operation captures the 2D nature of images. Also, by using the convolution kernels to scan an entire image, relatively few parameters need to be learned compared to the total number of operations. While CNNs with learned features have been in commercial use for over twenty years [3], their adoption has exploded in the last few years because of powerful computing resources and huge datasets. *Refer to the Appendix for understanding of the CNN.*

Fig1.Training the neural network

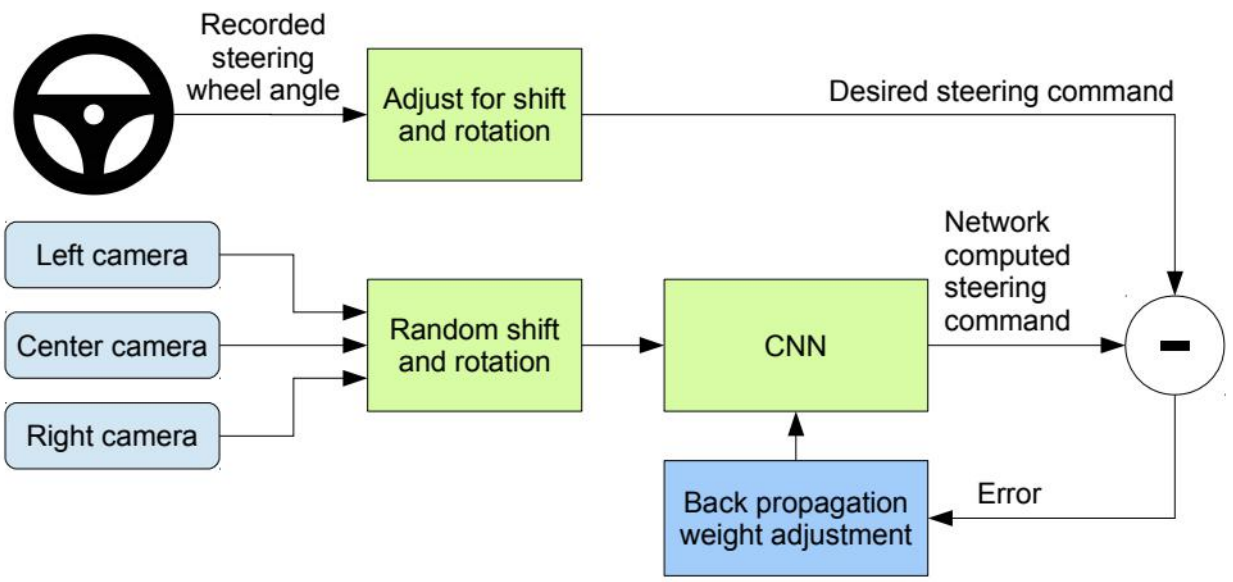
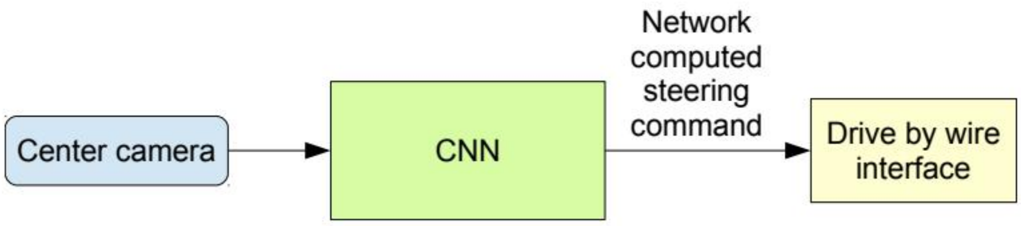

Recorded steering wheel angle
Adjust for shift and rotation
Desired steering command
Left camera
Center camera
Right camera
Random shift and rotation
CNN
Network computed steering command
-
Back propagation weight adjustment
Error

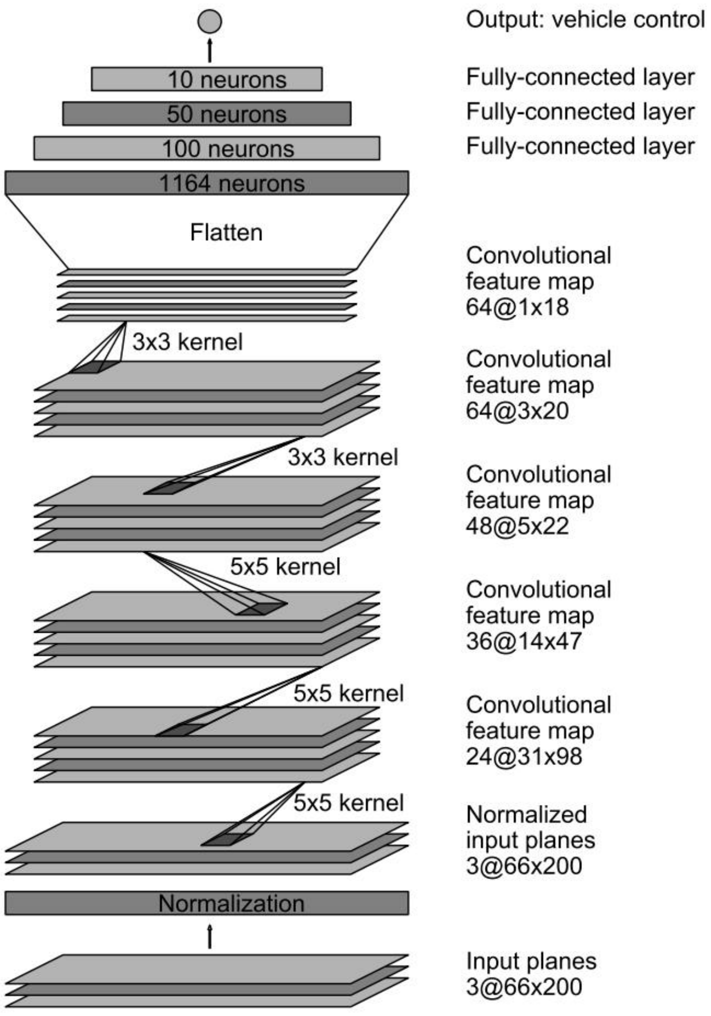
Fig2.The trained network is used to generate steering commands from a single front-facing centre camera.



5.2: Network Architecture :

We train the weights of our network to minimize the mean-squared error between the steering command output by the network, and the training data from the Self-driving simulator. The network architecture, which consists of 9 layers, a normalization layer, 5 convolutional layers, and 3 fully connected layers [*Fig3*]. The input image is split into YUV planes and passed to the network.

Fig3.CNN Architecture. The network has about 27million connections and 250 thousand parameters.



The convolutional layers were designed to perform feature extraction and were chosen empirically through a series of experiments that varied layer configurations. We use strided convolutions in the first three convolutional layers with a 2×2 stride and a 5×5 kernel and a non-strided convolution with a 3×3 kernel size in the last two convolutional layers. We follow the five convolutional layers with three fully connected layers leading to an output control value which is the angle of turning which in-turn given to the bot as speeds of inner and outer speeds. The relation between the speeds and angle of turning is described in section**.

5.3: Training Details:

5.3.1: Data Selection:

The first step to training a neural network is selecting the frames to use. Our collected data is labeled with angle of turning, throttle and speed. Images are captured by left, right and center aligned cameras of the car. Preprocessing techniques used are converting RGB format of the frame into YUV as recommended by Nvidia model architects, gaussian blur to smoothing the image and to reduce noise within the image, normalizing intensity values and resizing of the image to 200 by 66 which is the model's inputs size.

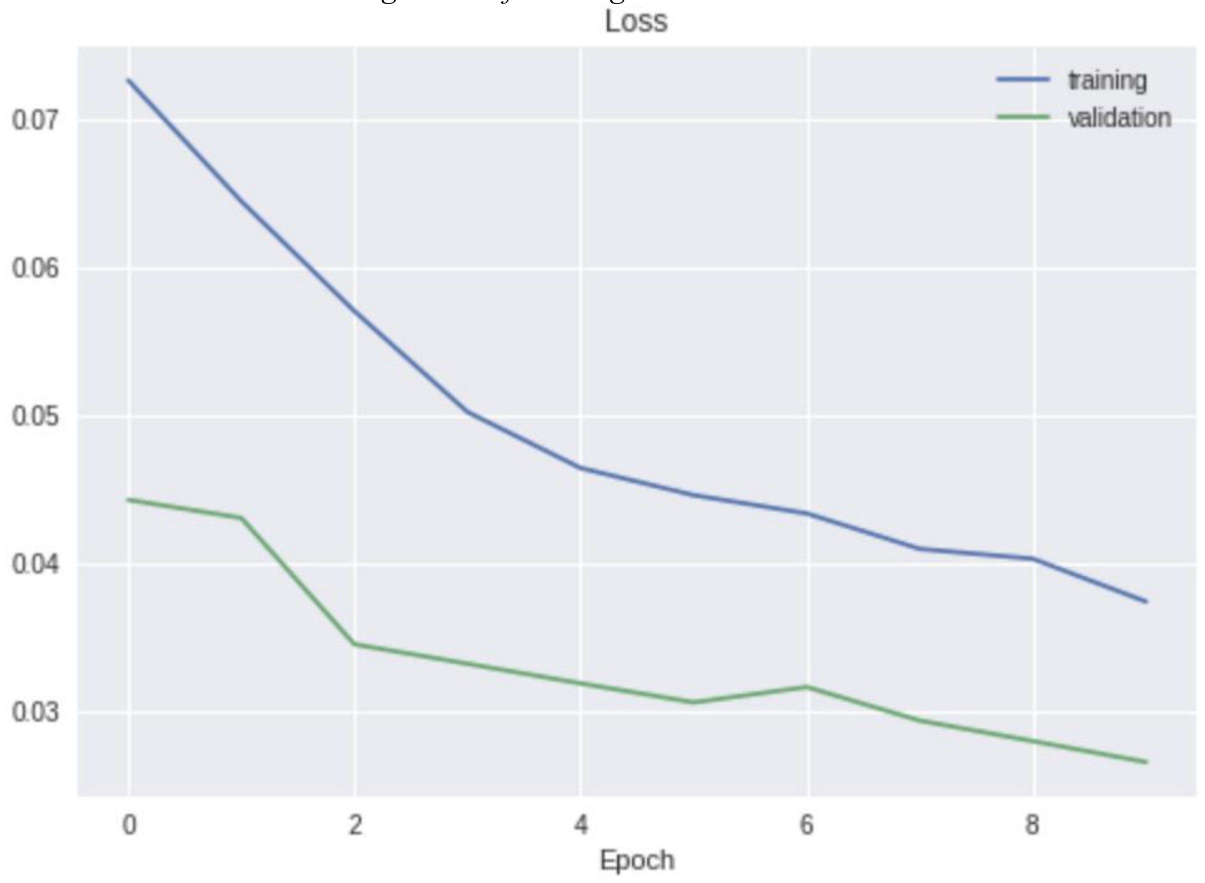
5.3.2: Data Augmentation techniques:

After selecting the final set of frames we augment the data by zooming, panning, random brightness, and flipping using cv2 and imgaug. We also used batch generator which is a memory efficient method to create small batches of images at the time only when the generator is called.

RESULTS

We have added certain dropout layers and used ELU (unlike suggested by NVIDIA) as the activation function. These changes have reduced the loss and improved the accuracy[*fig4*]. Dropout is a simple way to prevent Neural Networks from Overfitting. **Dropout** is a technique where randomly selected neurons are ignored during training. They are "dropped-out" randomly. we used ELU as it becomes smooth slowly until its output equal to -α whereas RELU sharply smoothes. ELU is a strong alternative to ReLU. Unlike to ReLU, ELU can produce negative outputs.

Fig4: loss of training and validation set.



Given below is an example how we captured a frame, pre-processed it and then used our CNN model to predict the angle of turning. This angle is used to find the speeds of the inner and outer wheels of the bot. The angle is predicted to be –27 degrees. The negative sign implies that the bot has to turn left.

Fig5. Frame captured by the camera

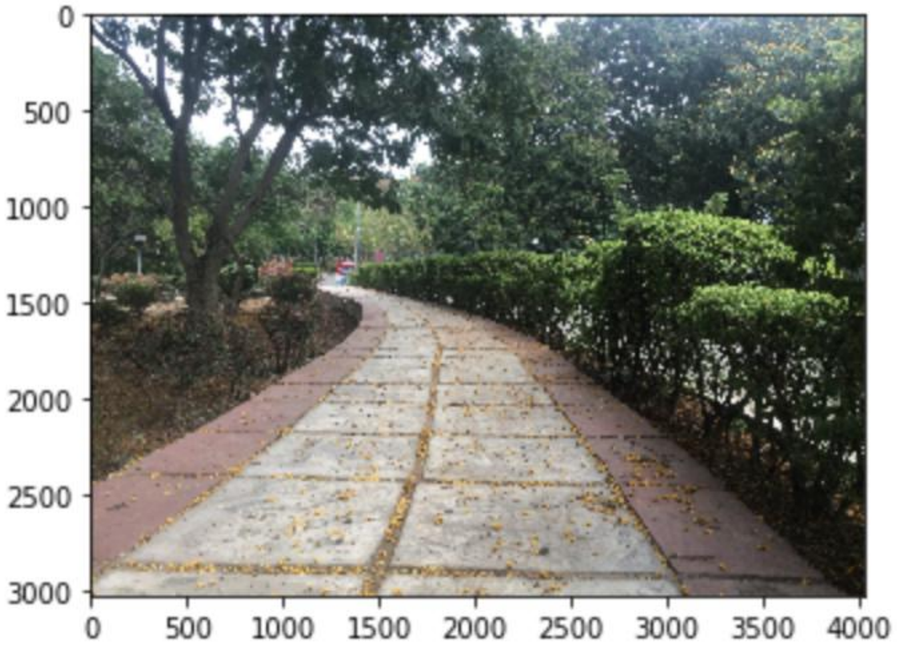
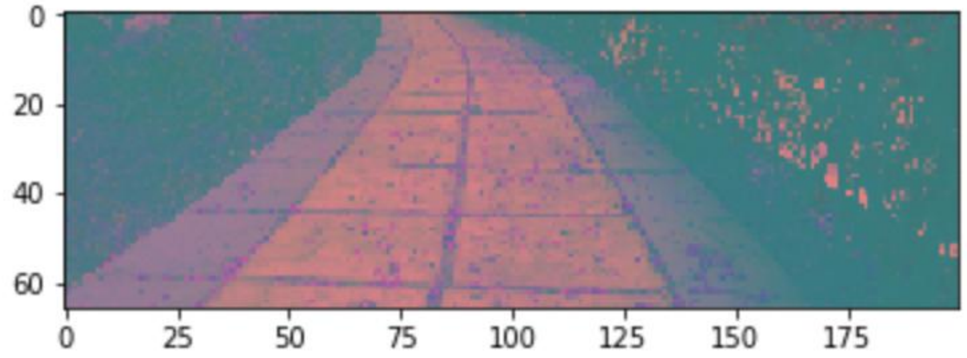


Fig6. Pre-processed image that is given to the CNN



CONCLUSION

In this report, a method to make a self-driving robot car is presented. The algorithm for the navigation system of the self-driving car is described. The different hardware components and their assembly are explained. Self-driving cars, as of now, are good but not perfect and many more changes and innovations are being done to make them more feasible and safer.

We have implemented the bot by using ultrasonic sensors and fuzzy logic for the turning of the bot. We plan to implement the same using a camera and Neural Networks as they can be used for many more purposes aside from turning the bot along the road such as object detection and identification like understanding the traffic signals and signs. Once the car starts traveling on the roads, it determines the obstacles on the way and notes their characteristic features.

REFERENCES

1 - https://www.raspberrypi.org/

2 - https://www.python.org/

3 - https://www2.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-102.pdf

4 - https://ikhlaqsidhu.files.wordpress.com/2013/06/self_driving_cars.pdf

5 - https://en.wikipedia.org/wiki/Fuzzy_logic

6 - http://journals.sagepub.com/doi/full/10.5772/54427

7 - https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

8 - https://devblogs.nvidia.com/deep-learning-self-driving-cars/

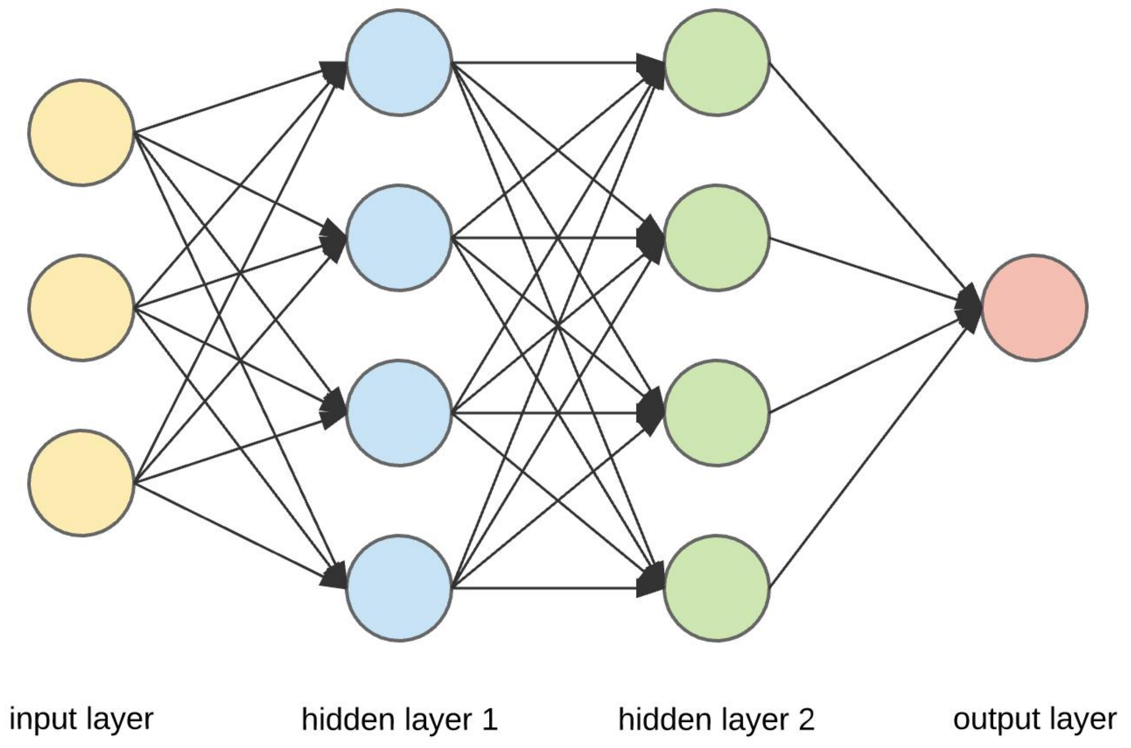
9 - https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8

APPENDIX

Neural network

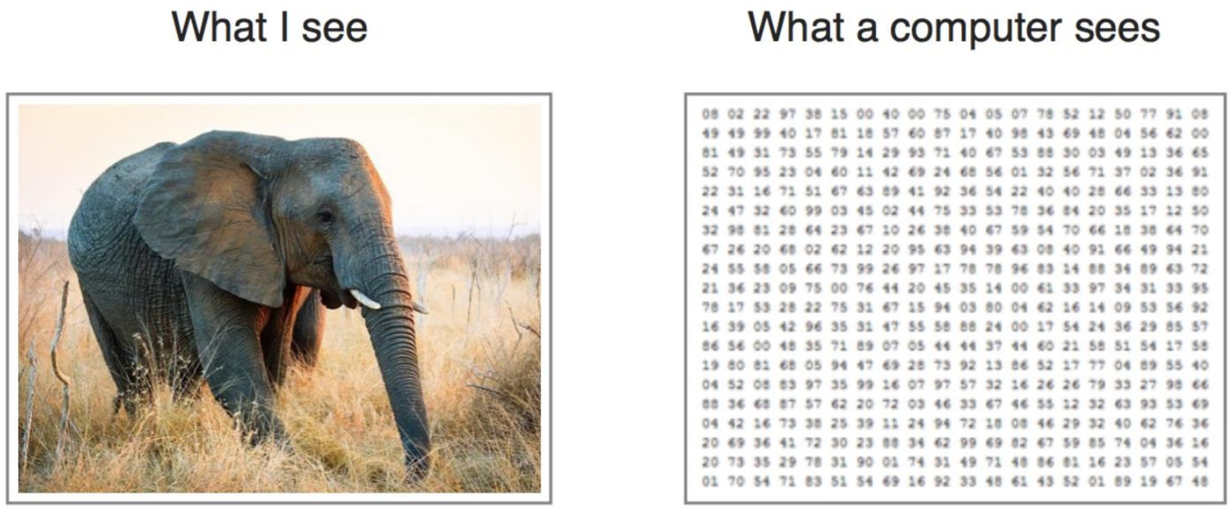
A machine learning algorithm is built on the principle of the organization and functioning of biological neural networks. This concept arose in an attempt to simulate the processes occurring in the brain by Warren McCulloch and Walter Pitts in 1943.
Neural networks consist of individual units called neurons. Neurons are in a series of groups—layers [Fig7]. Neurons in each layer are connected to neurons of the next layer. Data comes from the input layer to the output layer along these compounds. Each individual node performs a simple mathematical calculation. It then transmits its data to all the nodes it is connected to.

Fig7. Basic Neural Network



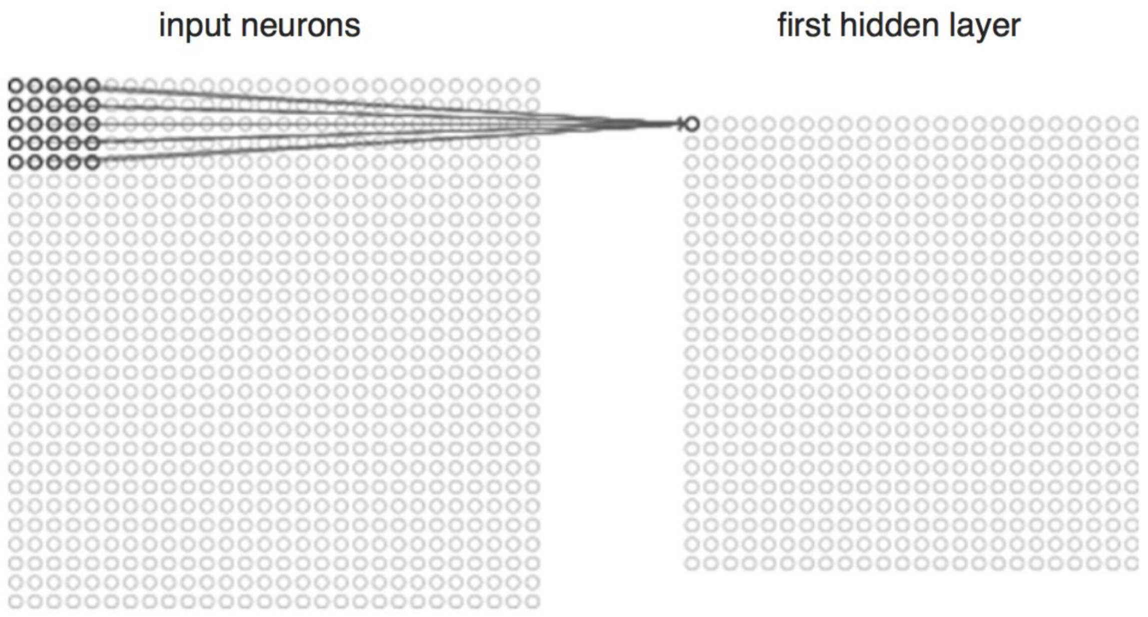
The computer sees an image different from human (Fig8).

Fig8. How a computer sees an Image.



Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.
To solve this problem the computer looks for the characteristics of the base level. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts.
In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.
The Convolution layer is always the first. The image (matrix with pixel values) is entered into it. Imagine that the reading of the input matrix begins at the top left of image. Next the software selects a smaller matrix there, which is called a filter (or neuron, or core). Then the filter produces convolution, i.e. moves along the input image. The filter's task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by 1 unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller than input matrix.

Fig9. After one convolution



This operation, from a human perspective, is analogous to identifying boundaries and simple colours on the image. But in order to recognize the properties of a higher level such as the trunk or large ears the whole network is needed.
The network will consist of several convolutional networks mixed with nonlinear and pooling layers. When the image passes through one convolution layer, the output of the first layer becomes the input for the second layer. And this happens with every further convolutional layer.
The nonlinear layer is added after each convolution operation. It has an activation function, which brings nonlinear property. Without this property a network would not be sufficiently intense and will not be able to model the response variable (as a class label).
The pooling layer follows the nonlinear layer. It works with width and height of the image and performs a down-sampling operation on them. As a result, the image volume is reduced. This means that if some features (as for example boundaries) have already been identified in the previous convolution operation, then a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.
After completion of series of convolutional, nonlinear and pooling layers, it is necessary to attach a fully connected layer. This layer takes the output information from convolutional networks. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the number of classes from which the model selects the desired class.