SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY

Faculty of Engineering

AI BASED SELF DRIVING CAR

By

Kavinda Buddhika Bandara

A thesis submitted in partial fulfillment of the requirements for the final year project for the
Degree of Bachelor of the Science of Engineering
Specialized in
Mechanical Engineering (Mechatronics)

December 2019

DECLARATION

Student ID : EN16528024

Project ID : METP19

Individual Contribution: Design, assembly, simulation, testing in path and make AI knowledge of self-driving car. Used Raspberry pi and NVidia jetson Nano based python programming, Anaconda navigator.

Supervisor: Dr. Migara H. Liyanage

Senior Lecturer – Department of Mechanical Engineering

S.L.I.I.T.

Signature: Date:

Co-Supervisor: Mr. Thakshila Dasun

Instructor – Department of Mechanical Engineering of S.L.I.I.T

Signature: Date:

Group Member(s) : 1. W.A.K.K.B.Bandara

EN16528024

Date:

Signature:

2. Ahamed M.J.F

EN16539914

Date:

Signature:

3. David .W.J EN16516762

Signature: Date:

ACKNOWLEDGEMENTS

The Final Year Project is a compulsory and important part of engineering degrees offered by Sri Lanka Institute of Information Technology (SLIIT). With the inclusion of a project with such priority, it has provided me with the opportunity to present my skills as an engineer.

Therefore, it is my duty to thank the SLIIT for providing me with this valuable opportunity.

It was a tremendous privilege for me to be supervised by Dr. Migara Liyanage, Senior Lecturer, Department of Mechanical Engineering of S.L.I.I.T. The guidance provided by Dr. Migara was instrumental in conceptualizing, developing and implementing the project to fruition.

Furthermore, it was a pleasure to have Mr. Thakshila Dasun as Co-Supervisor for the project. Their guidance was essential in a smooth progression and a successful completion of the project.

It is also noteworthy of mentioning all those who assisted in fabricating and providing parts and equipment for this research project.

This project would have never been possible without the effort and commitment of my colleagues and team members Ahamed M.J.F and David .W.J. And at last but not least I wish to thank my parents who gave their fullest support in helping complete the project.

ABSTRACT

Autonomous cars will impact our life. So people expect vehicles to be able to drive themselves place to place at any time in safely and comfortable.in this future, machine performing intelligent engagements without human supervision is one of long time for AI and is hurriedly becoming closer to reality. Tesla, Toyota, Ford, Volvo, Uber, NVidia and others have been working very heavily to reach ultimate goal of autonomous vehicle.

Autonomous cars include AI knowledge to perform its requires task. However the standard AI cars use advance technology (powerful cameras, sensors and micro controller) but regarding this project the car also present certain limitations in the constrained environment. Therefore, purpose of this research is to set a trend in the development of AI base self-driving car as well as the utilization of flexible AI base car for self-driving purposes.

In this project are design and develop a self-driving car using deep learning (neural network), which has the ability to identify lanes and drive without any human involvement. Both training and testing of the neural network should be done in a specifically designed platform. Additionally, this car should have the capability to identify the other vehicles and obstacles in the road, and drive according to the sign boards, speed limits and traffic lights. This thesis consists of an introduction regarding the field of study, methodology adopted to develop the robot and the results obtained through this research.

CONTENTS

DECLARATION i

ACKNOWLEDGEMENTS ii

ABSTRACT iii

CHAPTER 1 1

INTRODUCTION 1

1.1. Background 1

1.2. Statement of the Problems 3

1.3. Objectives of the Research 3

1.4. Scope 5

1.5. Organization of the Thesis 6

CHAPTER 2 7

LITERATURE REVIEW 7

CHAPTER 3 14

METHODOLOGY 14

3.1. Concepts 15

3.1.1. Artificial Intelligence 15

3.1.2. Machine Learning 15

3.1.3. Neural Network 17

3.1.4. Deep Learning 21

3.1.5. Convolutional Neural Networks 23

3.1.6. Image Processing 39

3.2 Methods 43

3.2.1. Literature review and data collection 43

3.2.2. Design and analysis 43

3.2.3. Manufacture and assembly 49

3.2.4. Software development 50

3.2.5. Deep Learning for Steering and Driving 50

3.2.6. Stereo Vision for distance measuring 55
3.2.7. Deep Learning for vehicle detection and passenger detection.....57
3.2.8. Deep Learning for traffic light detection and traffic sign detection 60
3.2.9. Mobile app and void command.....62
3.2.10. Programming.....64
3.2.11. Software Tools 65
3.2.12. Testing.....65
3.3. Contributions.....66
CHAPTER 4 68
RESULTS AND DISCUSSION 68
4.1. Result 68
4.1.1 Result of DNN 68
4.1.2 Udacity simulator result.....69
4.1.3 AI car steering results 73
4.1.4. Vehicle and distance detection using stereo vision.....74
4.1.5 Pedestrian detection 77
4.1.6 Sign board detection 77
4.1.7. Mobile app and voice command.....79
4.2 Discussion 81
CHAPTER 05 83
CONCLUSION AND RECOMMENDATION.....83
5.1 Conclusion 83
5.2 Recommendation 84
REFERENCE.....86
APPENDIX A i
APPENDIX B iii

LIST OF FIGURES

Figure 1: Biological Neural Network vs. Conceptual Neural Network 18
Figure 2: Neural Network Architecture 19
Figure 3: Very of output with threshold 19
Figure 4: Count the threshold and weighted 20
Figure 5: Perceptron (Left) & Mathematical Representation (Right) 20
Figure 6: Small change in any weight and change of output 21
Figure 7: Deep neural network 22
Figure 8: Convolutional Neural Networks architecture and output of the network 23
Figure 9: Linear function 24
Figure 10: Sigmoid or Logistic Activation Function 25
Figure 11: Tanh or hyperbolic tangent Activation Functio 25
Figure 12: ReLU (Rectified Linear Unit) 26
Figure 13: Leaky ReLU 26
Figure 14: Gradient descent 28
Figure 15: Gradient descent method 28
Figure 16: Convolutional Neural Networks with layers 29
Figure 17: Multiple renditions of X and O 30
Figure 18: Pixel of image 30
Figure 19: Search and compare each pixel 30
Figure 20: Match the entire image 31
Figure 21: Convolution of an Image/ step 1 31
Figure 22: Convolution of An Image/ step 2 32
Figure 23: Convolution of An Image/ step 2 32
Figure 24: Sobel Gradients process 33
Figure 25: ReLU Layer process/ step 1 34
Figure 26: ReLU Layer process/ step 2 34
Figure 27: Pooling Layer process/ step 1 35
Figure 28: Pooling Layer process/ step 2 36
Figure 29: Stacking Up the Layers 36
Figure 30: Stacking Up the Layers additional 36
Figure 31: Dense Layer- Last Layer get output 37
Figure 32: Decide final value 38
Figure 33: Canny edge detection 40

Figure 34: Hough Line Detection method 41
Figure 35: Structure of the Methodology 43
Figure 36: Raspberry pi night vision camera 46
Figure 37: Final design of AI car 48
Figure 38: Image of the path 49
Figure 39: Level view of the data collection system 51
Figure 40: Process of the autonomous self-driving car 52
Figure 41: NVIDIA Neural Network 54
Figure 42: Track 1 and Track 2 55
Figure 43: Stereo Vision for distance method 56
Figure 44: Measuring distance 57
Figure 45: Vehicle detection using classifiers 59
Figure 46: Process of Pedestrian detection 59
Figure 47: Types of sign boards and convert into gray 61
Figure 48: EdLeNet 3x3 Architecture 61
Figure 49: Circuit of the app 62
Figure 50: Blynk app dashboard 63
Figure 51: Voice command components and websites 63
Figure 52: Data set of simulator (csv file) 70
Figure 53: Test 01 CCN training (Architecture_1) 71
Figure 54: Test 01 CCN training (Architecture_1) 71
Figure 55: Test 01 CCN training (Architecture_2) 72
Figure 56: Test 01 CCN training (Architecture_2) 72
Figure 57: Real project path and images that captured by camera 73
Figure 58: Data set of AI car (csv file) 73
Figure 59: Final neural network training 74
Figure 60: Stereo vision 75
Figure 61: Measured distance verity of calculated distance 75
Figure 62: Bick detection 76
Figure 63: Vehicles detection 76
Figure 64: Pedestrian detection 77
Figure 65: Data set example of sign boards 77
Figure 66: Neural network training of sing boards 78
Figure 67: Loss of CNN 78
Figure 68: Mobile App 79

Figure 69: Assign the Google Assistance 80

LIST OF TABLES

Table 1: Performance of two boards 47
Table 2: Programming details 64
Table 3: Member's contributions.....66
Table 4: DNN level Precision 68
Table 5: Baseline Evaluation 69

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
DNN	Deep Neural Network
CNN	Convolutional Neural Network
GPS	Global Positioning System
3D	3 Dimension
RC	Car Remote Control Car
SLAM	Simultaneous Localization and Mapping
IMU	Internal Measuring Unit sensors
DL	Deep Learning
RC	Remote Control
IOT	Internet Of Things
KNN	K Nearest Neighbor
HOG	Histograms of Oriented Gradients
PD	Pedestrian Detection
ANN	Artificial neural network
PWM	Plus with Modulation
SVM	Support Vector Machine
PCA	Principle Component Analysis

ML	Machine Learning
OSL	Ordinary Least Squares
MSE	Mean Squared Error
AC	Air Conditions
GPU	Graphic processing unit
ACF	Aggregate Channel Features
PCB	Printed Circuit Board

CHAPTER 1

INTRODUCTION

Autonomous Vehicles are considered that it has the potential of saving thousands of lives per year and currently a higher number of companies are trying to develop it. Such as Tesla and Ford. This report provides the details about how the particular prototype platform for autonomous self-driving was assembled, with the details of required components, algorithm and software implementation, as well as considerations regarding the performance of the vehicle under controlled circumstances similar to highway conditions.

1.1. Background

Autonomous vehicles will revolutionize transportation. Self-driving cars will save millions of lives in situations where it is impossible for a person to prevent a car accident. The reaction times and alertness of a machine are much better. In addition, long distance cameras and ultrasonic sensors further equip these cars with super-human abilities. For such reasons, we see many corporations and researchers trying to develop technologies that would allow for a fully autonomous driving experience. Companies like Google or Udacity also try to educate the public on this topic. They provide free courses and open-source libraries as tensor flow that enable practically anyone to build machine learning models and participate in solving of the puzzle.

Autonomous technologies is modern and highly attractive research field in the engineering technology. It include many areas, where AI (Artificial intelligence) and machine learning are stay ahead. Deep learning (DL) has more complex areas, pattern matching, images recognition and complex game play also selection of example. In this report demonstrated deep learning an extremely desirable a field of knowledge, adeptness and research for several projects within several different fields of work.

In this report has started to precise interest in autonomous technology is automotive industry, wherever deep learning (DL) is applied as mean of allowing autonomy in the vehicles. The autonomous cars have several advantages, including a reduced amount of emissions in to

traffic movement, advanced operation due to the car fleets, also safety due to a smaller amount of human errors. An overview to this research regarding autonomous technologies also safety requests for vehicles that relate the technology is greatly valued advantage.

Convolutional Neural Networks (CNNs) and other deep architectures have achieved tremendous results in the field of computer vision. In most cases, they surpassed previous hand-crafted feature extraction based systems and set up a new state-of-the-art for tasks such as image classification [1], [2], [3], image captioning [4], [5], object detection [6] or semantic segmentation [7] [8]. These networks learn automatically from training data and are able to capture complex relationships that are otherwise very difficult for humans to describe by a set of hand-written rules. With the exponentially growing amount of available data and the increasing processing power of modern computers, we are able to train these networks to yield better and better results. Their impact can be seen almost anywhere. Whether it is the improved quality.

The ultimate goal of the thesis is to build a low cost prototype of an autonomous RC car through end-to-end machine learning, primarily using deep neural networks (details in Chapter 4).

This car should be able to drive itself on a flat surface that will mimic a simplified road. The main input of the car will be real time video from a camera, which will be mounted on the top. The system should then output corresponding steering commands and control the car accordingly. Since the camera will be the only input for the controller, the goal of the thesis will be to teach the car how to steer and also object detection. The car has specific mobile app and some systems working with voice command. Otherwise face recognize were tried.

Avoiding obstacles is a different problem which is also possible to solve, but combining it with steering and creating a single controller to handle both situations is beyond the scope of this thesis. However, ultrasonic sensors are used to detect obstacles on the road and stop the car accordingly. This will work as a separate module that will not interfere with the process of steering which will be managed exclusively by the neural network. The network will be trained on a separate machine and then transferred to an on-board computer that will control the car. The vehicle will then be fully independent of other machines. Another goal of the thesis is to make this system affordable and easy to build and use a mainstream single board computer

and an inexpensive car chassis. The software will be written with standard machine learning libraries and easy to extend and reproduce.

1.2. Statement of the Problems

Humans are always try to travel easily. Even through are capable in moving about in road. Our abilities are limited compared to other self-driving vehicle with respect to speed, mobility and survival, that mean vehicle is prototype. Human face great risks drive the vehicle. They can face accident because they feel sleep since drive or they can be drinking or drive careless. So roads will be safer, people will be more productive, people will can save money, move more efficiently also self-driving car could be huge benefit to the environment, therefore people should embrace self-driving car.

Autonomous vehicles, most commonly known as Tesla, Toyota, Ford, Volvo, Uber ,Google and NVidia has been developed over the years enabling humans to travel in public road. However, the use of such vehicles are confined to public or private transportation and military purposes. Furthermore, usage of self-driving car poses a threat to the involved humans as any critical failure encountered by the vehicle could cause imminent death. Therefore, it is necessary to equip ourselves with a fully or partially autonomous car which can travel the road with allowing humans to observe safely during the driving. Autonomous vehicles are the best solution to overcome the problems of traveling the road. So An Autonomous car, in this technology all the time support the people to makes life easy and safe transportation. In this report using new technique of the computer vision to finding the self-driving car way, road line, traffic signals, object detection so can able to see next generation car is completely autonomous.

1.3. Objectives of the Research

With the analysis of problem statement, it is clear that a research must be conducted on autonomous vehicle. The complexity and functionality of the autonomous car may vary depending on the task at hand. However, in general, autonomous car will include a vision based navigation system GPS system, powerful camera powerful radar so it is made a 3D map

(30-25 circle area) while the traveling. Furthermore it has IMU (internal measuring unit sensors) and special algorithm always running the AI knowledge.

As mentioned in the problem statement, this prototype car imposes limitations to the traveling the road. To counter these limitations, an innovative method of developing this car must be employed. Because it can't identify all the object in the road why the time was limited. If more images and more objects were can trained the car should have more accuracy. Furthermore this prototype vehicle have special path to train and take images etc. also hard to train with raspberry pi because it had low GPU and speed. So always have problems communication between Wi-Fi router, laptop and vehicle micro controller. Therefore raspberry only use to face recognize.

Therefore, the research project is broken down to primary and secondary objectives, where the primary objectives are essential for the success of the project and secondary objectives are optional and will provide more value if implemented.

Primary Objectives:

1. Design and Development of the self-driving car
2. Design and development of training path
3. Take the images and run the car on the path
4. Validate the overall system with experiments. .

Secondary Objectives and research:

1. To identify the basis of the Deep Neural Network related to autonomous vehicle.
2. To identify the obstacles and working of the stereo vision related to obstacle identifying and path detecting.
3. Implementation of path designing and teaching of the autonomous vehicle.
4. To define the hidden layers of the neural networks to make the output precise.

5. To learn the basics of the Artificial Intelligence and Machine Learning related to the autonomous vehicles.
6. To identify the Android Blynk App and ISO Blynk app
7. To identify the Google assistant, Adafruit and IFTTT web base service.
8. Coding with raspberry pi and NVidia jetson Nano
9. Crop images and taking string angle to Excel sheet (Open CV) and face recognize.
10. Training neural network (Anaconda navigator)/traffic sign board, object, light identification and measuring vehicle speed using stereovision technology. 11. Vehicle type identification (classifiers)
12. Minimizing errors.

1.4. Scope

The purpose of this project is to develop an autonomous car with the hardware and software. Autonomous cars, AI technology are a rising trend in the field of robotics and AI industry. However, Self-driving cars are rare in the world. Especially Autonomous car and this research haven't in Sri Lanka. Such a design will prove to extremely useful for public and private transportation applications.

With the constraints of time and budget, the scope of this project was limited to develop a selfdriving car of a general design greatly influenced by NVidia self-driving car. However, with the availability of materials and parts several other designs had to be considered in designing the self-driving car.

The main focus of this project will be the image processing, neural network training and stereovision technology. Those technology's vehicle extremely rare in the automobile industry. Therefore, this development will be an important stepping stone in the arena of autonomous car.

1.5. Organization of the Thesis

The remainder of the thesis is organized as follows.

Chapter 2 covers the literature review carried out to assess the work done by others and build up a basis on which the project can be done. So discuss about

Chapter 3 consists of the methodology followed to complete the research project. It have concept and method that use to project in to the success. It has furthermore discuss about theories and equations. (DNN, design and analysis, software development, stereo vision technology, mobile app, voice command and contribution)

Chapter 4 discusses results of the project throughout its development and testing. Also discuss about the results. In this chapter discuss about result of DNN, udacity simulator, vehicle, traffic sign board and pedestrian detection.

Chapter 5 concludes the project and provides recommendations for future work.

CHAPTER 2

LITERATURE REVIEW

The AI base self-driving car is an upcoming technology. In our project, we are taking a step towards this vision by developing a system using Raspberry Pi, Image processing and machine learning and connect the system to the RC (remote control) car.

In the literature review of AI base self-driving car the research team studied numerous published research papers, articles and documents from various sources. Research team learned that this research has got a great influence from the past experiences of other researchers who successfully launched their researches different ways, and who could not apply their solution to the real world in the way they intended. As going deeper into the research, the group could come up with distinguished ideas as listed below.

- Image Processing
- Neural network
- Controlling Vehicle (steering angle, speed of the vehicle)

According to S. Shreyas Ramachandran, A. K. Veeraraghavan, UvaisKarni and K. Sivaraman "Development of flexible Autonomous Car Systems Using Machine Learning and black chain" is a project, which has been developing system by Raspberry pi, image processing, machine learning and connect to several electric car. This paper aim to reach the above by image processing which train by neural network, security and transparency of the data transmission could be achieved. The Hardware component is used. In this project use Raspberry Pi Microcomputer and also Raspberry Pi camera module. Also it is use the CNN (convolutional neural network). "The authors would like to acknowledge that downscaled model developed for testing, explained in the newspaper above had b made in to a prototype and is accessible at IEEE SS12 Maker Fair 2018 Pilot at Jeppiaar Institute of Technology, Chennai, India and is recognized as being a noteworthy project and concept [9].

Wen-Yen Lin, Wang-Hsin Hsu and Yi-Yuan Chiang is released the research paper of "A Combination of Feedback Control and Vision-Based Deep Learning Mechanism for Guiding

Self-Driving Cars" This research paper is launched by developing behavior of human driving car in 2018. That is involved the position, velocity and surrounding environment. Self-Driving car is use deep neural network as a computational framework to position of car is related to the road. It is a real car and it had I/O ports, which can be connected to state of the art sensors, including camera, laser range sensors, depth sensors and IMUs. The researcher is used many extreme large steering angles. So he can control the vehicle regarding the images and steering angle data. This method of to set steering angles are very accurate. Researcher is thought and changed the direction to (+) or (-). It can easy to handle steering angles. The model architecture was inspired from NVIDIA research paper. It has been 10 layers [10].

According to Eshed Ohn-Bar and Mohan Manubhai Trivedi,"Looking at Humans in the Age of Self-Driving and Highly Automated Vehicles" research paper is launched by the role of humans in the next generation of driver assistance and intelligent vehicles.

The researcher is done to this research in to real vehicle not for prototype in 2016. There is three main domains where humans and highly automated or self-driving vehicles interact, Human in vehicle, Human around the vehicle. Furthermore this vehicle has measure driver distance, highly autonomous vehicles, human intent and behavior analysis, human robot interaction, intelligent vehicles, pedestrian/vehicle tracking, risk forecasting, vehicle driver hand off. A highpoint of the research pipeline, Overview of the sensing and learning pipeline commonly used to study humans in the cabin. A multi sensor driver gesture recognition system with a deep neural network [11].

Qudsia Memon,Shahzeb Ali, Wajiha Shah, Muzamil Ahmed and Azam Rafique Memon (Department of Electronic Engineering,Mehran University of Engineering & Technology, Jamshoro Sindh, Pakistan) researched by "Self-Driving and Driver Relaxing Vehicle" in this research focused to be automated to give human driver relaxed driving. Researcher has started working on the self-driving cars since 2010. A three wheeled mobile robot is used for this research. It has multiple sensors, an Adriano communicate with Google maps API (Application program interface). Also in this project Researcher use GPRS module to recognize the path. It is connected to directly and get route and moves in the direction. Google launched and tested some different features where each prototypes speed is capped at a 25mph, and during this stage safety drivers onboard with a changeable steering wheel, accelerator pedal, and brake pedal that allow them to take over driving if needed. That was revolution on selfdriving car [12].

Rejwan Bin Sulaiman (Institute of Arts, Science and Technology, Department of Computing Glyndŵr University, Wrexham) have done research about "Artificial intelligence Based Autonomous Car". It is included mainly Ethical issues with driverless cars and Debating issues. It is directly affect in to vehicle control. AI base car technology is deepened on the sensors cameras motors and microcontroller.so it has more rick. PS and other Ai base car will make it independent to operate just with the command by the users of the destination. There is huge probability to face accident by the other conventional car. So safety of passengers is important. Then controlling part in the vehicle was must accurate. Cording to this research papers, In 2020 Autonomous car will be launched, it will not proto type [13].

"Using Artificial Intelligence to create a low cost self-driving car" This research papers mostly according about real self-driving car. It is a solution of accident as well as 2.5 million deaths and 50 million injured. A human driver as able to recognize only traffic signs, traffic lanes and obstacles. The latest statistics are converting vehicle accidents reveal the 87% of traffic crashes. So they is created the self-driving car. So this car has 3D data from the lieder, 3 cameras (front, left, right) and basically it is taken in road traffic details. That data are controlled by four laptops and those are controlled by one main laptop. So it is behave like microcontroller. This has different types of neural networks. There are two neural networks for traffic signs forms (blue forms and red forms).For each traffic sign form there is a neural network associated with. The networks are learned in software. So this research and project are successfully [14].

Dean A. Pomerleau (computer science Department, Carnegie Mellon University) has been released the research paper of "ALVINN: AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK" ALVINN (Autonomous Land Vehicle in a neural network) had 3 layer back-propagation network design for the task of road following. ALVINN take images from the camera and laser range finder. It is produced as output direction of the vehicle should travel in order to following the road. Training has been conducted using simulated road images.

ALVINN's current architecture consists of a hidden layers it is single hidden layer backpropagation network. There are most layers in the neural network and finally, the output layer consists of 46 units and there is divide in to two group. First 45 units is a linear representation middle unit represents the " travel straight ahead" condition and other units to the left and right of the center represent successively shaper left and right turns. There are more units in the NLVINN. So more accurate has this project [15].

[16] Shows, A low cost Micro-controller implementation of Neural Network base hurdle avoidance controller for a car-like robot. This research is carried out by 6 members. Here, they mentioned that, the Conventional controller is not suitable since, noisy environment and unpredictable results occurred. Artificial neural network (ANN) is used here to implement the project, which is more powerful. But, the computational cost requires to employ the ANN method is a disadvantage. To develop the controller it is based on multilayer feed-forward neural network with back propagation learning or modified versions of this algorithm. Also after the training. PWM signal has to be used to control the speed of the motors while the car travels in the curves. In order to reach a specify destination, the robot has to be connected with GPS. As they mention here, ANN maybe a suitable method as the neural network part for the AI based self-driving car.

Study [17] emphasizes an experiment on Real-time Image processing in the Controlling of Mechanism wheel Robotic Car, which was carried out by two researchers. Image Processing is their Choice, which can be coded in java, c++ , python and etc. Here Python is used. Open CV is a Machine Learning Image Processing Tool is used to recognize the Patterns. K-mean clustering method waws used to locate the center point of the pattern. They have designed their model using four mecanum wheels, which can be driven omni directional. The motor drivers are driven by Arduino Mega. Radio Frequency module, nrf24101, is used for wireless communication with the computer. The computer is used as the transmitter with an Arduino Uno, where the messages sent from the computer and are converted into movements. The transceiver module has a SPI connection to the microcontroller of the Robot. Which has enough transaction speed of 2Mbps and the payload length is 1-32 bytes within 100m. The car's position is controlled by the pattern recognition of image processing. The pattern recognition is happened when the key positions of the car has found. An algorithm called Features from Accelerated Segment Test (Fast) is used to find the key points of the location. Then after, Brute-Force Matcher is used for machining the key images and query image. The right specific center point is required, due to pattern position is used for the controlling. The principle of the K-mean clustering method is finding the mean center point from a cluster of the points. The target position and the Robot position are detected by the Open CV. Since, the Car is driven by the Arduino microcontroller, the A special python library called pyserial is used which, can access communication (serial) ports from the computer. In this way, the movement commands or the stop command from the image processing python program could be sent to the microcontroller with a communication module. The program starts, when the camera placed above the driving path turns on, captures the target position then turns off.

Likewise, it continuously detects the car's position. Real time Image processing is a complex thing to implement in a robot.

Study [18] illustrates Real time road edges detection and road signs recognition, which was carried out by two researchers. Here, they use image processing method with Visual Studio 2013 environment with Open CV 2.4.91. They have collected the data with a camera and then they are implemented for the results. For smooth processing the color images have to be converted into gray images. The interferences are reduced using the Mean shift Gaussian filter is used. They have used the "Canny edge Detection" to detect the edges. First the noises were filtered in the image then, the edge strength was found by taking the gradient of the image. The last step to find the edge direction is trivial. Hough transform method is used for extracting elements from an image. This project was done using SURF method which is using in the field of road signs recognition.

Another related work related to the project is toward fully Autonomous Driving with Systems and algorithms. The car was designed with real basis model and an interface box was attached for the controlling of the functions of the car and as well as brake pressure and turn signals. In advantage the car was able to switch to manual function when a malfunction is occurred in the system interface or to power failure. A 64-beam LIDAR and four cameras which is a module of Ladybug 3 spherical camera and two Flea's for forward stereo and grasshopper for high resolution. For the navigation a GPS module is attached to it. A laser calibration was done for recovering optimal parameters for each beam's orientation and distance response function. A PID control was attached to help the Model Predictive Control Strategy. The designed for obstacle finding, line following, path detecting, Traffic light and etc. Comparing to other known methods this one is advanced due to the proper calculations have been obtained and the recognition of images and obstacles will be accurate due to algorithm created for identification and working [19].

A research article-based Development of Small – Scale Research Platform for Intelligent Transportation System which was researched by the senior members of IEEE. The hardware contains an arena, indoor localization system, RC Cars and road side monitoring facilities. The indoor localization system is used to mimic the GPS in real life and it is designed based on optical motion capture system. It works in high accuracy. Here a mic is attached in the back to mimic the collision sound. Tracking control algorithm allows the RC car to obtain predefined trajectories and the signals are sent using Xbee wireless communication. The camera was attached in the way of 360 degrees which could be able to cover the whole area. The Xbee

wireless data rate up to 250 kb/s and can serve a communication channel between the car and the infrastructures beside the road. The embedded board which is used here is ATmega 162. The advantage is that the experiment results with tracking algorithm is helpful for identifying the pros and cons. It is required that the results which are obtained should be checked with the on-process module and what is the advantage of relying on the 360degree camera. It is required that a study should be done on the traffic identification algorithm for more concerns [20].

The next research was based on Computer – Vision – Based Enhanced Vehicle Safety which was carried out by the Trivedi, Gandhi and Joel McCall. The system was categorized based on the Environment infrastructure, Vehicle which required every telematics devices and information gadgetry and driver which is important for the human-vehicle system. The car is designed according to the basis and safety measures if there is a malfunction which might affect the passengers. This is a camera-based system which is designed to support real-time control of vehicles. Also, it requires to identify obstacles while moving on the road. The multimodal signal processing, pattern recognition and decision and control theories should be examined before driving. Xeon processor is used as the embedded system. MATLAB is used for the implementation of face-orientation estimation. It also has the advantage of detecting heads in a vehicle. The implementation of looking in and looking out should be further researched for the purpose of the advancement of on-going project [21].

Based on the survey of Technical Trends of ADAS and Autonomous Driving which was published by Ryosuke, Yuki and Kazuaki, Renesas Electronic Corporation, Japan. The papers provide us the issues regarding the technology towards autonomous driving based on the functionality of the driving. For the reduction of traffic accident 6D-vision is being used. It has two generations, first is to calculate movement and depth information from the sequence of images obtained by the stereo images using stereo matching and optical flow technique. Second is based on the stereo vision and tracking algorithms. The tracking algorithm is a combination of matching methods and cost functions. This was implemented in FPGA platform. The input image will be in a scale of 750 x 480 in a gray image with 12bit/px, but the computation is limited to 680 x 400 with disparity range from 0 through 127. Optical flow algorithm is another main scope of 6D-vision. NVDIA graphics adapter CUDA capability, and achieving 25Hz at 640 x 480px input images [22].

“Experiment on Real-time Image processing in the Controlling of Mecanum wheel Robotic Car” is a research which was carried out by two researchers. Image Processing is their Choice, which can be coded in java, C++ , python and etc. Here Python is used. Open CV is a Machine

Learning Image Processing Tool is used to recognize the Patterns. K-mean clustering method waws used to locate the center point of the pattern. They have designed their model using four meconium wheels, which can be driven Omni directional. The motor drivers are driven by Arduino Mega. Radio Frequency module, nrf24101, is used for wireless communication with the computer. The computer is used as the transmitter with an Arduino Uno, where the messages sent from the computer and are converted into movements. The transceiver module has a SPI connection to the microcontroller of the Robot. Which has enough transaction speed of 2Mbps and the payload length is 1-32 bytes within 100m. Positioning of the car is done by image processing. The pattern recognition is happened when the key positions of the car has found. An algorithm called Features from Accelerated Segment Test (Fast) is used to find the key points of the location. Then after, Brute-Force Matcher is used for machining the key images and query image. The right specific center point is required, due to pattern position is used for the controlling. The principle of the Kmean clustering method is discovery the mean center point from a cluster of the points. The target position and the Robot position are noticed by the Open CV [32].

CHAPTER 3

METHODOLOGY

To research on the possible solutions for building a prototype which would be able to fulfill the requirements methods such as scientific literature, popular science articles and online forums were reviewed. The primary focus was to find similar projects related to image processing, computer vision, self-driving algorithm, obstacle identifying algorithm and neural network algorithm are some of them.

The goal of this project is to develop a physical model to prove that the objectives of the thesis can be achieved. Therefore, it is necessary to perform a series of a theoretical and practical analysis to obtain successful results.

It is possible to perform preliminary calculations and modelling to estimate physical variables and to decide on design requirements. Based on these theoretical results, it is possible to fabricate the physical model depending on the design requirements.

The fabricated physical model must be subjected to a series of tests to prove that it can handle the purposes it's built for. These tests will measure durability, functionality and efficiency of the design.

Several sketches were made for the planning of the component layout and assembly of the prototype. The stage was to culminate with creating a 3D-model to visualize the placement of the components and to figure out the total size of the demonstrator. The required components were obtained from shops and some of the parts were 3D printed.
The protype was assembled and underwent some minor iterative changes to improvise the fixing of the components. Banners and paints were used to implement the required environment for training. The design experiment design, result, discussion and conclusion were presented at the end of the report.

3.1. Concepts

The report contains the concepts behind Machine Learning Fundamentals, Theory of Neural Network, Autonomous Car Principles and Image Processing, Android app and Google Assistance.

3.1.1. Artificial Intelligence

Artificial intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans. Some of the activities computers with artificial intelligence are designed for include:

- Speech recognition
- Learning
- Planning
- Problem solving

3.1.2. Machine Learning

Autonomous cars are very closely associated with IOT. IOT combined with other technologies such as machine learning, artificial intelligence, local computing etc. are providing the essential technologies for autonomous cars. Very inquisitive questions for many is how are these autonomous cars functioning. What, actually is working inside to make them work without drivers taking control of the wheel.

One of the main tasks of any machine learning algorithm in the self-driving car is a continuous rendering of the surrounding environment and the prediction of possible changes to those surroundings. These tasks are mainly divided into four sub-tasks.

- Object detection
- Object Identification or recognition Object classification
- Object localization
- Prediction and vehicle of movement.

Machine learning algorithms can be loosely divided into four categories: regression algorithms, pattern recognition, cluster algorithms and decision matrix algorithms.

3.1.2.1 Regression Algorithms

In Autonomous vehicle, images (camera) play a very important role in localization and actuation, while the biggest challenge for any algorithm is to develop an image-based model for prediction and feature selection. The type of regression algorithms that can be used for self-driving cars are a Bayesian regression, neural network regression, and decision forest regression, among others.

3.1.2.2 Pattern Recognition Algorithms (Classification)

In Autonomous vehicle, the images obtained possess all types of environmental data,filtering of the images is required to recognize instances of an object category by ruling out the irrelevant data points. Pattern recognition algorithms are good at ruling out unusual data points. Recognition of patterns in a data set is an important step before classifying the objects. These types of algorithms can also be defined as data reduction algorithms.
These algorithms help in reducing the data set by detecting object edges and fitting line segments (polylines) and circular arcs to the edges. Line segments are aligned to edges up to a corner, then a new line segment is started. Circular arcs are fit to sequences of line segments that approximate an arc. The image features (line segments and circular arcs) are combined in various ways to form the features that are used for recognizing an object.

The support vector machines (SVM) with histograms of oriented gradients (HOG) and principal component analysis (PCA) are the most common recognition algorithms used in AI car. The Bayes decision rule and K nearest neighbor (KNN) are also used.

3.1.2.3 Clustering

Sometimes the images obtained by the system are not clear and it is difficult to detect and locate objects. It is also possible that the classification algorithms may miss the object and fail to classify and report it to the system. The reason could be low-resolution images, very few

data points or discontinuous data. This type of algorithm is good at discovering structure from data points. Like regression, it describes the class of problem and the class of methods. Clustering methods are typically organized by modeling approaches such as centroid--based and hierarchical. All methods are concerned with using the inherent structures in the data to best organize the data into groups of maximum commonality. The most commonly used type of algorithm is K-means, Multi-class Neural

3.1.2.4 Decision Matrix Algorithms

This type of algorithm is good at systematically identifying, analyzing, and rating the performance of relationships between sets of values and information. These algorithms are mainly used for decision making. Whether a car needs to take a left turn or it needs to brake depends on the level of confidence the algorithms have on the classification, recognition, and prediction of the next movement of objects. These algorithms are models composed of multiple decision models independently trained and whose predictions are combined in some way to make the overall prediction while reducing the possibility of errors in decision making. The most commonly used algorithms are gradient boosting (GDM) and AdaBoosting.

3.1.3. Neural Network

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input.so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems. So Neural networks are modeled loosely after the human brain that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all realworld data, be it images, sound, text or time series, must be translated.

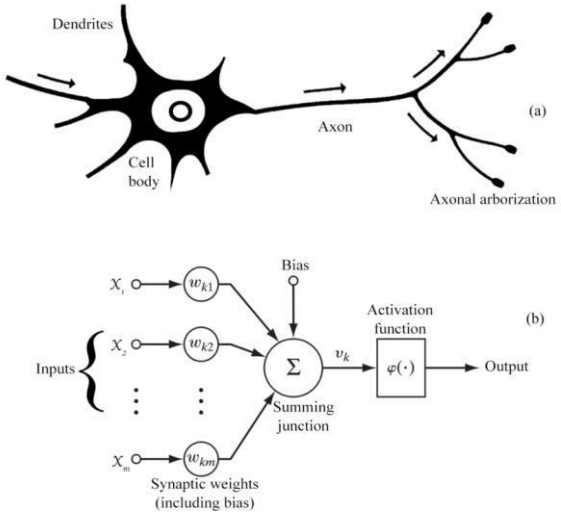


Figure 1: Biological Neural Network vs. Conceptual Neural Network

3.1.3.1 Basics of Neural Networks

Neural networks, in the world of finance, assist in the development of such process as time-series forecasting, algorithmic trading, securities classification, credit risk modeling and constructing proprietary indicators and price derivatives. A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis. A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

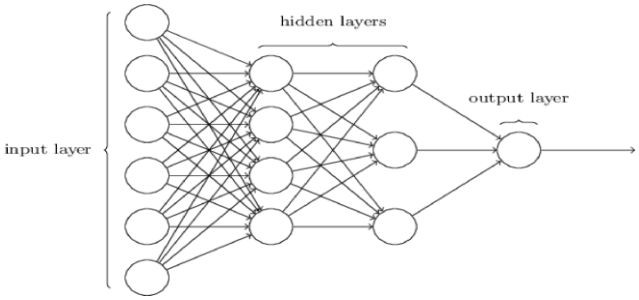


Figure 2: Neural Network Architecture

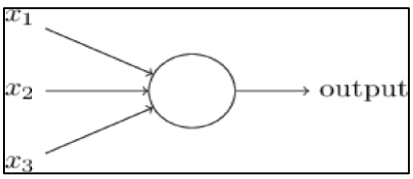
3.1.3.2 Perceptron

A perceptron can have any number of inputs, but this one has three binary inputs x1, x1, and x3, and produces a binary output, which is called its activation.

First, we assign each input a weight, loosely meaning the amount of influence the input has over the output.

A weight is known as the connection between the neurons which carries the values. When the value is high the weight will be large. Most importantly the neurons will be added to the input side of the weight. In mathematical representation and programming the weights are represented as matrix

In the picture above, weights are illustrated by black arrows, call each weight w. Each input, x above has an associated weight: x1has a weight w1, x2 a weight of w2, and x3, a weight of w3.



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Figure 3: Very of output with threshold

The neuron's output, 0 or 1, is determined by whether the weighted sum ∑j wj xj is less than or greater than some threshold value. (j=1, 2, 3…)[30]

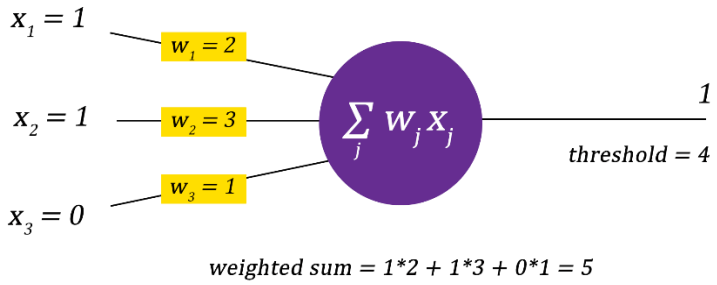


Figure 4: Count the threshold and weighted

3.1.3.3 Sigmoid Neurons

Perceptron model takes several real-valued inputs and gives a single binary output. In the perceptron model, every input xi has weight wi associated with it. The weights indicate the importance of the input in the decision-making process. [33]The model output is decided by a threshold W_o if the weighted sum of the inputs is greater than threshold W_o output will be 1 else output will be 0. In other words, the model will fire if the weighted sum is greater than the threshold.

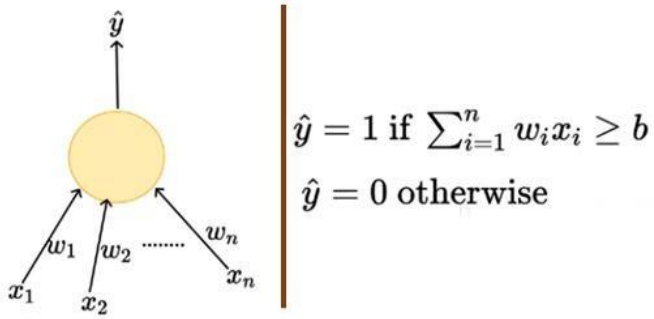


Figure 5: Perceptron (Left) & Mathematical Representation (Right)

In a neuron small change in weight should cause only a small corresponding change in the output from the network so that's why use Sigmoid Neurons Figure 6:

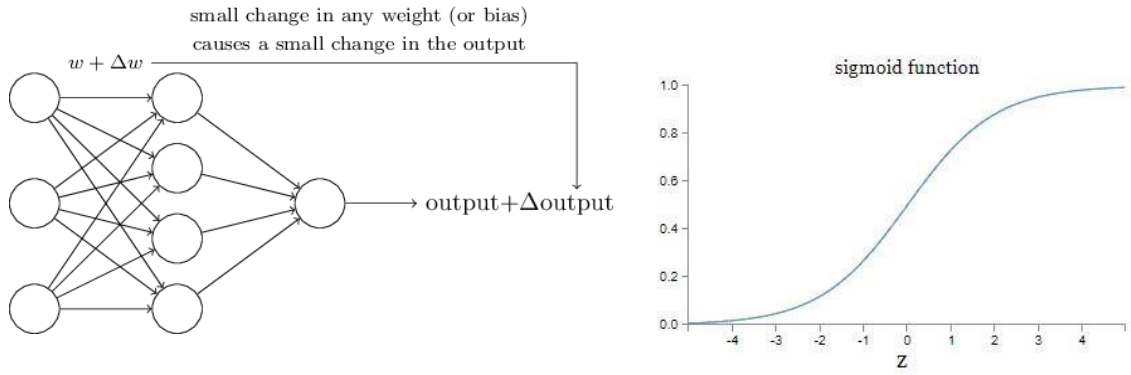


Figure 6: Small change in any weight and change of output

The smoothness of σ means that small changes Δwj in the weights and Δb in the bias will produce a small change Δoutput in the output from the neuron. In fact, calculus tells us that Δoutput is well approximated by following equation,

Equation 1: Output with change weights and bias

$$\Delta \text{output} \approx \sum_j \frac{\partial \,\text{output}}{\partial w_j} \Delta w_j + \frac{\partial \,\text{output}}{\partial b} \Delta b,$$

3.1.4. Deep Learning

Mostly DL is based on the observation of the working process of the brain while processing the information and also, it is believed that in every level of the brain, there is feature at an increased abstraction level. Main purpose of the DL in ML is to copy the behavior in the form of architecture.

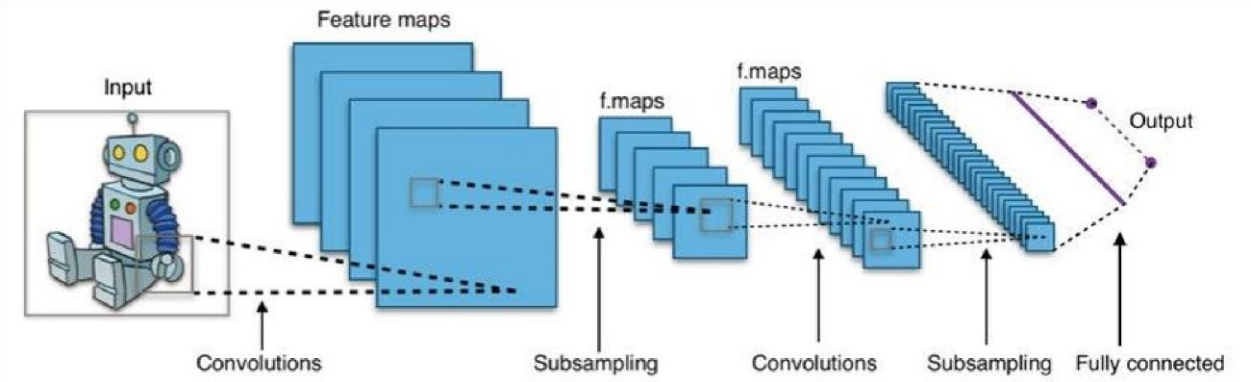


Figure 7: Deep neural network

Deep Neural Networks in the human brain and applied to the natural structures which are related to the human brain and this technology in to the modern computes. These kinds of typical applications are to use models such as complex functions approximates.

Deep Neural Networks are the sub part of the Deep learning and the structure can be divided as,

1. Multilayer perceptron
2. Deep auto-encoder

The multi-layer perceptron is considered as the fully connected feed forward network where a supervised approach is used to obtain the results. The deep auto-encoder where it is in the feed forward condition and the main purpose of it to make prediction on the input of the network where it is related to the unsupervised learning.

Neural network for an artificial intelligence is mathematical models which are inspired by the natural structures.

3.1.5. Convolutional Neural Networks

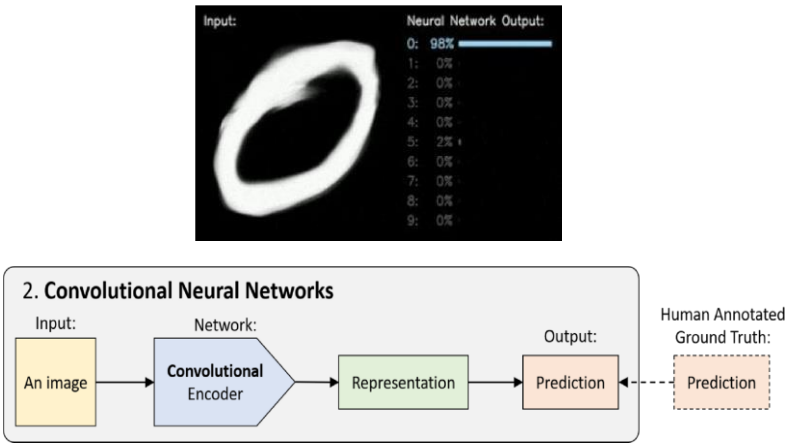


Figure 8: Convolutional Neural Networks architecture and output of the network

CNNs are feed forward neural networks that use a spatial-invariance trick to efficiently learn local patterns, most commonly, in images. Spatial-invariance means that a cat ear in the top left of the image has the same features as a cat ear in bottom right of the image. CNNs share weights across space to make the detection of cat ears and other patterns more efficient. Instead of using only densely-connected layers, they use convolutional layers (convolutional encoder). These networks are used for image classification, object detection, video action recognition, and any data that has some spatial invariance in its structure.

The Activation Functions can be basically divided into 2 types

- Linear or Identity Activation Function
- Non-linear Activation Functions

3.1.5.1 Linear or Identity Activation Function

The activation is proportional to the input. . This can be applied to various neurons and multiple neurons can be activated at the same time.

- Equation : f(x) = x

- Range : (-infinity to infinity)

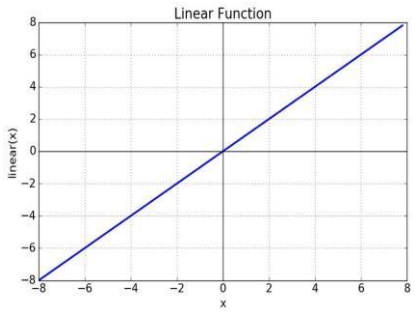


Figure 9: Linear function

3.1.5.2 Non-linear Activation Functions

The Nonlinear Activation Functions are the most used activation functions. It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the outputs.

1. Sigmoid or Logistic Activation Function
2. Tanh or hyperbolic tangent Activation Function:
3. ReLU (Rectified Linear Unit) Activation Function
4. Leaky ReLU
5. Softmax

1. Sigmoid or Logistic Activation Function

- Equation : f(x) = 1 / 1 + exp(-x)
- Range : (0 to 1)
- It gives rise to a problem of “vanishing gradients”, since the Y values tend to respond very less to changes in X It saturate and kill gradients.

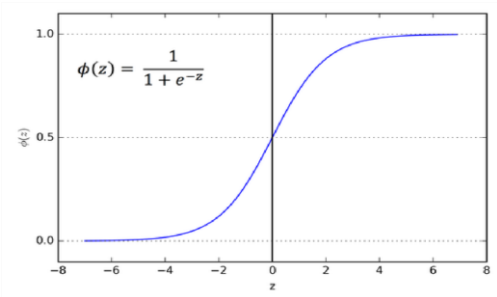


Figure 10: Sigmoid or Logistic Activation Function

2. Tanh or hyperbolic tangent Activation Function

- Equation : f(x) = 1—exp(-2x) / 1 + exp(-2x) or 2 *sigmoid(2x)-1
- Range : (-1 to 1)
- It also suffers vanishing gradient problem. It saturate and kill gradients.

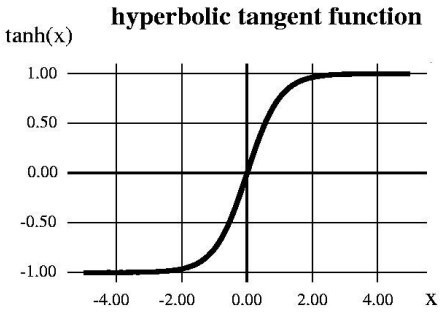


Figure 11: Tanh or hyperbolic tangent Activation Functio

3. ReLU (Rectified Linear Unit)

- The ReLU is the most used activation function in the world right now
- Equation : f(x) = max(0,x)
- Range : (0 to infinity)
- The outputs are not zero centered similar to the sigmoid activation function
- When the gradient hits zero for the negative values, it does not converge towards the minima which will result in a dead neuron while back propagation.

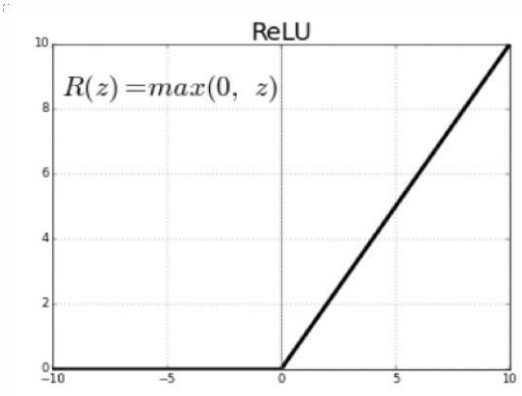


Figure 12: ReLU (Rectified Linear Unit)

4. Leaky ReLU

- To solve the ReLU problem we have leaky ReLU
- Equation : f(x) = ax for x<0 and x for x>0
- Range : (0.01 to infinity)

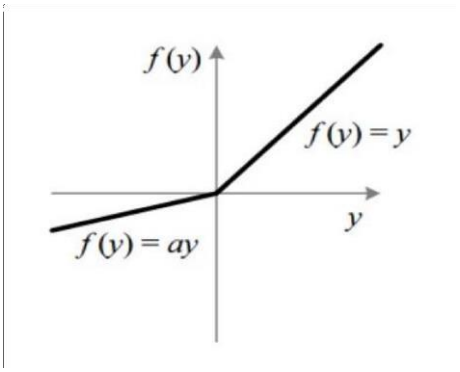


Figure 13: Leaky ReLU

5. Softmax

The softmax function is also a type of sigmoid function but it is very useful to handle classification problems having multiple classes. The softmax function is shown above, where z is a vector of the inputs to the output layer the softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

Equation 2:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

6. Loss function

- Mean Squared Error (MSE), or quadratic, loss function is widely used in linear regressions the performance measure, and the method of minimizing MSE is called Ordinary Least Squares (OSL).

Equation 3: loss (01)

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

- Cross Entropy is commonly-used in binary classification (labels are assumed to take values 0 or 1) as a loss function (For multi classification, use Multi-class Cross Entropy),which is computed by,

Equation 4: loss (02)

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})\right]$$

3.1.5.3 Optimizers

Optimization algorithms helps us to minimize (or maximize) an Objective function (another name for Error function) E(x) is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target values(Y) from the set of predictors(X) used in the model For example we call the Weights(W) and the Bias(b).

- Gradient Descent

Gradient Descent is the most important technique and the foundation of train. It Find the Minima, control the variance and then update the Model's parameters and finally lead us to Convergence.

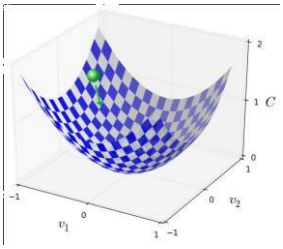


Figure 14: Gradient descent

Equation 5: Gradient descent

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2.$$

Gradient descent is majorly used to do Weights updates in a Neural Network Model, as the example update and tune the Model's parameters in a direction so that can minimize the Loss function. Now Neural Network trains via a famous technique called back propagation, in which we first propagate forward calculating the dot product of Inputs signals and their corresponding Weights and then apply a activation function to those sum of products.

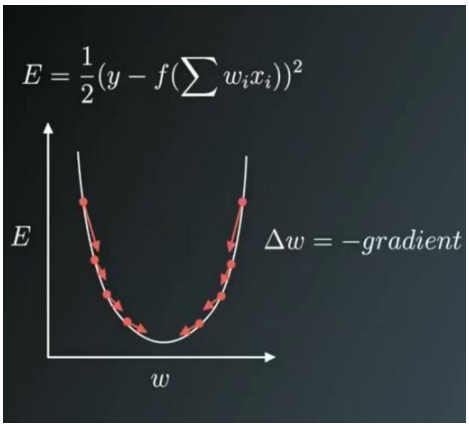


Figure 15: Gradient descent method

3.1.5.4 Convolutional Neural Network working

Convolutional Neural Networks are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on Convolutional Neural Networks.

The process of determining whether a picture contains a cat involves an activation function. If the picture resembles prior traffic sign board images the neurons have seen before, the label "name of the board" would be activated. Hence, the more labeled images the neurons are exposed to, the better it learns how to recognize other unlabeled images that call this the process of training neurons.

There are four layered concepts we should understand in Convolutional Neural Networks:

1. Convolution
2. ReLu
3. Pooling
4. Dense

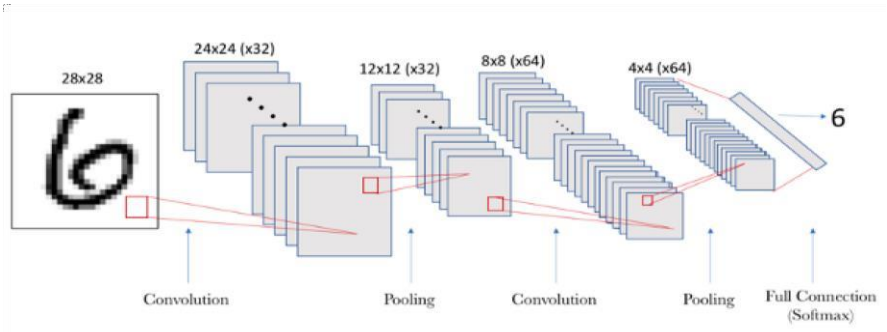


Figure 16: Convolutional Neural Networks with layers

There are multiple renditions of X and O's. This makes it tricky for the computer to recognize. But the goal is that if the input signal looks like previous images it has seen before, the "image" reference signal will be mixed into, or convolved with, the input signal. The resulting output signal is then passed on to the next layer.

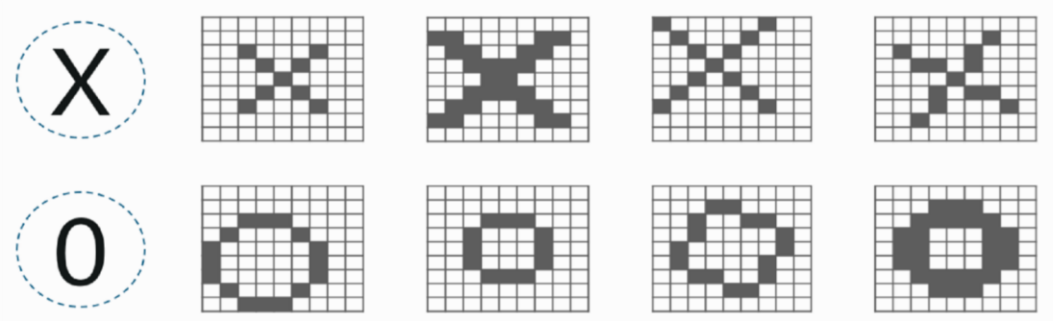


Figure 17: Multiple renditions of X and O

In this example, the computer understands every pixel. In this case, the white pixels are said to be -1 while the black ones are 1. This is just the way implemented to differentiate the pixels in a basic binary classification.

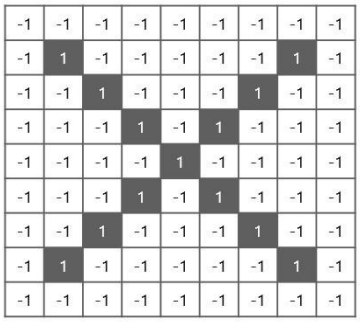


Figure 18: Pixel of image

Just normally search and compare the values between a normal image and another ‘x’ rendition, can get a lot of missing pixels.

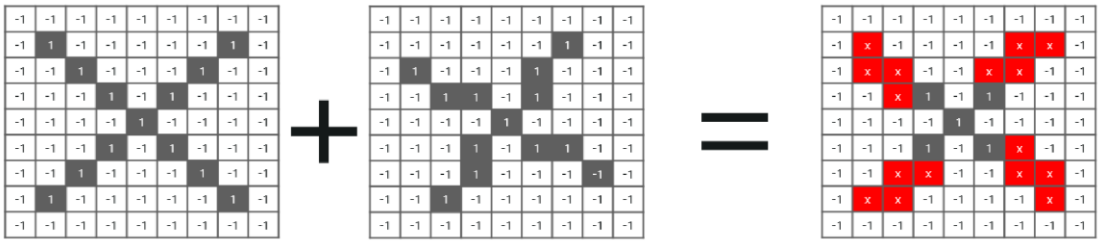


Figure 19: Search and compare each pixel

After take small patches of the pixels called filters and try to match them in the corresponding nearby locations to see if get a match. By doing this, the Convolutional Neural Network gets a lot better at seeing similarity than directly trying to match the entire image.

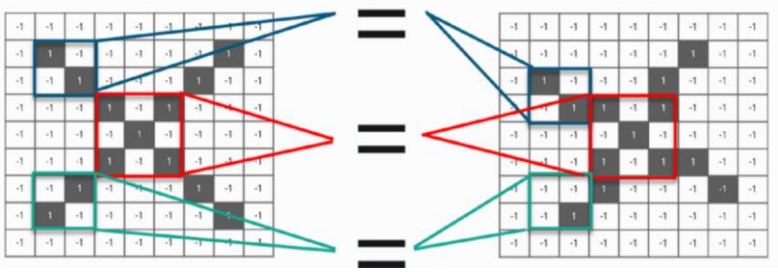


Figure 20: Match the entire image

- Convolution Of An Image

Convolution has the nice property of being translational invariant. Intuitively, this means that each convolution filter represents a feature of interest (e.g. pixels in letters) and the Convolutional Neural Network algorithm learns which features comprise the resulting reference (i.e. alphabet).There are 4 steps for convolution:

1. Line up the feature and the image
2. Multiply each image pixel by corresponding feature pixel
3. Add the values and find the sum
4. Divide the sum by the total number of pixels in the feature

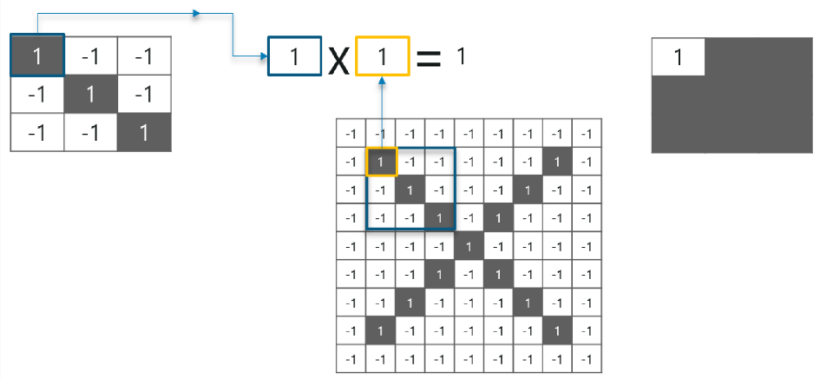


Figure 21: Convolution of an Image/ step 1

Consider the above images are done with the first 2 steps also considered a feature image and one pixel from it and multiplied this with the existing image and the product is stored in another buffer feature image.

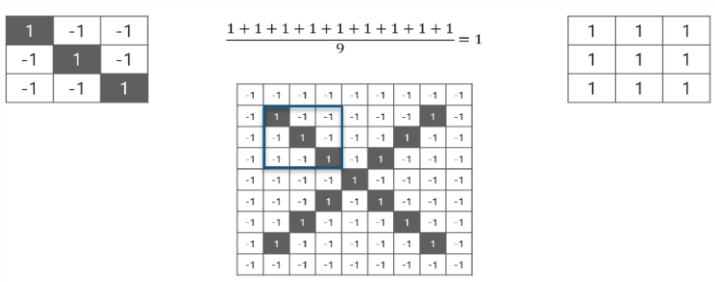


Figure 22: Convolution of An Image/ step 2

With this image completed the last 2 steps. We added the values which led to the sum. After divide this number by the total number of pixels in the feature image. When that is done, the final value obtained is placed at the center of the filtered image as shown below:

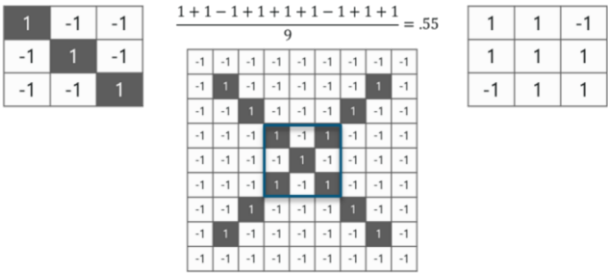


Figure 23: Convolution of An Image/ step 2

Here considered just one filter. Similarly, perform the same convolution with every other filter to get the convolution of that filter. The output signal strength is not dependent on where the features are located, but simply whether the features are present. Hence, an alphabet could be sitting in different positions and the Convolutional Neural Network algorithm would still be able to recognize it.

- Sobel Gradients

The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image.

In theory at least, the operator consists of a pair of 3×3 convolution kernels as shown in Figure 24 One kernel is simply the other rotated by 90°.

Here considered just one filter. Similarly, perform the same convolution with every other filter to get the convolution of that filter. The output signal strength is not dependent on where the features are located, but simply whether the features are present. Hence, an alphabet could be sitting in different positions and the Convolutional Neural Network algorithm would still be able to recognize it.

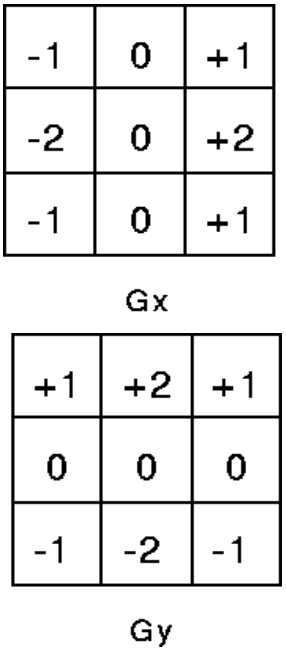


Figure 24: Sobel Gradients process

- ReLU Layer

Rectified Linear Unit (ReLU) transform function only activates a node if the input is above a certain quantity, while the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable.

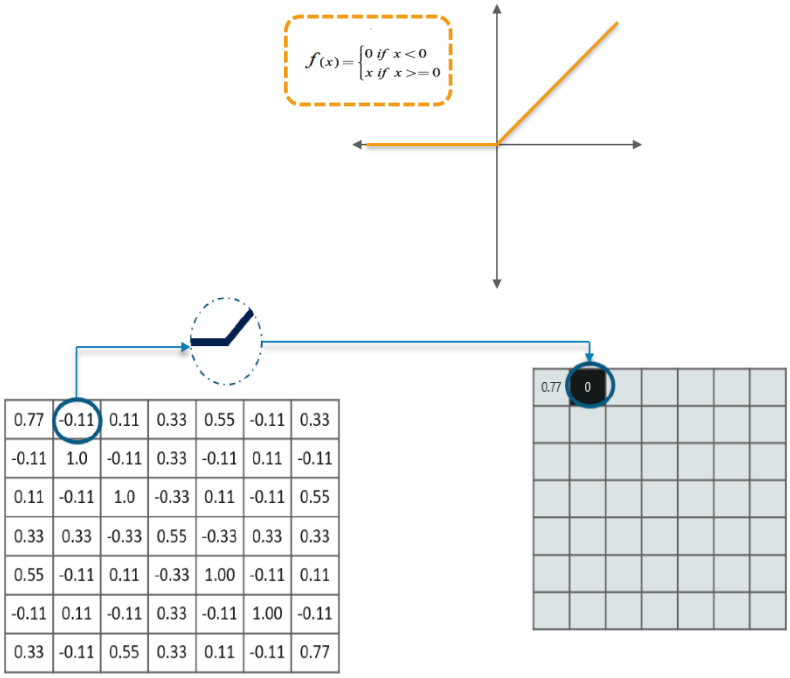


Figure 25: ReLU Layer process/ step 1

The main aim is to remove all the negative values from the convolution. All the positive values remain the same but all the negative values get changed to zero as shown below:

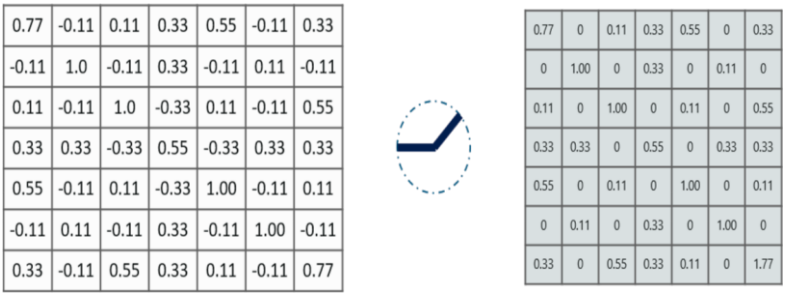


Figure 26: ReLU Layer process/ step 2

- Pooling layer

In this layer we shrink the image stack into a smaller size. Pooling is done after passing through the activation layer. It has the following 4 steps:

1. Pick a window size (usually 2 or 3)
2. Pick a stride (usually 2)
3. Walk your window across your filtered images
4. From each window, take the maximum value

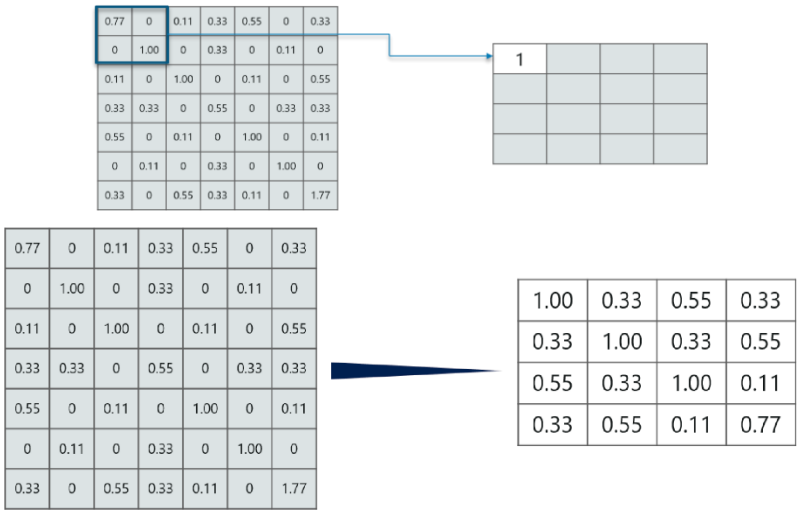


Figure 27: Pooling Layer process/ step 1

Took window size to be 2 and got 4 values to choose from. From those 4 values, the maximum value there is 1 so we pick 1. Also, note and after started out with a 7×7 matrix but now the same matrix after pooling came down to 4×4.After, move the window across the entire image. The procedure is exactly as same as above and we need to repeat that for the entire image.

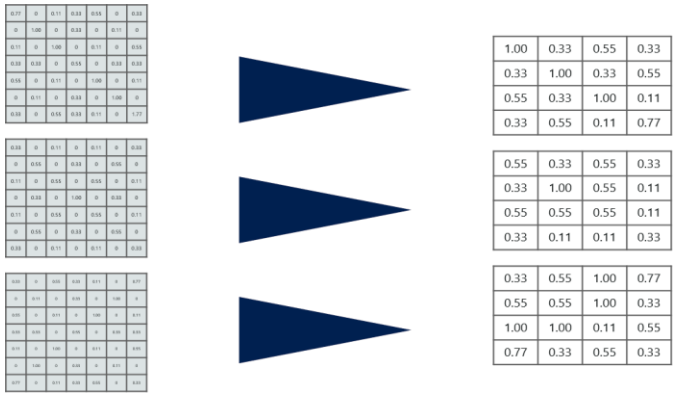


Figure 28: Pooling Layer process/ step 2

- Stacking Up The Layers

So to get the time-frame in one picture , here with a 4×4 matrix from a 7×7 matrix after passing the input through 3 layers – Convolution, ReLU and Pooling as shown above:

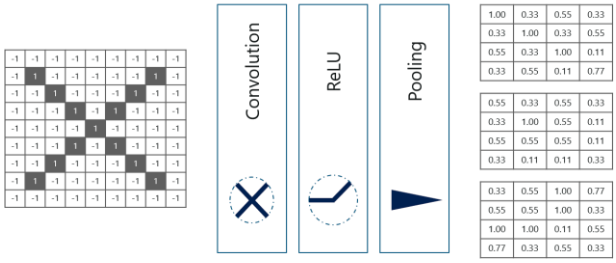


Figure 29: Stacking Up the Layers

Perform the 3 operations in an iteration after the first pass. So after the second pass we arrive at a 2×2 matrix as shown below:

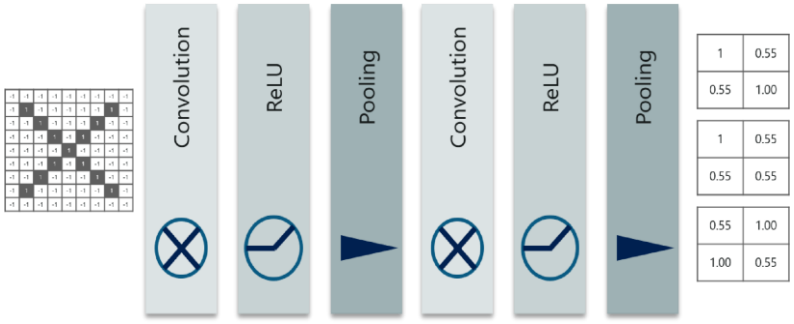


Figure 30: Stacking Up the Layers additional

- Dense Layer- Last Layer

This mimics high level reasoning where all possible pathways from the input to output are considered. Also, fully connected layer is the final layer where the classification actually happens. Here can take filtered and shirked images and put them into one single list as shown below:

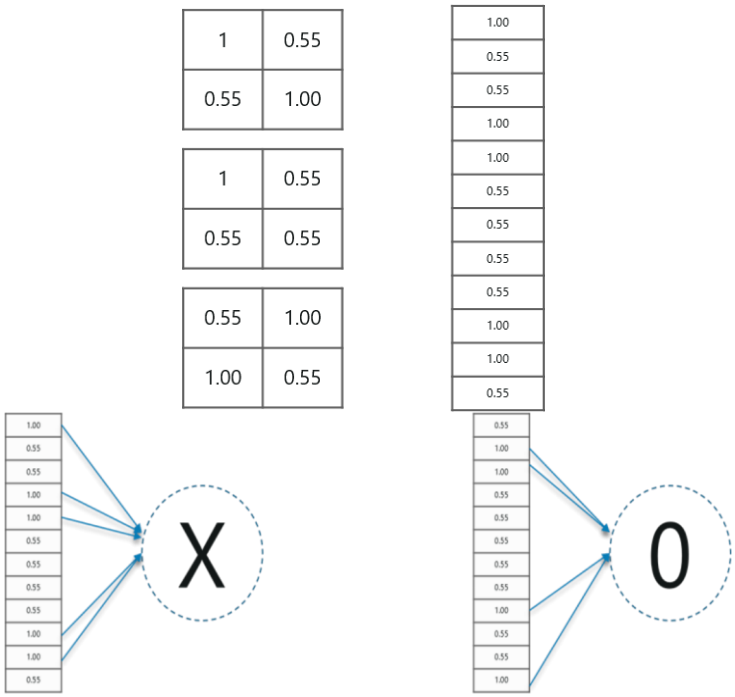


Figure 31: Dense Layer- Last Layer get output

Thus next, when we feed in, 'X' and 'O' there will be some element in the vector that will be high. Consider the image below, as you can see for 'X' there are different elements that are high and similarly, for 'O' we have different elements that are high. When the 1st, 4th, 5th, 10th and 11th values are high, we can classify the image as 'x'. The concept is similar for the other alphabets as well – when certain values are arranged the way they are, they can be mapped to an actual letter or a number which require.

In the above image, we have a 12 element vector obtained after passing the input of a random letter through all the layers of network and also can make predictions based on the output data by comparing the obtained values with list of 'x' and 'o'!. When can divide the value we have a probability match to be 0.91.

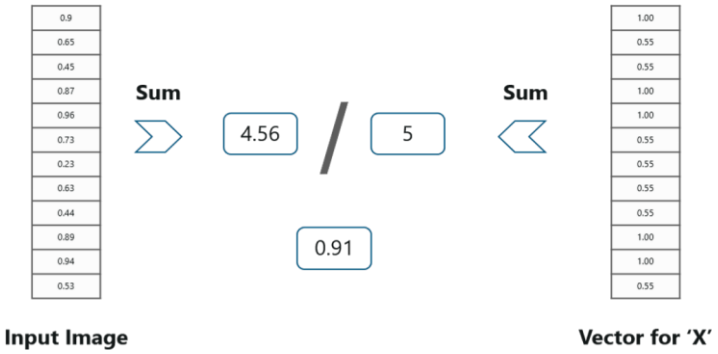


Figure 32: Decide final value

- Dropout

Dropout is a regularization technique. On each iteration, randomly shut down some neurons (units) on each layer and do not use those neurons in both forward propagation and backpropagation. Since the units that will be dropped out on each iteration will be random, the learning algorithm will have no idea which neurons will be shut down on every iteration; therefore, force the learning algorithm to spread out the weights and not focus on some specific features (units). Moreover, dropout help improving generalization error by:

- Since we drop some units on each iteration, this will lead to smaller network which in turns means simpler network (regularization).
- Can be seen as an approximation to bagging techniques. Each iteration can be viewed as different model since dropping randomly different units on each layer. This means that the error would be the average of errors from all different models (iterations). Therefore, averaging errors from different models especially if those errors are uncorrelated would reduce the overall errors. In the worst case where errors are perfectly correlated, averaging among all models won't help at all; however that in practice errors have some degree of uncorrelation. As result, it will always improve generalization error.

3.1.6. Image Processing

Image processing is a method which is used to perform an operation which is related with obtaining a large image or to extract some information. It is also considered to be a signal processing type where the input will be an image and the output can be an image or a characteristic or a feature. It is considered as a vast growing technology in the world. The image processing is done depending on,

- Importing the image through image acquisition tool
- Analyzing and manipulating the image
- Output as an altered image or as a report related to image analysis.

Basically, the image processing is done through two types of operations which are known as analogue and digital image processing. Analogue is meant to be the hardcopies like photos and digital can be made changes with the help of the computers.

3.1.6.1 Canny edge detection

It is important for many image recognition applications to detect edges in an image containing one or more color channel. This concept returns a monochromatic image which contains back and white pixels describing whether each pixel is part or edge or not. For human eye the outline of an original image will appeared to be marked in the output image as given below.

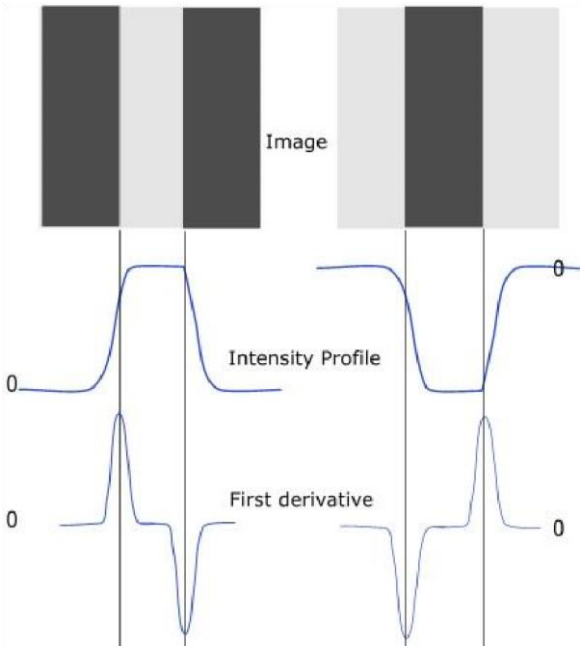


Figure 33: Canny edge detection

An edge is defined as a set of connected pixels that lie on the boundary between two regions which are supposed to be separated by a gradient in color values. Both regions are distinguished by two different color. If the gradient is exceeded from the threshold value, it is classified as an edge. The process will be complicated if it is interrupted with a noise to edge. A smoothing algorithm is requested to be used as solution due to vary in the gradient. Algorithm such as Gaussian filter is much usable. In addition, by decreasing the image resolution, the areas which are in relation with noise can be averaged into single pixel value, while decreasing the execution time required for edge detection algorithm due to less amount of data is required to be handle. This approach allows the detection of edges/ gradient changes in a one-dimensional array of values. The process is expandable for the handling of two-dimensional images which has the chance of containing several color images.

Hough Transform is a popular technique to detect any shape and it can represent that shape in mathematical form. It can detect the shape even if it is broken or distorted a little bit.

3.1.6.2 Hough Line Detection

Hough Transform is a popular technique to detect any shape and it can represent that shape in mathematical form. It can detect the shape even if it is broken or distorted a little bit.

A line can be represented as y = mx+c or in parametric form, as rho = x cos(theta) + y sin(theta) where rho is the perpendicular distance from origin to the line, and theta is the angle formed by this perpendicular line and horizontal axis measured in counter-clockwise .Check below image:

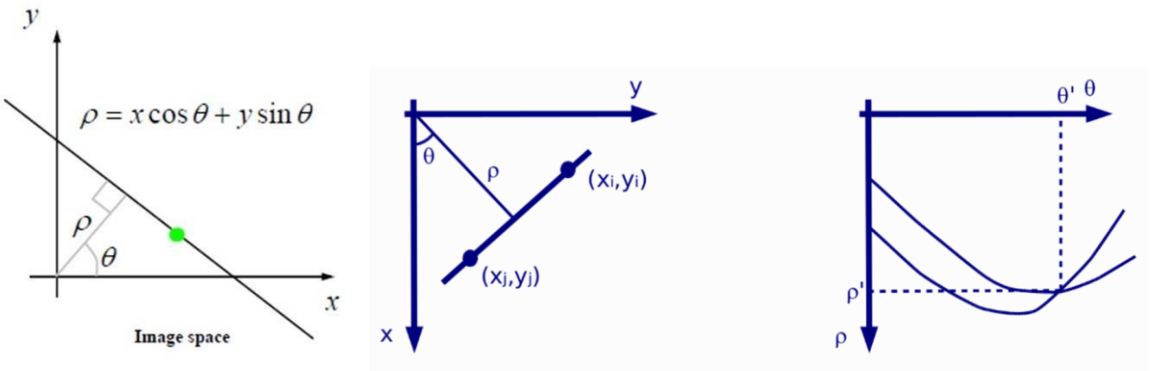


Figure 34: Hough Line Detection method

Mostly the edge detection algorithm is not able to detect all the edges which are available in an image due to the noise/ gradient which is below the threshold. To increase the quality of edge detection, a linking algorithm is designed to assemble edge pixels into meaningful edges or region boundaries. Here Hough transform can be used to link the edges into line. It is known as a global approach which works with the entire image.

3.1.7. DC Motor Control

The DC motors are rotary electrical devices which is supposed to actuate with the flow of current. It is important to control the rotational speed, torque and position of the motor. Here some equations are mentioned in related to DC motor calculations.

Equation 6: DC motor characteristic equation

$$U_A = R_A I_A + E$$
$$E = K_2 \phi \omega$$
$$M = K_2 \phi I_A$$

UA: Voltage over motor

RA: Internal resistance

K2: Device constant

E: Voltage drop

M: Motor torque

IA: Draw current

For controlling the rotational speed and torque it is required to regulate the voltage UA due to the internal resistance is constant.

3.2 Methods

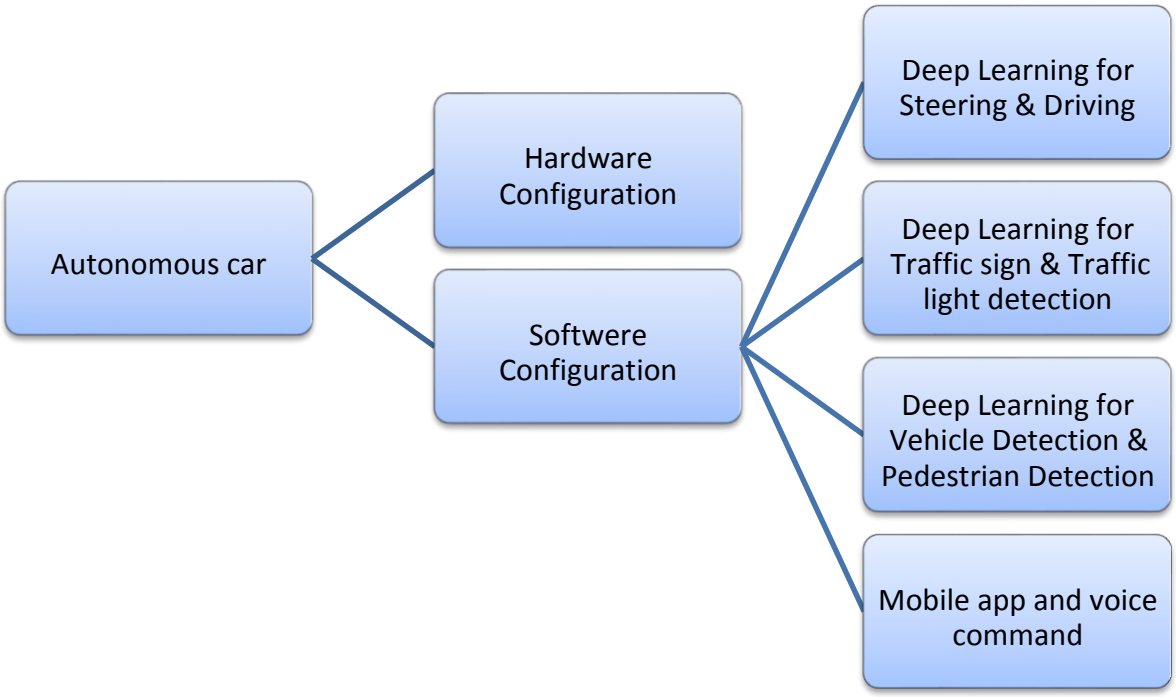


Figure 35: Structure of the Methodology

3.2.1. Literature review and data collection

Prior to setting scope and initializing designs, a literature review was done. Previous work carried out were thoroughly studied and the methods and technologies adopted by them were compared. Thereafter, it was possible to identify the problems they have encountered and provide solutions to those problems according to the predetermined budget and time constraints. Finally, goals for this project was set and the design stage of the self-driving car was initiated.

3.2.2. Design and analysis

Once the existing technologies were reviewed, conceptual designs were formed. Feasibility of adding certain features were considered with respect to the budget and time constraints. Thereby, it was decided that the AI car will consist of a basic design with RC car, raspberry pi, NVidia Jetson Nano and camera.

Design Considerations:

- Operating distance – maximum 50m in path
- Moving - 2 DC motor to wheels
- Turn - servo motor to steering angle
- PCB – control battery voltage and current
- Body – RC car(plastics)
- 3D print parts – battery, microcontrollers ,PCB and Camera holder
- Controller – Raspberry pi, NVidia jetson Nano
- Power -7.4V two LiPo battery,12V battery
- Camera module - Raspberry Pi camera
- Motor controller- L298N DC motor controller and 16-CH Servo Motor Controller/PWM Servo Driver Board
- Power controllers- Two Buck controllers

Parts were designed and main problem was training path. Thus it should be black and should have white line on the path. Furthermore path should be long and ability to fix the sign boards and traffic lights. Since this would complicate the path design and further extend the budget, printed flax was applicable and decided to be included that flax, provided that the time and budget constraints are met.

The initial idea for designing the car body was with a material like plastics. Because plastic is very expansive and can't design simply there have an option would have been ideal to produce the body in vehicle shape desired. However, it was not cost effective and would have consumed more time and effort. Therefore RC car was choose because the design will be modified with the procured parts which were available in the market.

The AI base self-driving car is the most important feature of this research project. The basic idea for the Self-driving car is to have RC car body with two DC motors. Subsequently it can run parallel to same move speed. Therefore the DC motors can be controlled if speed was can be controlled. Because Motors PWM can be changed by code. Microcontroller, batteries and camera module, motor drivers and other holders were fixed in to the car body.
The original design was intended to have the self-driving car divided into two segment.
RC car, and printed parts.

3.2.2.1 Final car setup

Generally the car is attached a remote control. Therefore signals sends to the steer the wheels that depend on which button press in remote control. In this case, vehicle is controlled purely by computer programming. Thus, several extra parts were attached in to the car and also servo connected to the onboard computer. Also Camera was needed to collect images and motor controller.

Accordingly RC car was a one of most components of whole system. There were several things that want to fix. The main problem was take and control steering angle in the vehicle. Therefore front of the car should have be sufficient space to get steering angle (right, Center, left). It was possible make such car drive and should have enough angle to get huge bend in the training track why if not vehicle goes over or under steering. The last thing was considered the size of vehicle that mean more indeed scale of the model of vehicle. Scales could range from 1/32 to 1/8 also beyond of that range. Only limit was that the car wanted to be big sufficient to carry every parts and batteries.

Final Choice – Buggy Radio Car 1/12 2.4GHz Exceed RC Car

1/12 scale size was choose sufficient to fit, carry all compulsory components. It could be trained in the making path. Thus, normally 10 000 images were taken to drive this RC car hence it could be strong. Also it was inexpensive, had best servo and driven very well.

Servo, Motors

Usually 1/12 RC car had 3 DC motors to drive the wheels and steer. Therefore Steering measurements were difficult to taken with steering DC motors. Subsequently servo motor

(MG996R Servo Motor /Analog) was used to measure the angles. Therefor measurement could get degree by degree and servo motor was adjusted easily. Also could get maximum steering angle.

Camera

Camera resolution was not very important as well as down sample the input to decrease training time. Furthermore images not should be very quality. Because more than 10 000 images were got hence storage of image was important. Wider angle camera was advantageous, as it could be captured more images and information about car's surrounds.

Final choice: Raspberry pi night vision camera

The Raspberry pi night vision camera was an excellent tradeoff between prices and improve quality. It was sufficient for path that made for training.

Figure 36: Raspberry pi night vision camera

Onboard Computer

Depending on scale of this self-driving project, Powerful onboard computer was choose from a wide variety. NVidia Jetson Nano was very powerful onboard microcontroller that used in to this project. Also Raspberry pi was used in this project for face recognize. Raspberry pi was stood on other end of spectrum as lower budget single board computer.

Final choice: NVidia Jetson Nano, Raspberry pi 3 model B

NVidia Jetson Nano was chose why it was performer then Raspberry pi 3. If raspberry pi was used in this project while real time neural network had big delay. That was a main reason why NVidia Jetson Nano was choose. Raspberry pi also had a Build in Wi-Fi module so that was also well accepted by users. Consequently raspberry pi and NVidia Jetson Nano were the accessibility of tensorflow. So that was another important factor. But as mentioned, NVidia Jetson Nano had relatively good processing power than raspberry pi.

Table 1: Performance of two boards

Raspberry pi 3 model B	NVidia Jetson Nano
• SoC: BCM2837B0 64-bit systemon-chip with four ARM CortexA53 • CPU cores clocked at 1.4GHz • CPU: 4x ARM Cortex-A53, 1.4GHz • GPU: Broadcom Video Core IV • RAM: 1GB LPDDR2 SDRAM • Networking: Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi • Bluetooth: Bluetooth 4.2, Bluetooth • Low Energy (BLE) • Storage: micro SD • GPIO: 40-pin header, populated Ports: HDMI, 3.5mm analog audiovideo jack, four USB 2.0, Ethernet, Camera Serial Interface (CSI), • Display Serial Interface (DSI)	• GPU:NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores CPU: Quad-core ARM Cortex-A57 MP Core processor • Memory:4 GB 64-bit LPDDR4, • 1600MHz 25.6 GB/s • Storage:16 GB eMMC 5.1 • Camera:12 lanes (3x4 or 4x2) MIPI CSI-2 D-PHY 1.1 (1.5 Gb/s per pair) • Connectivity: Gigabit Ethernet, M.2 • Key E • Display: HDMI 2.0 and eDP 1.4 • USB:4x USB 3.0, USB 2.0 Micro-B

Motor and Servo Controller

To control the DC motors and servo motor, Specific PWM signals were controlled the motor that was made easier. NAVIO2 had lot of useful sensor like accelerometers, magnetometers and gyroscopes. It was 164 dollars. Unfortunately, it was very expansive for in this type of project.

Final Choice: Adafruit PCA9685 16 Channel 12 Bit PWM servo driver, L298N Dual H-Bridge DC Motor Controller Driver Module

Adafruit PCA9685 16 Channel 12 Bit PWM servo driver was chose why it was could control PWM speed. It was relatively small, so not get in the way. Finally it was easy to set up also with publicly software library. Also L298N Dual H-Bridge DC Motor Controller Driver Module was very easy, primary and simple.

Additional improvements

Node MCU (ESP8266 Wi-Fi Programming & Development Kit), Humidity and temperature sensor (HYT939), relay, reed switch, Soil Hygrometer Humidity Detection Moisture Water Sensor Module, Rain sensor

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The upload procedure for ESP8266 boards is a little different from the normal Arduino procedure. Most Arduinos will automatically reset when a new program is being uploaded, and will automatically enter programming mode. On some ESP boards you have to manually enter programming mode, and on the bare-bones modules, you even have to reset them manually. However, there are some simple circuits you can use to get automatic uploads. Other sensors and components use to make mobile app and voice command control.

Final design

Figure 37: Final design of AI car

3.2.3. Manufacture and assembly

This requires a certain combination of parts. The main assembly consists of the flexible boxes and plates (battery and PCB holder) on the CR car. Moreover most parts of the project were procured as readymade products from the market. Certain parts were custom designed as they could not be bought from the market. The assembly process is as follows.

- Procuring readymade parts
- Custom parts were fabricated through,
    - 3 D printing
    - Machining
- Assembly of the Flexible RC car

There was a requirement for track for the training of the vehicle. So, the expected track was built using the paint in a banner. The track was designed based on the angle rotation which the car can turn and other specific parameters such as the width of the vehicle and camera positioning etc. While obtaining the images due to the positioning of the camera the vehicle went outwards the line and made some turns but the requirement of final precise model. So, data for the requirement was obtained during running in the real path. The placement of the camera was in front of the vehicle because when a driver is driving, he should be able to observe the situations at the front. So, corresponding to it, the pi camera was placed in front of the vehicle.

Figure 38: Image of the path

3.2.4. Software development

The development of software in mainly done in computers where it has a multiple operating systems and programming environments, without a built in GPUs. For the access of the GPU a solution based on remote desktop was set up on the main development board, Raspberry Pi. Research and the assessment of the framework which contribute on the management of the DNN were it is relevance; performance and the learning curve of the framework were important aspects for the project. Due to this, it was concluding that the TensorFlow for the providence of effortless backend in managing neural networks, combined with TFLearn for the providence of the frontend for structures and utilities in TensorFlow, was the expected framework for the purpose. With these frameworks we were able to utilize the GPU on the Nvidia Jetson Nano development board for the usage of the real time training and obtaining images during the running in real time. The training process was done through COLAB which is a combination of Python and Jupyter NoteBook which is used to train the DNN. The goal was to decrease the time and low occupation of the GPU of the board due to it required to run 6 DNN at simultaneously. Furthermore, by using this a key advantage was observed which was the ease use and the TFLean consist of a high-level API for DNN and it enabled rapid development and prototyping which was very important in trying different configurations and the ability to optimize the significant in a fast and easy way. The DNN was created using TensorFlow in the combination with Keras. Mostly after selecting the Jetson Nano with the research method were able to choose TFLearn over Keras where TFLearn is optimized for using the TensorFlow, or using Theano as a backend framework.

3.2.5. Deep Learning for Steering and Driving

3.2.5.1 Simulate a self-driving car

In this section self-driving car fully training and running in a simulated environment. Audacity was recently open source their simulation so the build a simulator specifically for self-driving cars. It was built for their self-driving car an inner degree any people. So it was built with unity and could recommend most of research papers. So it was added a bunch of pre-built scripts. Also control things, momentum and acceleration and all sorts of things would be in a selfdriving car simulation. Programmer could modified that things as well as could can change in code.

Step process of this simulation have three steps.

- Data generating part
- Training part
- Testing part

Thus, first part was data generation part. Fist NVidia car simulation was studied. So nine layer convolutional network was built by them and attached three cameras to the head of car. While human driver was driven the car while the cameras were attached and all the car has to do was collect data and steering command. That the human driver was inputting as well as the camera feed so the three set of camera images were come from all three cameras at the same time and that was the data generation part. There were images from the center, left and right cameras with the associated steering angle speed throttle and break. So there was four physics variables as well as the static images that were come in and it was saved.

Second part was training part. So the human driver was driven around and recoded all that data. After data was get a machine to clone that behavior. So this process was Behavioral cloning. So doing the behavioral cloning, nine layers were built of convolutional network. Nine layers was in keras. There was nine lines of code and it would base off of NVidias and to end learning for selfdriving paper. That car model was trained whole 72 hours in different driving conditions (sleet, rain hail, wet rain all such things). Therefore here is a hardware design.

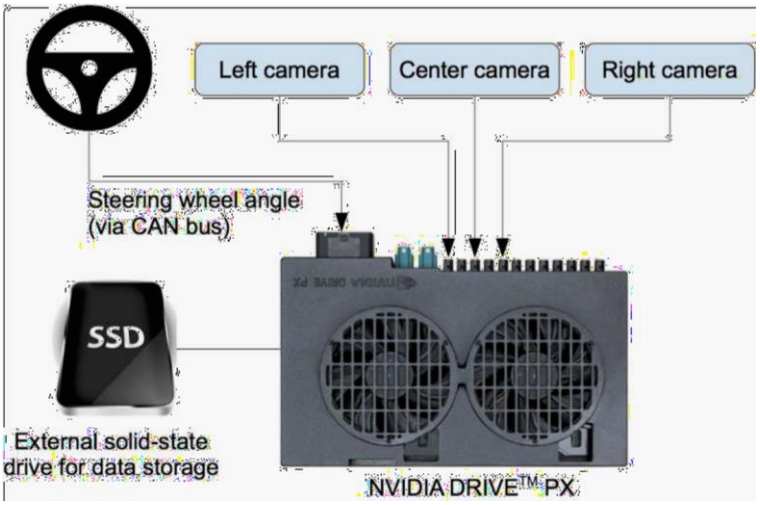


Figure 39: Level view of the data collection system

So hard ware design was obviously a steering wheel. But the steering wheels was attached to a controller. That was the so many acronyms here because the controller area network bus and it was feeding in those four variables. Three cameras were feeding in continuous streams. All the images (frame by frame) were seeing by the car and those were fed in to this NVidia drive PX computer. It was essentially like a motherboard with a cluster of jeep on board GPUs attached to it and stored the results.

So regarding the research paper was recorded instead of inputting the steering command directly. So it was inputted 1/R (R= training radius for driving straight). Now can found it may be turn smoother and it was made autonomous angle that the car moved independent of the geometry of the car. So the car was faced left or right not matter. It was deepened on what the cameras see. So that steering angle was moved by signal. It was independent of the car directions. So software was designed that it was depend on a code.

Process of the autonomous self-driving car

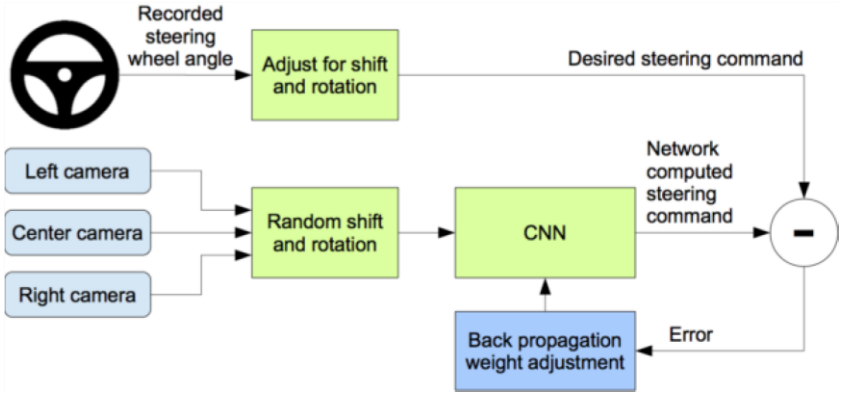


Figure 40: Process of the autonomous self-driving car

There were have two inputs (Steering angle and sets of cameras) and three set of images (left, right and center). So input labels were images and output labels were steering angle. So let's assuming the generated dates were 72 hours of human generated driving data. So both of result (our data and NVidia data) could compared together. So error or loss could computed between the two. So cameras Angles and the steering wheel variable were human generated data and

compressed both this value in to one value. There had error value between that dataset and used propagation to update weights base on that error value. So images were faded in to CNN (Convolutional neural network) then it was computed a proposed during commit. That was predicted steering command. So difference between the predicted steering command and actual steering command were depended of training data. Then proposed command was expected to command right label the target versus the actual output. The weights of the CNN were adjusted to bring the CNN output closer to the desired output and the weight adjustment were accomplished via back propagation. So in the CNN, prediction was the angle of the steering with just one output. So that was for training.

For testing, this Autonomous model could think of the terms of a simulator as a server client architecture. Server was gone to be the simulator itself. It mean app was downloaded and client was gone to be python program that write. So it was gone just feedback loop. Client was piped in steering angles and throttles to the server and the server was typed backup images from car and steering angles. So it could train right it was a feedback loop.

Then the augmented images were fed through the network which is configured with 10 layers and resulting in the steering angle as the required output.

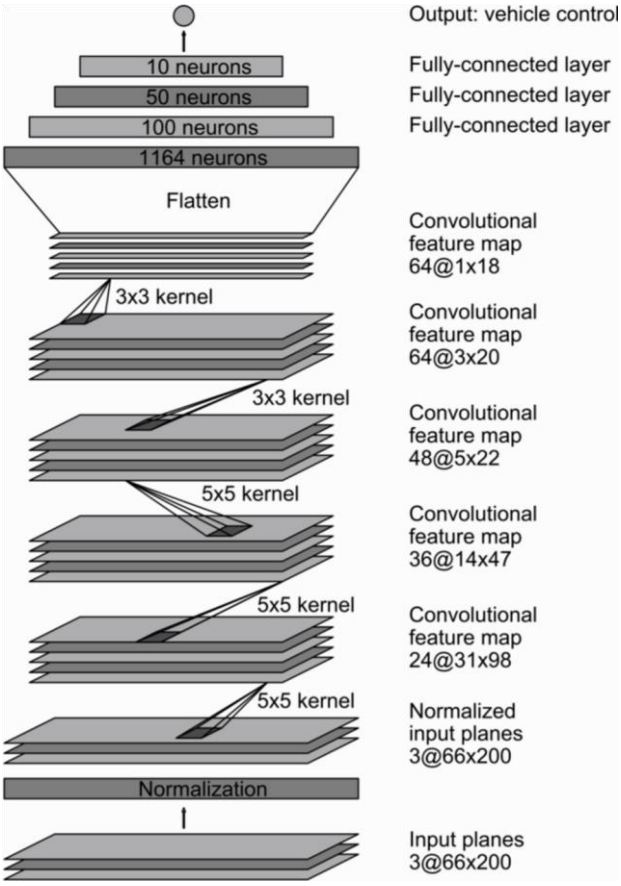


Figure 41: NVIDIA Neural Network

The steering angle of the network were thereafter compared with the correct steering angle of the image from the data set, the difference beside the steering angle where then propagated backwards to make adjustments in the weights of the network to reduce the loss function.

Udacity Simulator

Udacity has built a simulator for self-driving cars and made it open source for the enthusiasts, so they can work on something close to a real-time environment. It is built on Unity, the video game development platform. The simulator consists of a configurable resolution and controls setting and is very user friendly.

The first actual screen of the simulator can be seen in Figure (42) and its components are discussed below. The simulator involves two tracks. One of them can be considered as simple and another one as complex that can be evident in the screenshots attached in Figure (42). The word “simple” here just means that it has fewer curvy tracks and is easier to drive on, refer Figure.42

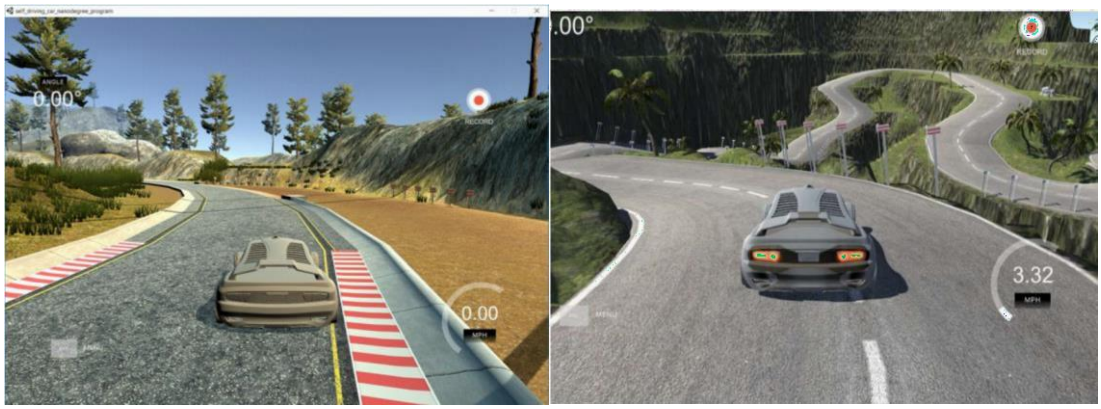


Figure 42: Track 1 and Track 2

There are two modes for driving the car in the simulator: (1) Training mode and (2) Autonomous mode. The training mode gives you the option of recording your run and capturing the training dataset. The small red sign at the top right of the screen in the Figure.depicts the car is being driven in training mode. The autonomous mode can be used to test the models to see if it can drive on the track without human intervention. Also, if you try to press the controls to get the car back on track, it will immediately notify you that it shifted to manual controls.

3.2.6. Stereo Vision for distance measuring

Stereo vision is a technique used for recording and representing stereoscopic (3D) images. A pair of cameras is required to implement stereo vision. An illusion of depth is created by using two pictures taken at slightly different positions. Stereo image is captured using a web camera and a horizontal angle of view in DX-format equal to 30. The cameras are aligned in a parallel stand with a fixed distance. The object's distance is measured when it enters the overlapping views of the two cameras.

The distance B is expressed as the sum of distance B1 and B2

$$B_1 = D\tan\theta_1 \text{ and } B_2 = D\tan\theta_2$$

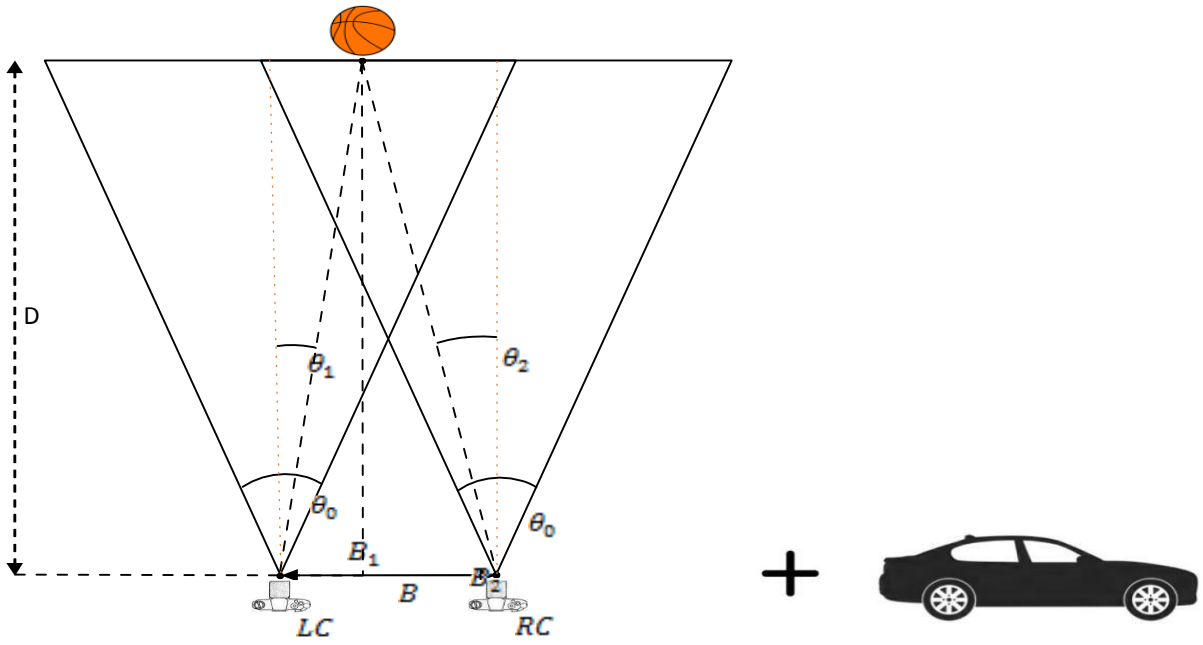


Figure 43: Stereo Vision for distance method

Equation 7: Distance with angels

$$D = \frac{B}{\tan\theta_1 + \tan\theta_2}$$

Equation 8: Distance by left angle

Using Figure 3 and trigonometry

$$D = \frac{\tan\theta_1}{x_1} = \frac{\tan\left(\frac{\theta_0}{2}\right)}{\frac{x_0}{2}} \quad \text{in LC view and}$$

Equation 9: Distance by right angle

$$D = \frac{\tan\theta_2}{-x_2} = \frac{\tan\left(\frac{\theta_0}{2}\right)}{\frac{x_0}{2}} \quad \text{in RC view}$$

Equation 10: Ratio of distance and tan values of angels

Then

$$\frac{x_1}{\frac{x_0}{2}} = \frac{\tan\theta_1}{\tan\left(\frac{\theta_0}{2}\right)}, \quad \frac{-x_2}{\frac{x_0}{2}} = \frac{\tan\theta_2}{\tan\left(\frac{\theta_0}{2}\right)} \quad \rightarrow \quad \tan\theta_1 = \frac{2x_1\tan\left(\frac{\theta_0}{2}\right)}{x_0}, \quad \tan\theta_2 = \frac{-2x_2\tan\left(\frac{\theta_0}{2}\right)}{x_0}$$

Equation 11: Total of tan values

$$\tan\theta_1 + \tan\theta_2 = \frac{2\tan\left(\frac{\theta_0}{2}\right)(x_1 - x_2)}{x_0}$$

Equation 12: Final distance

By implementing in Eq. (1) distance D can be measured as follows:

$$D = \frac{Bx_0}{2\tan\left(\frac{\theta_0}{2}\right)(x_1 - x_2)}$$

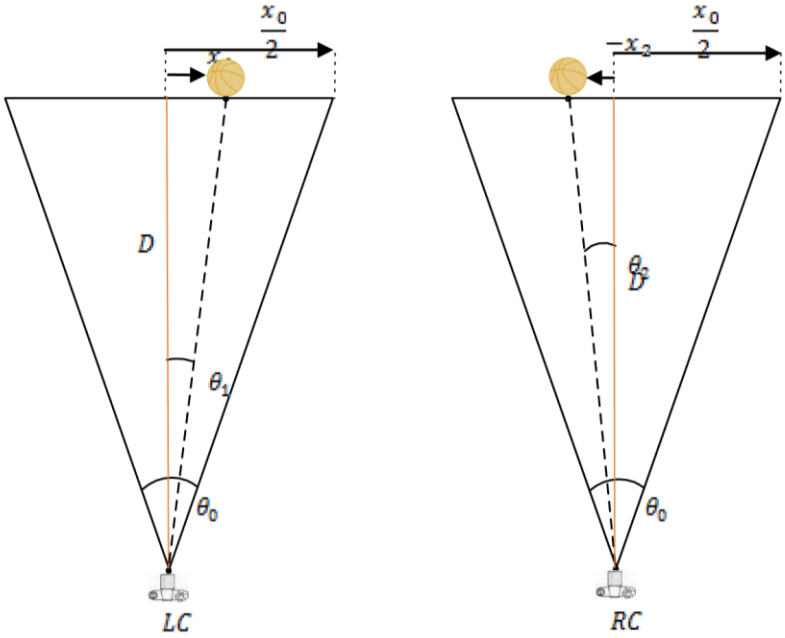


Figure 44: Measuring distance

3.2.7. Deep Learning for vehicle detection and passenger detection

Object detection is actually a two-part process,

- Image classification
- Image localization.

Image classification is defining what the objects in the image are, like a car or a person, while image localization is providing the detailed location of these objects, as seen by the bounding boxes above.

To perform image classification, a convolutional neural network is trained to identify various objects, like traffic lights and perambulators. A convolutional neural network performs convolution operations on images in order to classify them. So CNN (Conclusion neural network) should be build.

3.2.7.1 Vehicle detection

For the implementation of the vehicle detection and pedestrian detection were objected to be done through Cascade Classifiers which is used to detect objects with the trained data or own trained data. The classifier is known to be trained with few hundred of sample views of the particular object requirement which is known as positive examples with the same scale size. Negative images are known to be the arbitrary image of the same size.

After the classifier is trained then it can be used in the region of interest which it was used to train in an input image. The output of the classifier will be one if the image shows us the exact same in the region of interest or else the expected output will be zero. For searching every segment in the image, the search image can be moved across the image and check every location with the classifier. The classifier can be easily resized to find the objects within an image with different size of region of interest.
For this an own classifier was created for different vehicles such as cars, buses, motor bikes, vans etc. Data set of hundred thousand for each requirement was used for the training where the data were augmented with brightness, contrast, rotation, enlarging and etc. The trained files were used in the format of XML and the expected results were obtained.

Figure 45: Vehicle detection using classifiers

3.2.7.2 Passenger detection

The main contributions are a novel and real-time Deep Learning person detection and a standardization of personal space that can be used with any path planer. In the first stage of the approach, we propose to cascade the Aggregate Channel Features (ACF) detector with a deep Convolutional Neural Network (CNN) to achieve fast and accurate Pedestrian Detection (PD). For the personal space definition that used a mixture of asymmetric Gaussian functions, to define the cost functions associated to each constraint. Both methods were evaluated individually. The final solution (including both the proposed pedestrian detection and the personal space constraints) and tested in a typical domestic indoor scenario and test path in distinct experiments.

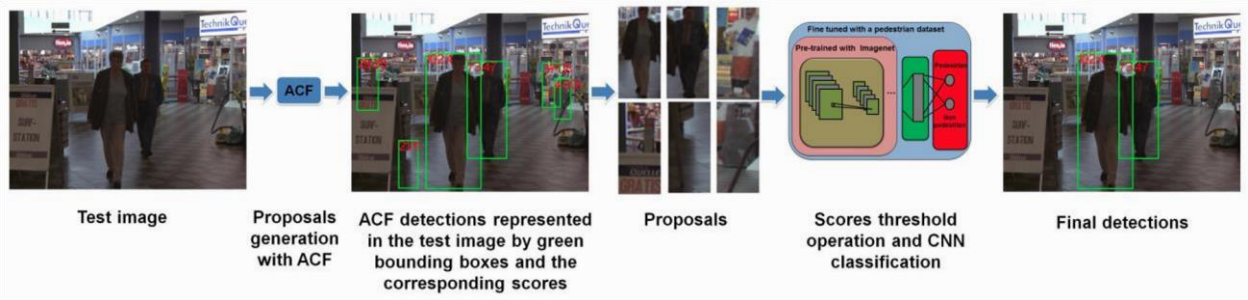


Figure 46: Process of Pedestrian detection

3.2.8. Deep Learning for traffic light detection and traffic sign detection

Traffic signs are component and parcel of our street infrastructure. They provide critical information for road users, sometimes compelling recommendations, which in turn requires them to adjust their driving behavior to ensure that they abide by whatever road regulation is currently being enforced. Without such helpful indications, we would most probably face more accidents, as riders would not receive critical feedback on how quickly they could safely go, or inform about road works, sharp turns, or school crossings ahead. Every year, around 1.3 million individuals die on highways in our modern age. Without our road signs, that amount would be much greater.

Standard computer vision methods have traditionally been used to detect and classify traffic signs, but these have required considerable and time-consuming manual work to handle important image features. Instead, by applying deep learning to this issue, we generate a model that reliably classifies traffic signs, learning by itself to define the most suitable characteristics for this issue. A deep learning architecture should be created in this project that can identify traffic signs.

The dataset is divided into sets of training, testing and validation, with the following features:

- Images are 32 (width) x 32 (height) x 3 (RGB color channels)
- Training set is composed of 9356 images(take different angles)
- Validation set is composed of 3365 images
- Test set is composed of 7256 images
- There are 43 classes (e.g. Speed Limit 20km/h, No entry, Bumpy road, etc.)

Moreover, we will be using Python 3 with Tensor flow to write our code.

There is sample of the images from the dataset, with labels shown above the row of concomitant images. Some of them are quite dark so we will look to build contrast a bit later.

Figure 47: Types of sign boards and convert into gray

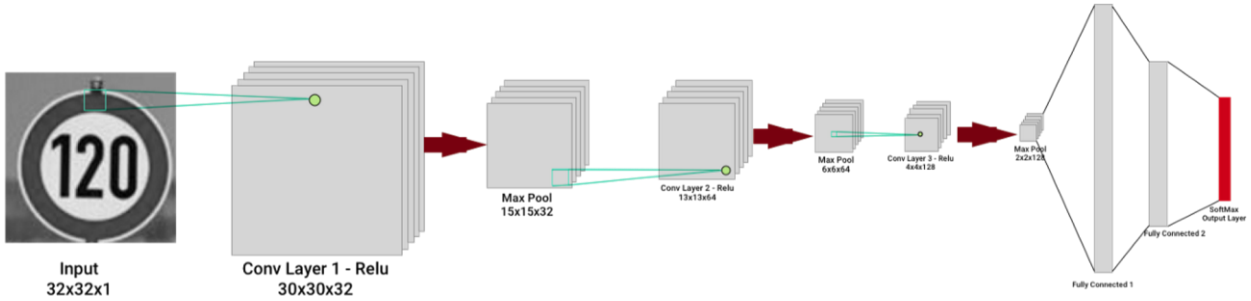


Figure 48: EdLeNet 3x3 Architecture

The network is consists of 3 convolutional layers—kernel size is 3x3, with depth doubling at next layer-using ReLU as the activation function, each followed by a 2x2 max pooling operation. The last 3 layers are fully connected, with the final layer producing 43 results (the total number of possible labels) computed using the SoftMax activation function. The network is trained using mini-batch stochastic gradient descent with the Adam optimizer.

In this case, how deep learning can be used to categorize traffic signs with high accuracy, employing a diversity of pre-processing and regularization techniques, and trying different model architectures. We built vastly configurable code and developed a bendable way of evaluating multiple architectures. This model reached close to close to 98% accuracy on the test set, achieving 99% on the validation set.

3.2.9. Mobile app and void command

Circuit

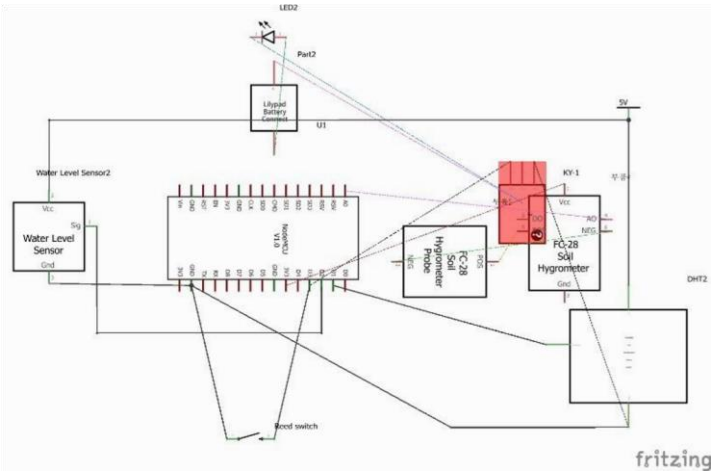


Figure 49: Circuit of the app

3.2.9.1 Mobile app

Blynk is a Platform with IOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet. It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets. Blynk application can be found from the following links

1. Android Blynk App
2. IOS Blynk App

After also need to install the Blynk Arduino Library, which helps generate the firmware running on your ESP8266. Blynk works with hundreds of hardware models and connection types. Select the Hardware type. After this, select connection type. In this project we have select Wi-Fi connectivity. The Auth Token is very important – you'll need to stick it into your ESP8266's firmware. For now, copy it down or use the "E-mail" button to send it. Finally app can upload to the Google play store and people can download to their mobiles easily

Figure 50: Blynk app dashboard

.

3.2.9.2 Void command

Google assistant is AI (Artificial Intelligence) based voice command service. Using voice, we can interact with Google assistant and it can search on the internet, schedule events, set alarms, control appliances, etc. This service is available on smartphones and Google car. So it can control autonomous car devices including lights, AC, any system using our Google Assistant.

If This Then That, also known as IFTTT is a free web-based service to create chains of simple conditional statements, called applets. An applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, or Pinterest.

Figure 51: Voice command components and websites

3.2.10. Programming

NVidia jetson Nano was decided to be used as the high level controller which will handle the servo motors and DC motors of the car. Python programming language through the jetson Nano will be used for that purpose. NVidia jetson Nano connect to Wi-Fi router use by Ethernet cable. Furthermore Wi-Fi module can be used accordingly can connect Wi-Fi, microcontroller and laptop without Ethernet cable. Also Ardiuno mega use to control traffic light.

Raspberry Pi will process of Face recognize. Image processing was meant to be implemented as a secondary objective if time permitted.

Table 2: Programming details

Controller	**Language**	**Devices controlled**	**Purpose**
NVideia Jetson Nano	Python	• DC motor • controller and DC motors • Servo motor controller and Servo motor • Camera module	• Motion controlling of Car • Control steering angle • Take images and record data • Real time training. • Communicating between laptop and microcontroller
Raspberry Pi version 3	Python	Pi Camera	Image processing(optional) Face recognize
Arduino Mega	C (Arduino and Processing IDE)	LED lights	Traffic Light control
Node MCU	Arduino	Sensors	App development and void assess

3.2.11. Software Tools

With the use of python programming for the programming of the Embedded System and in addition to python, some related programming software required to be installed and they are OpenCV and Scikit Learn and Tensor Flow. At the starting of the project for the simulation purposes MATLAB Simulink was decided to be used. The usage and details of the additional python tools are mentioned below.

3.2.12. Testing

Toward ensure that the robot works in the desired conditions, a series of tests will be conducted during and after the manufacturing process. Small scale tests are planned to be done on the path. While tests, Images were taken and steering angle date were recoded. So sign boards, traffic lights, vehicles and bridges are in the path. Large scale and final testing is planned to be done in a self-handling vehicle.

Tests:

- Motion Motor testing
- Path drawing and angle measuring testing
- Steering angle testing
- Camera testing
- Vehicle run testing in the path
- Final testing
- App and voice command testing

3.3. Contributions

Throughout the project all three members have contributed efficiently to complete the project.

Table 3: Member's contributions

Group member	Task
Bandara W.A.K.K.B	1. Solid work design Designed 2. Motor calculations and simulation 3. Assembly of car and Make path (Design bends, bridge, sing board etc.) 4. Programming of raspberry pi and jetson Nano 5. Vehicle detection 6. vehicle speed and distance detection 7. Passenger detection 8. Neural network training, calculations and simulation 9. Testing in path 10. Application development and voice command accesses
David .W.J	1. Design (basic) 2. Assembly of car and Make path (Design bends, bridge, sing board etc.) 3. Programming of raspberry pi and jetson Nano 4. Coding and make face recognize 5. Assembly of body 6. Sign board identification 7. Neural network traning 8. Testing in path

M.J.F. Ahamed	1. Solid work design Designed 2. LED light designing 3. Assembly of car and Make path (Design bends, bridge, sing board etc.) 4. PCB design 5. Wiring 6. Testing in path 7. Camera testing and steering angle fixing 8. Neural network training

CHAPTER 4

RESULTS AND DISCUSSION

Results of this project will be broken down into two parts. Development of the Artificial intelligence self-driving car. Therefore, tasks were allocated among the group members in a manner which will simplify the overall project while providing all members with adequate exposure to all aspects of the research

4.1. Result

4.1.1 Result of DNN

The result of the project includes successful implementation of all the components in the system, which includes DNN, computer vision, mechanical component and Local Host Network in achieving the goal of the project. From the findings, it indicates that the performance in the simulator is increased when the application of the computer vision-based lane detection algorithm.

According to the dependency of the DNN the level of precision can be identified as,

Table 4: DNN level Precision

Level of Autonomy	Techniques
CV	55%
DNN	94%
CV + DNN + Control Unit	98%

Training of the DNN was initially performed using the default values of the hyper-parameters of the DNN. This shows us the baseline evaluation progress of the training.

Table 5: Baseline Evaluation

Loss Function	Mean Squared Error (MSE)
Gradient Optimizer Algorithm	Adam
Learning rate	0.001

The learning rate was the only parameter which was adjusted during the training process because the default values for the gradient optimizer algorithm were found to be sufficient for the project.

Different learning rate showed that the initial rate of 0.001 was considered to be best.

4.1.2 Udacity simulator result

The following results were observed for each of the previously described architectures. For a comparison between them, I had to come up with two different performance metrics.

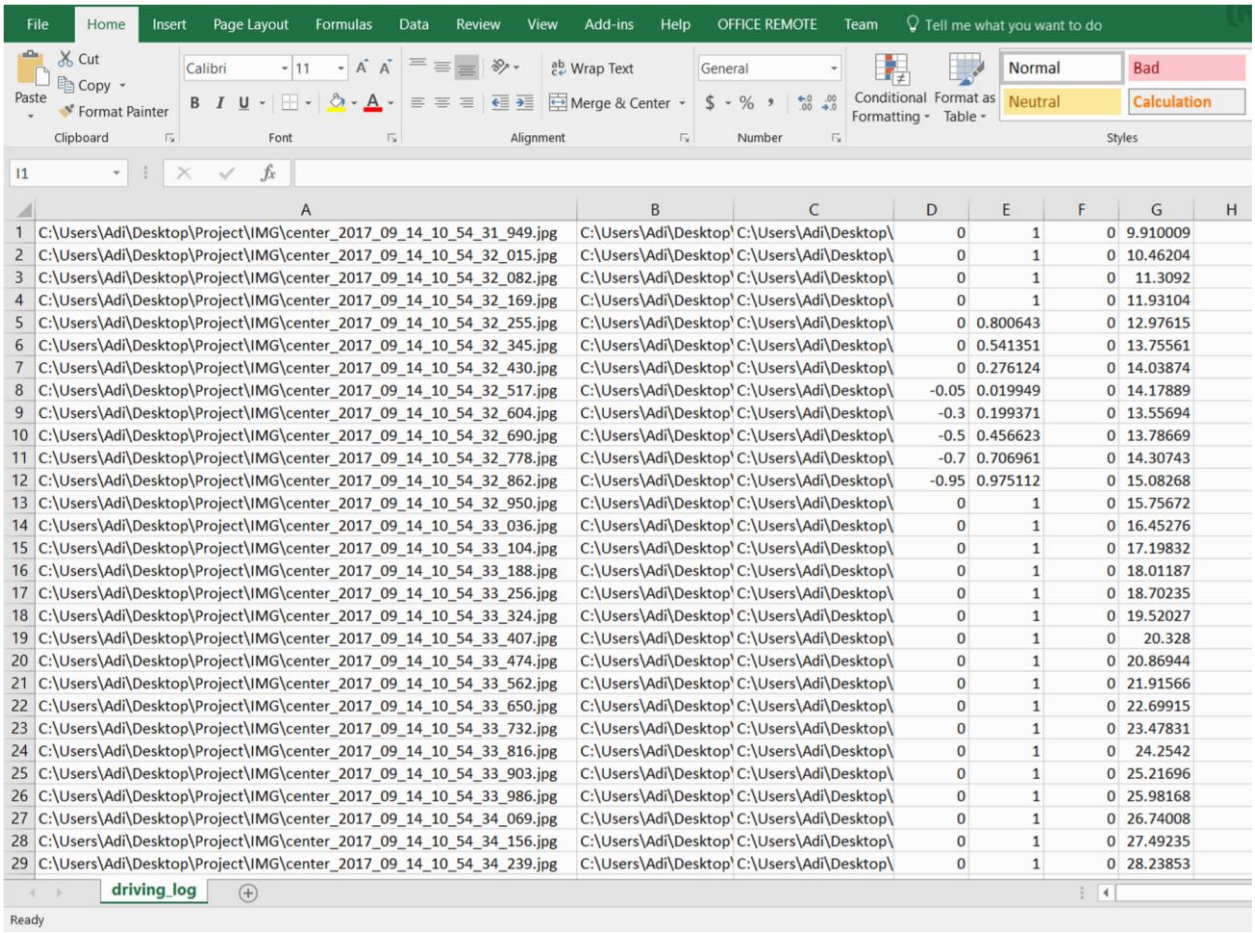
1. Value loss or Accuracy (computed during training phase)
2. Generalization on Track_2 (drive performance)

4.1.2.1 Value loss or Accuracy

The simulator's feature to create your own dataset of images makes it easy to work on the problem. Some reasons why this feature is useful are as follows:

- The simulator has built the driving features in such a way that it simulates that there are three cameras on the car. The three cameras are in the center, right and left on the front of the car, which captures continuously when we record in the training mode.
- The stream of images is captured, and we can set the location on the disk for saving the data after pushing the record button. The image set are labelled in a sophisticated manner with a prefix of center, left, or right indicating from which camera the image has been captured.

- Along with the image dataset, it also generates a datalog.csv file. This file contains the image paths with corresponding steering angle, throttle, brakes, and speed of the car at that instance.



	A	B	C	D	E	F	G
1	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_31_949.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	9.910009
2	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_015.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	10.46204
3	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_082.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	11.3092
4	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_169.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	11.93104
5	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_255.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	0.800643	0	12.97615
6	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_345.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	0.541351	0	13.75561
7	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_430.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	0.276124	0	14.03874
8	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_517.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	-0.05	0.019949	0	14.17889
9	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_604.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	-0.3	0.199371	0	13.55694
10	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_690.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	-0.5	0.456623	0	13.78669
11	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_778.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	-0.7	0.706961	0	14.30743
12	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_862.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	-0.95	0.975112	0	15.08268
13	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_32_950.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	15.75672
14	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_036.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	16.45276
15	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_104.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	17.19832
16	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_188.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	18.01187
17	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_256.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	18.70235
18	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_324.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	19.52027
19	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_407.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	20.328
20	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_474.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	20.86944
21	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_562.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	21.91566
22	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_650.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	22.69915
23	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_732.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	23.47831
24	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_816.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	24.2542
25	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_903.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	25.21696
26	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_33_986.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	25.98168
27	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_34_069.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	26.74008
28	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_34_156.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	27.49235
29	C:\Users\Adi\Desktop\Project\IMG\center_2017_09_14_10_54_34_239.jpg	C:\Users\Adi\Desktop\	C:\Users\Adi\Desktop\	0	1	0	28.23853

Figure 52: Data set of simulator (csv file)

The first evaluation parameter considered here is “Loss” over each epoch of the training run. To calculate value loss over each epoch, Keras provides “val_loss”, which is the average loss after that epoch. The loss observed during the initial epochs at the beginning of training phase is high, but it falls gradually, and that is evident by the screenshots below. Refer to Figure 53 which shows the run of Architecture_1 in the training phase.

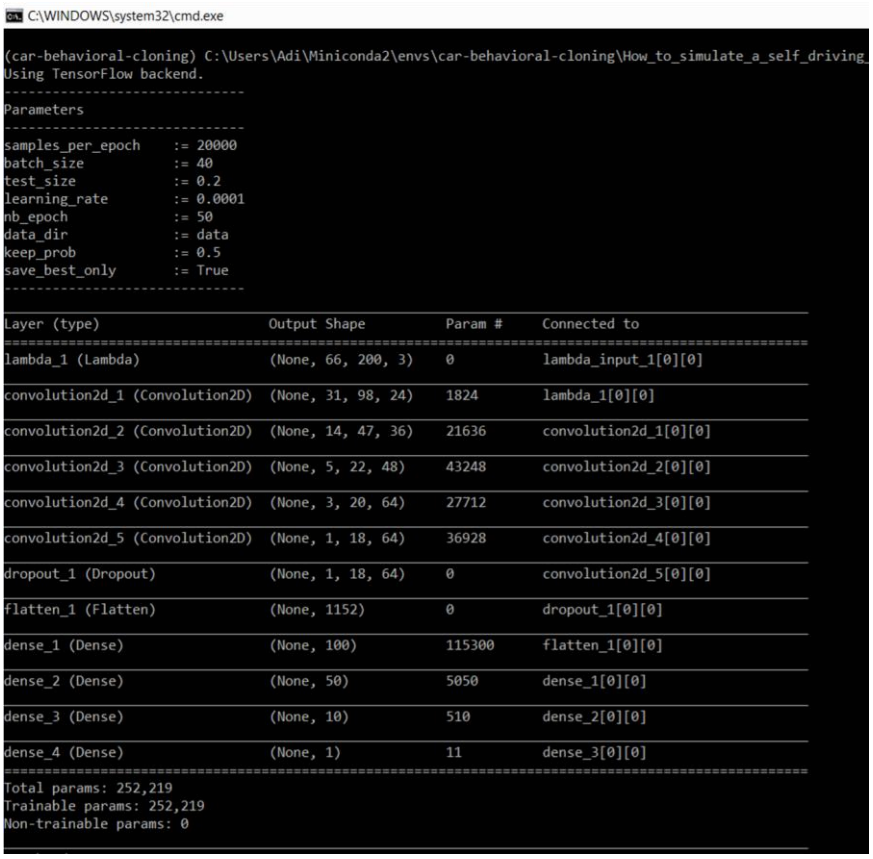


Figure 53: Test 01 CCN training (Architecture_1)

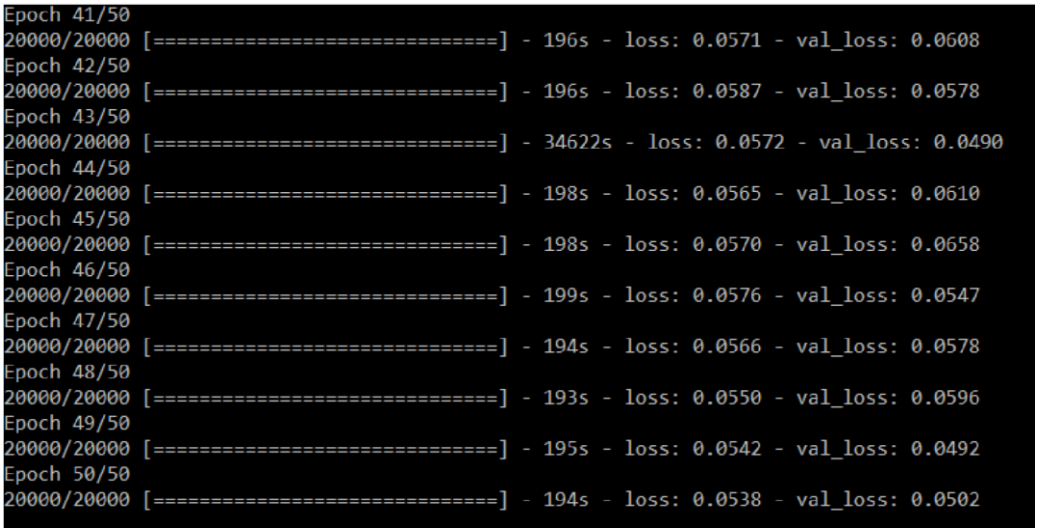


Figure 54: Test 01 CCN training (Architecture_1)

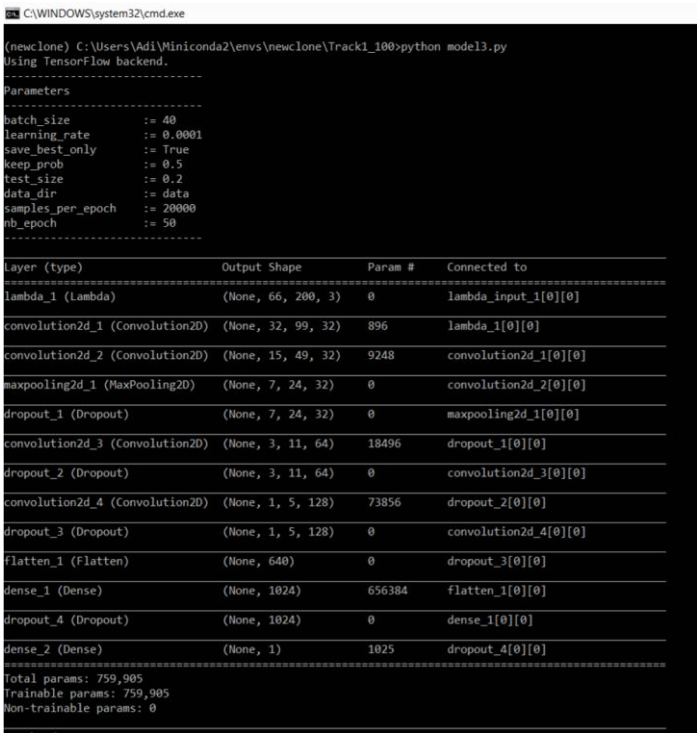


Figure 55: Test 01 CCN training (Architecture_2)

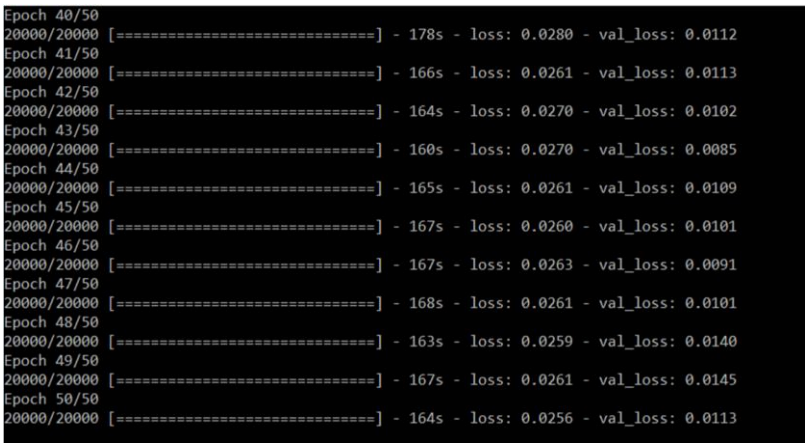
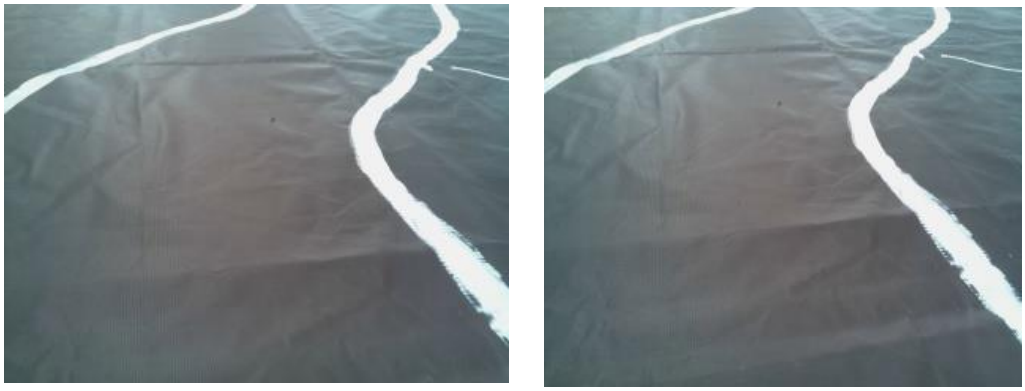


Figure 56: Test 01 CCN training (Architecture_2)

4.1.3 AI car steering results

Step 1: train the car on a path

When trying to improve the system's performance and generalization, one of the best ways is to simply collect more data. In most situations, experimenting with different models and hyper parameters will lead to better results, but these computations are very costly.



:

Figure 57: Real project path and images that captured by camera

Step 2: Saved csv file- there have image name and steering angle

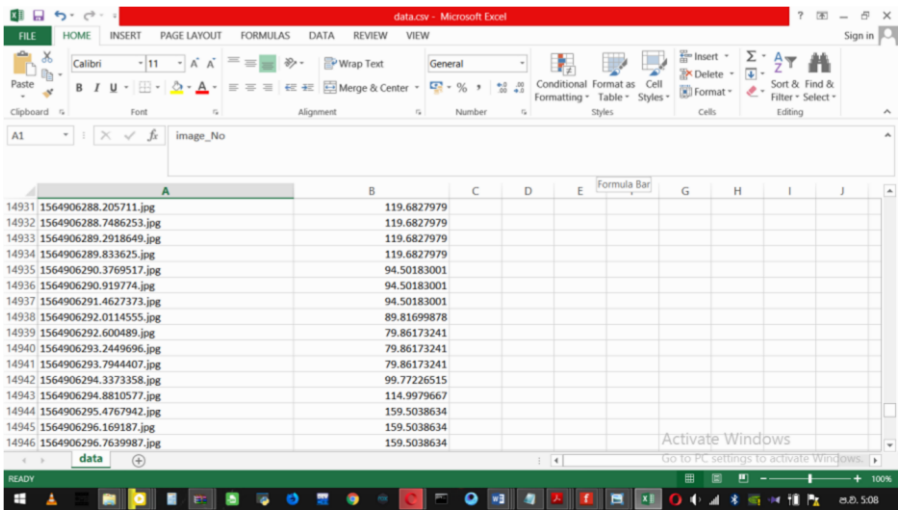


Figure 58: Data set of AI car (csv file)

Step 3: Neural network training

There were various combinations of architectures tried, predicting the steering angle and input for the car to drive in training mode. Neural Network layers were organized in series and various combinations of Time-Distributed Convolution layers, Max Pooling, Flatten, Dropout, Dense and so on are used in architectures. The best performing ones are shown in detail. Refer to the model listings for the parameters used to build them.

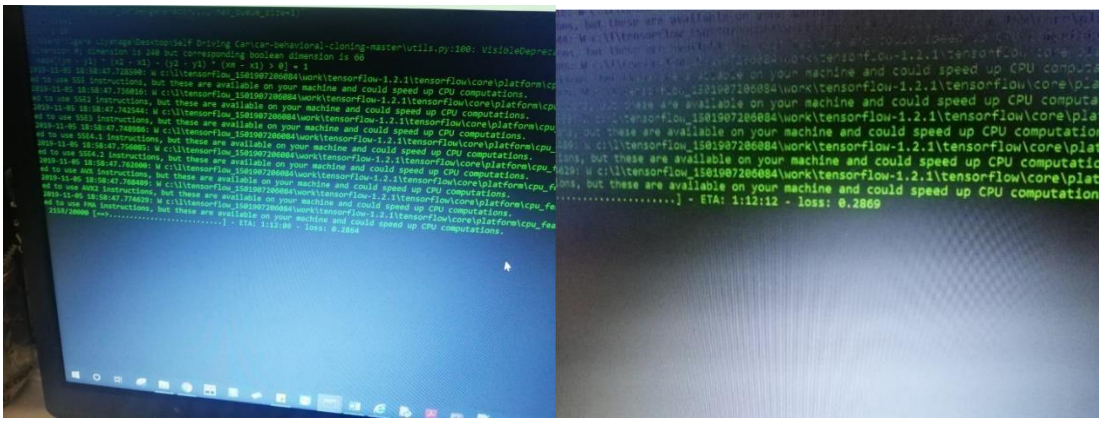


Figure 59: Final neural network training

4.1.4. Vehicle and distance detection using stereo vision

The proposed method for object distance measurement in self-driving cars designed to rely purely on stereo cameras provide good results. This method is able to aid in measuring the distance between the cars and the objects to determine the safe driving distance.

The method developed in this work uses only two cameras to capture the front obstacles because the purpose of using stereo vision is to replace the existing active method and perception systems. For future work, the same method can be implemented using more web cameras or 360° cameras to detect obstacles in more directions.

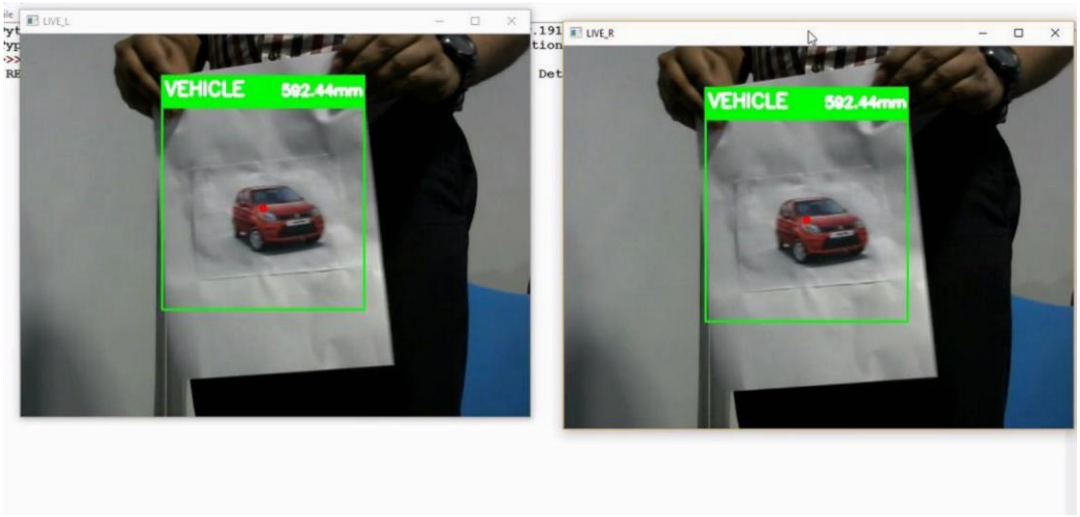


Figure 60: Stereo vision

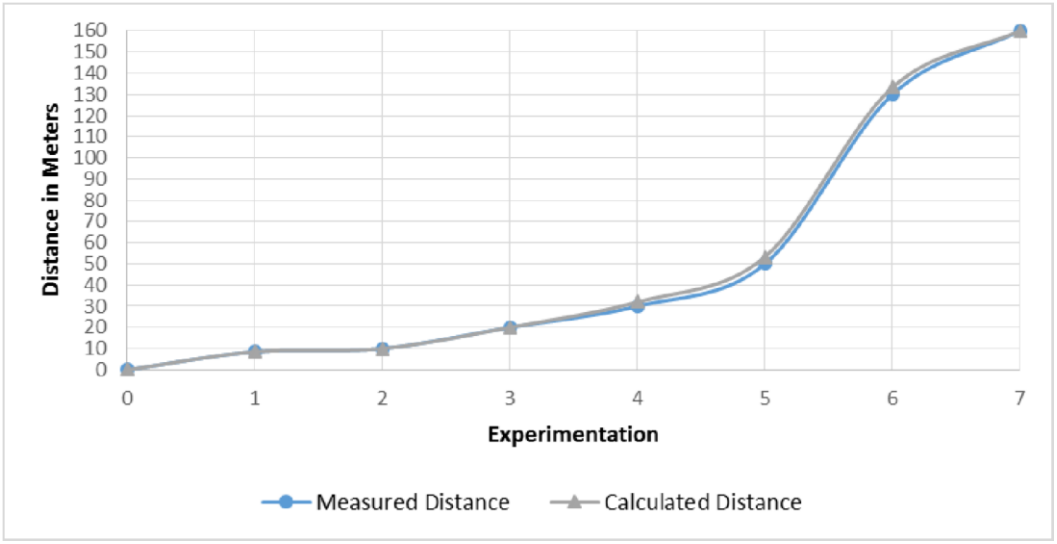


Figure 61: Measured distance verity of calculated distance

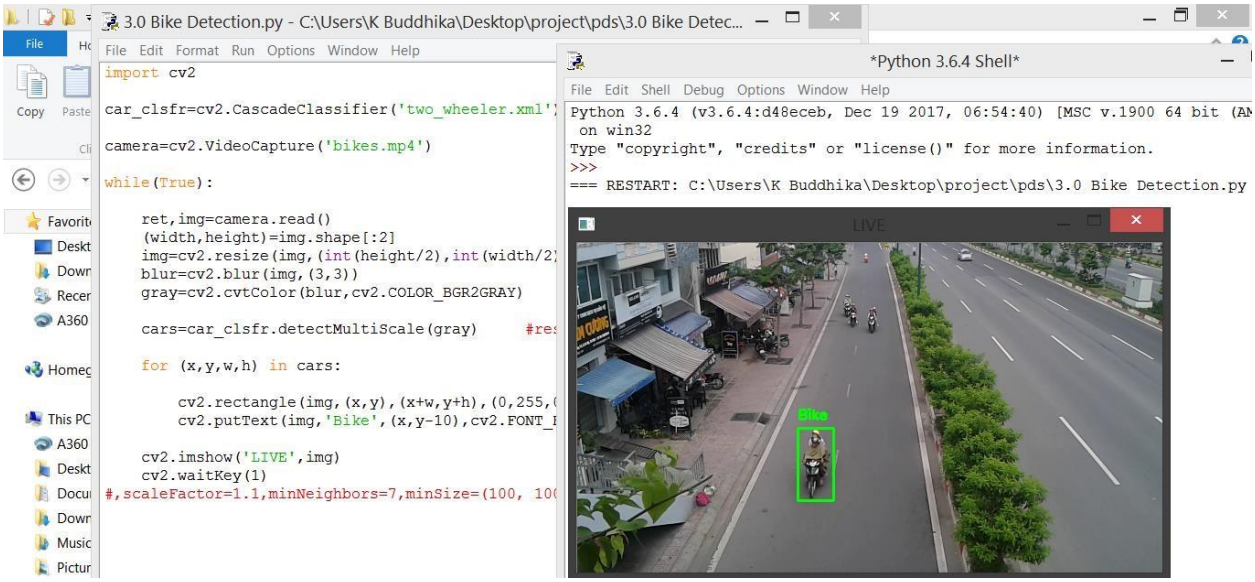


Figure 62: Bick detection

Figure 63: Vehicles detection

4.1.5 Pedestrian detection

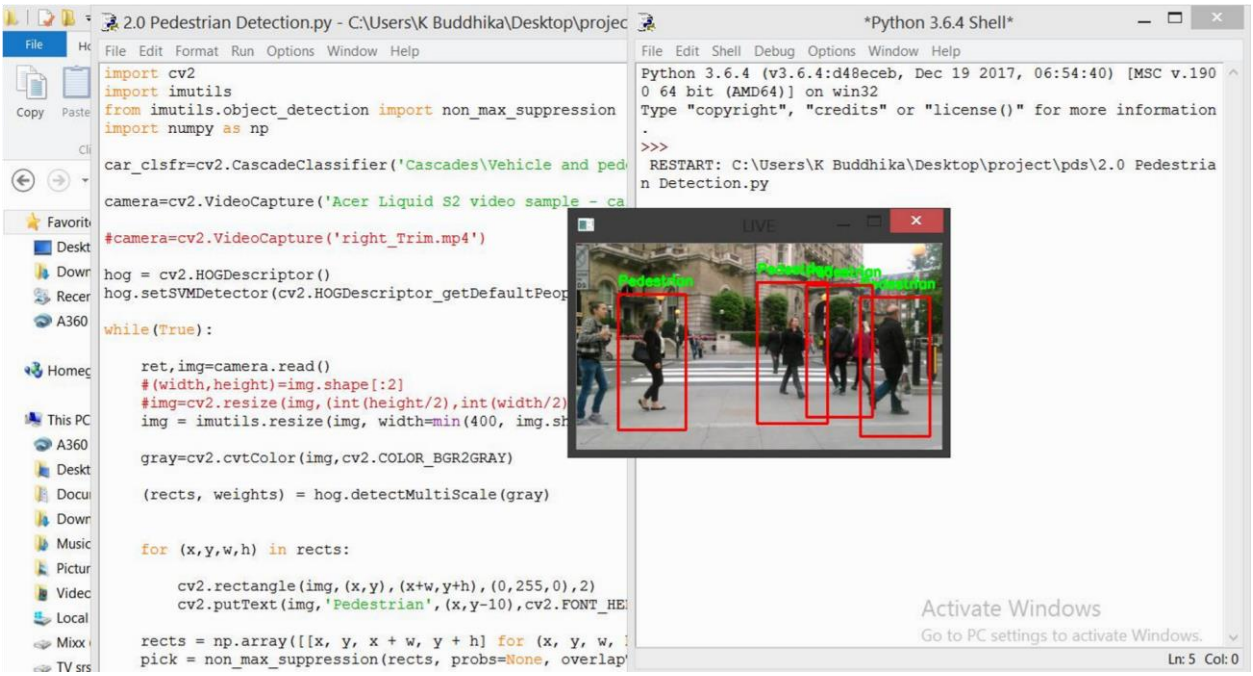


Figure 64: Pedestrian detection

4.1.6 Sign board detection

Step 1: Data sample results

Figure 65: Data set example of sign boards

Step 2: Neural network training

Figure 66: Neural network training of sing boards

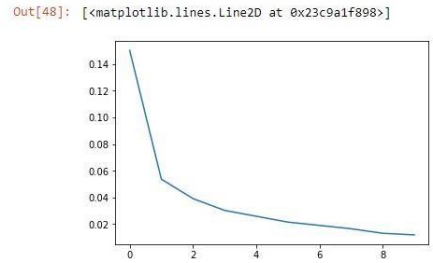


Figure 67: Loss of CNN

4.1.7. Mobile app and voice command

Results: App

- Humidity and temperature of vehicle
- Send the message about door open or close
- Send the message about water level in the water tank

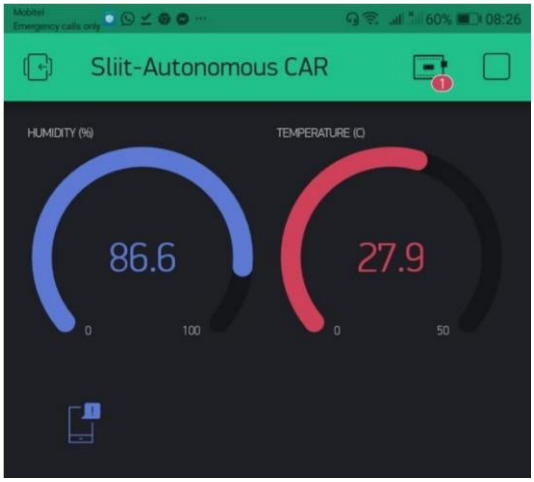


Figure 68: Mobile App

Result: Voice command

An applet may send an e-mail message if the user tweets using a hashtag or copy a photo on Facebook to a user's archive if someone tags a user in a photo. Here, IFTTT used to use Google assistant service and Adafruit service in chain. So, when Google assistant use to control light of car by saying Ok Google, turn the light ON or OFF. Then IFTTT interpret the message and can send it to Adafruit's dashboard as a understandable command to the created feed. So, when I use Google Assistant on my mobile and give voice command as "Ok Google, Turn Lights ON", applet created in IFTTT receive this command and will send data '1' to the Adafruit feed. This will trigger the event on Adafruit dashboard which is continuously monitored by the microcontroller (here NodeMCU). This microcontroller will take action as per the data change on the Adafruit dashboard.

- Control AC
- Control vehicle lights

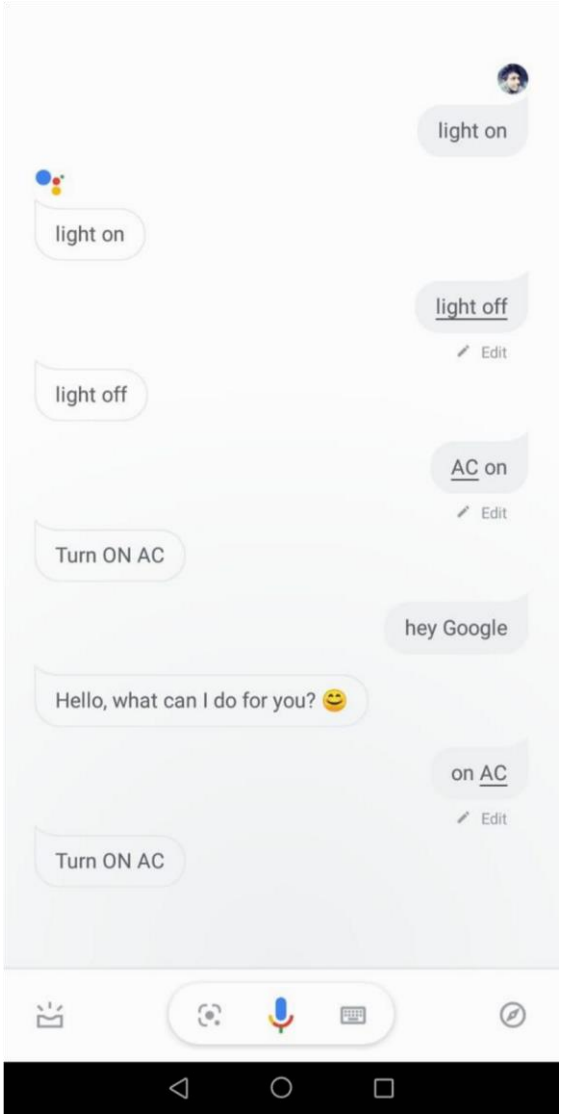


Figure 69: Assign the Google Assistance

4.2 Discussion

Camera Limitation

The Raspberry Pi camera which we were using was v1.3 which was supported with Raspberry pi but when the board was changed to Jetson Nano the was able to work with pi camera above V2.0 due to this we were required to find a new camera for the project which supports to Jetson. The cost for the camera was very high depending on our budget. But finally web camera were used and it was working really well.

Hardware Limitations

Due to the technology improvement the usage of the deep learning has improved vastly and depending on that the project was occupied with both hardware and software implementations. For the process without GPU, the real-time process of the training data will be possible but running the DNN in real-time will be difficult. So, mostly the running and testing procedure of the DNN is mostly GPU depended due to the reason the training was done through google Cloud using COLABS. Here only one camera was used to obtain images and real-time run when using stereo vision technology to figure out the distance of the vehicle but for this purpose minimum two cameras are required and one was attached to the microcontroller due to one pin attachment was available for camera. Also, it is required to obtain images from sideways as well so there it is required to attach cameras to the side as well but when attaching all the equipment to the RC car the load would be get increased and it would be not practical and all the calculations must be required from the beginning. There is also a requirement for sensors values which is required to be obtained along with the images where it can only be attained when implementing to a real vehicle. The Jetson developer doesn't have a Wi-Fi module so it is required to be installed additionally and most of the modules does not support with it.

Software Dependencies

The main goal of the project was to implement neural network rather than creating them from a scratch. For this a lot of examination and research was done due to obtain many frameworks as possible with the advantages received through different libraries such as TFLearn, CUDA and etc. This process helped us to reduce the time requirement for development and implementation but while implementing some outcomes required a vast area of dependencies.

Many of the dependencies were installed through some research but some areas were difficult in achieving and some was hard and some took most of the time from our project. Due to the change of the microcontroller we faced lot of hardware and software issues basically some of the libraries does not support the platform and installing some important libraries required more time to installed because the board was new in market of Sri Lanka and most of them have not experience with it. So most probably the change of the board cost us to implement the requirements from the beginning which caused us a lot of time.

Simulator

An opportunity to evaluate the robustness of the system is to create another track other than the predefined jungle and Lake Track and see if the model trained would be able to drive the car on the track. Also, if it could drive relatively well, it would be interesting to observe the difference in how well it drives on the trained track versus the track that the model hadn't been trained on. During the project period it was decided to use a simulator to obtain the steering angle to not be depended on the hardware.

CHAPTER 05

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In this thesis described how to use deep convolutional networks to control a prototype of a simplified self-driving RC car. It built the vehicle from scratch using custom hardware parts. Then multithreaded system were designed a that controls it and provided a light web and command line interface which users can use to interact with the car. Lastly, the car was equipped with autopilot controller based on state-of-the-art convolutional neural network architecture. The first chapter focused on the current state of autonomous driving and approaches that are used in the industry. Chapters two and three described how to actually create such car. What are the hardware requirements and how to write software that will be able to control it? In the next chapter, the Controller, various neural network were trained, architectures that could perform well on the task of autonomous driving and could measure their performance and chose the best for additional improvements. The last part of the thesis consisted of several experiments aimed to improve and validate the car's autonomy. Enriching the training dataset and using proper forms of regularization turned out to yield the best results.

Our main goal was to run the vehicle in the path which were created for the RC car with a proficiency of 98% and obtain a success of more than 90% in the avoidance of collision and detection as well as the self-decisioning. All the requirements were expected to be achieved through only one camera as an input. From this it is expect to demonstrate a method which could be useful in training an autonomous vehicle which is capable of allowing the fast redeployment under new condition and new systems.

From our results it can be said that the DNN was the best in contributing to obtain the running the system in a simulator. When the same system is applied to the real-world scenario such as a miniature model of the car in the environment, where lots of shortcomings from the DNN were identified and highlighted. Demonstrating the DNN in the real world where different types of environments encountered where they are required to be explicitly trained. So due to that the combination of computer vision was used to imply the significance of the different environment with DNN. Considering both together it can be highlighted that to provide a safe

environment the system requires both a vision and a brain like a human driver by achieving through computational techniques of DNN and computer vision techniques.

The system learned to steer autonomously when running on fixed speed. It can drive on simplified flat roads without traffic only using camera as its input. In the end, the car was able to drive indefinitely on a circular loop and a set of three flat tracks that it had never seen before. The biggest challenges had to face were the absence of driving simulator and the problem of over fitting.

5.2 Recommendation

The changes are mentioned in the forthcoming sections which would help to test the system continuously and also to improve the performance of the system.

- To make the system more robust, a much larger dataset of different track types and different road conditions would need to be collected. This could be an interesting use-case for generative adversarial networks [61]. The driving task could also become harder by introducing intersections, signs, traffic or pedestrians. That would require a modular approach with specific parts designed for object detection. Ultrasonic sensors and radar could then be used to gather more complex information about the car's surroundings.
- A natural extension in the research would be retrain the existing neural network original data with offset labels and amplifying the hidden, pooled layers and FC layers with increased number of neurons. Also, providing video transmission towards the latency in the improvement performance were turn is required to be done in different scenarios.
- There is more exploration can be done in the neural network architecture. One of the limiting factors which is required to be considered in the process is the size of the GPU memory. A microcontroller which is provided with a vast amount of onboard memory can be used to implement complex networks with different structures. With the additional resources it would be possible to combine the throttle control, steering control, vehicle & pedestrian detection and Traffic sign & light detection into a single branching neural network. This could save us overhead of running a greater number of neural networks simultaneously.
- To its current state we are trying to input a Virtual Support like Google assistant to improvise the requirement of the passenger. Also, providing a face recognition based

door unlocking system for the car to keep the privacy of the person protected and adding a manual control system when the autonomous system of the car is being hacked. Providing a communication between cars to avoid collisions in roads and increasing the safety of the passengers.

REFERENCE

[1] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images.05 2012.

[2] Krizhevsky, A.; Sutskever, I.; et al. ImageNet Classification with Deep Convolutional Neural Networks.
Commun. ACM, volume 60, no. 6, May 2017: pp. 84–90, ISSN 0001-0782, doi:10.1145/3065386. Available from: http://doi.acm.org/10.1145/3065386

[3] He, K.; Zhang, X.; et al. Deep Residual Learning for Image Recognition. CoRR, volume abs/1512.03385, 2015, 1512.03385. Available from: http://arxiv.org/abs/1512.0338559 Bibliography

[4] Vinyals, O.; Toshev, A.; et al. Show and Tell: A Neural Image Caption Generator. CoRR, volume abs/1411.4555, 2014, 1411.4555. Available from: http://arxiv.org/abs/1411.4555

[5] Fang, H.; Gupta, S.; et al. From Captions to Visual Concepts and Back. CoRR, volume abs/1411.4952, 2014, 1411.4952. Available from: http://arxiv.org/abs/1411.4952

[6] Redmon, J.; Divvala, S. K.; et al. You Only Look Once: Unified, Real-Time Object Detection. CoRR, volume abs/1506.02640, 2015, 1506.02640. Available from: http://arxiv.org/abs/1506.02640

[7] Badrinarayanan, V.; Kendall, A.; et al. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. CoRR, volume abs/1511.00561, 2015, 1511.00561. Available from: http://arxiv.org/ abs/1511.00561

[8] Long, J.; Shelhamer, E.; et al. Fully Convolutional Networks for Semantic Segmentation. CoRR, volume abs/1411.4038, 2014, 1411.4038. Available from: http://arxiv.org/abs/1411.4038

[9] S. S. Ramachandran and A. K. Veeraghavan, "Development of flexible Autonomous Car System Using
Machine Learning and black chain," Development of flexible Autonomous Car System Using Machine Learning and black chain, Dec. 2018.

[10] W. Y. Lin, W. H. Hsu, and Y. Y. Chiang, "A Combination of Feedback Control and Vision-Based Deep
Learning Mechanism for Guiding Self-Driving Cars," A Combination of Feedback Control and VisionBased Deep Learning Mechanism for Guiding Self-Driving Cars, pp. 262–266, 2018.

[11] San Diego, "Looking at Humans in the Age of Self-Driving and Highly Automated Vehicles," Looking at
Humans in the Age of Self-Driving and Highly Automated Vehicles, vol. 1, no. NO, pp. 90–104, Mar. 2016.

[12] Q. Memon, M. Ahmed, S. Ali, A. R. Memon, and W. Shah, "Self-driving and driver relaxing vehicle," 2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI), 2016.

[13] R. B. Sulaiman, "Artificial Intelligence Based Autonomous Car," SSRN Electronic Journal, 2018.

[14] J. Kim, G. Lim, Y. Kim, B. Kim, and C. Bae, "Deep Learning Algorithm using Virtual Environment Data for Self-driving Car," 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2019.

[15] V. Porto and D. Fogel, "Neural network techniques for navigation of AUVs," Symposium on Autonomous Underwater Vehicle Technology, pp. 305–313.

[16] Farooq, U., Amar, M., Hasan, K., Akhtar, M., Asad, M. and Iqbal, A. (2019). A low cost microcontroller implementation of neural network based hurdle avoidance controller for a car-like robot - IEEE Conference Publication. [Accessed 8 Apr. 2019].

[17] Min Yan Naing, A. and Thinda Than, I. (2019). Experiment on Real-Time Image Processing in the Controlling of Mecanum Wheel Robotic Car - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/8646199 [Accessed 9 Apr. 2019].

[18] Duan, J. and Viktor, M. (2019). Real time road edges detection and road signs recognition - IEEE Conference Publication. [Accessed 4 Apr. 2019].

[19] Publications.lib.chalmers.se. (2019). [online] Available at: http://publications.lib.chalmers.se/records/ fulltext/251868/251868.pdf [Accessed 9 Apr. 2019].

[20] Anon, (2019). [online] Available at: https://www.researchgate.net/publication/328455196_Working_ model_of_Selfdriving_car_using_Convolutional_Neural_Network_Raspberry_Pi_and_Arduino [Accessed 9 Apr. 2019].

[21] Pdfs.semanticscholar.org. (2019). [online] Available at: https://pdfs.semanticscholar.org/21a7 /c524319650b3e85c7ffaaae7d9d584fe0e50.pdf [Accessed 10 Apr. 2019].

[22] Anon, (2019). [online] Available at: https://www.researchgate.net/publication/328455196_Working _model_of_Selfdriving_car_using_Convolutional_Neural_Network_Raspberry_Pi_and_Arduino [Accessed 9 Apr. 2019].

[23] Farooq, U., Amar, M., Hasan, K., Akhtar, M., Asad, M. and Iqbal, A. (2019). A low cost microcontroller implementation of neural network based hurdle avoidance controller for a car-like robot - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5451340 [Accessed 8 Apr. 2019].

[24] "Raspbian Stretch: Install OpenCV 3 Python on your Raspberry Pi," PyImageSearch, 25-Jul-2018. [Online]. Available: https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3pythonon-your-raspberrypi/. [Accessed: 12-Apr-2019].

[25] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition una ected by shift in position. Biological Cybernetics, volume 36, no. 4, Apr 1980: pp. 193–202, ISSN 1432-0770, doi:10.1007/BF00344251. Available from: https://doi.org/10.1007/BF00344251

[26] Yamins, D. L. K.; Hong, H.; et al. Performance-optimized hierarchical models predict neural responses in higher visual cortex. Proceedings of the National Academy of Sciences of the United States of America, volume 111, no. 23, 06 2014: pp. 8619–8624, doi:10.1073/pnas.1403112111. Available from: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4060707/ Bibliography

[27] Russakovsky, O.; Deng, J.; et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), volume 115, no. 3, 2015: pp. 211–252, doi:10.1007 /s11263-015-0816y.

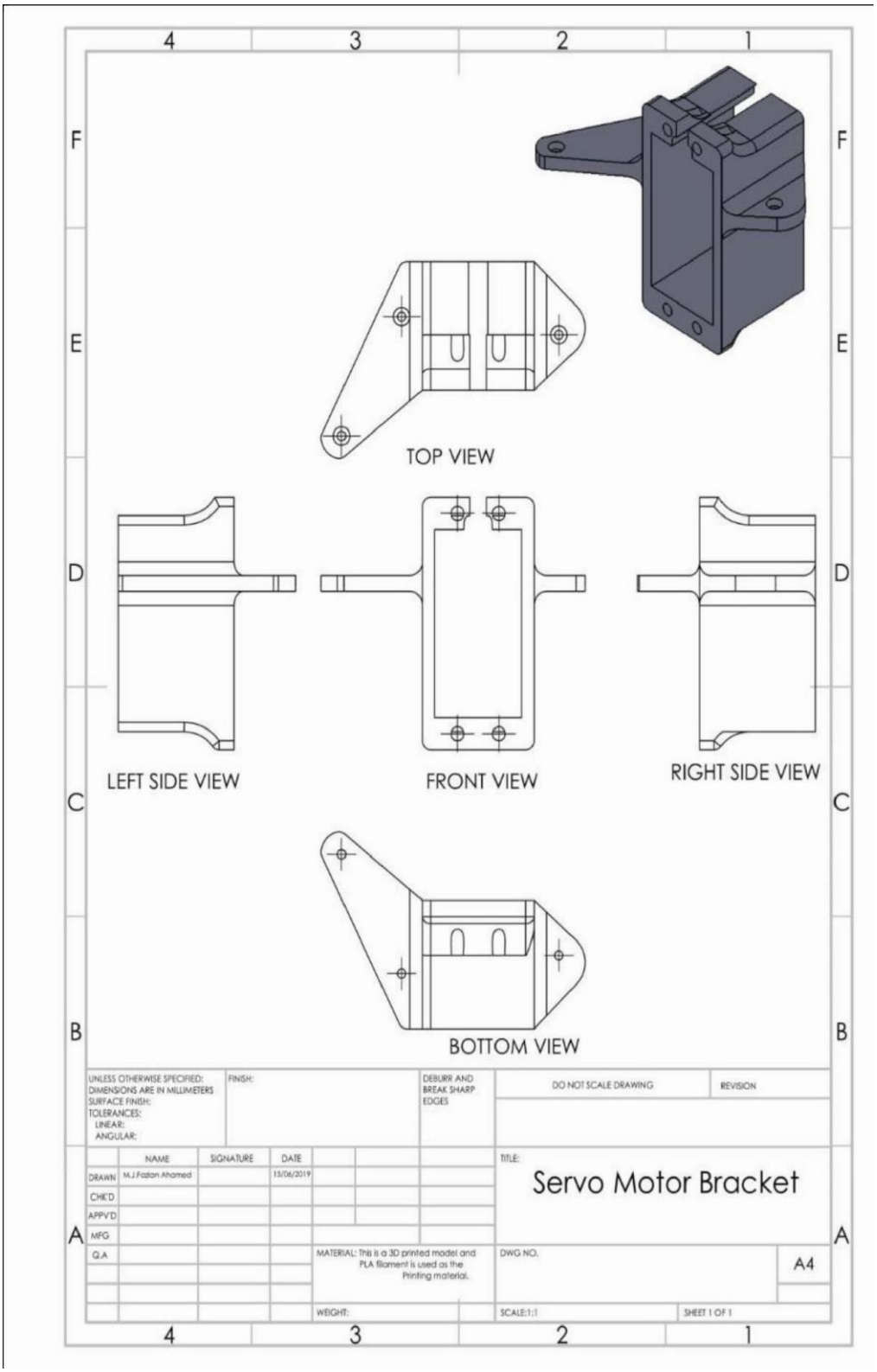
[28]]Mishkin, D.; Sergievskiy, N.; et al. Systematic Evaluation of Convolution Neural Network Advances on the ImageNet. 05 2017.

[29] Krizhevsky, A.; Sutskever, I.; et al. ImageNet Classification with Deep Convolutional Neural Networks.
Commun. ACM, volume 60, no. 6, May 2017: pp. 84–90, ISSN 0001-0782, doi:10.1145/3065386. Available from: http://doi.acm.org/10.1145/3065386

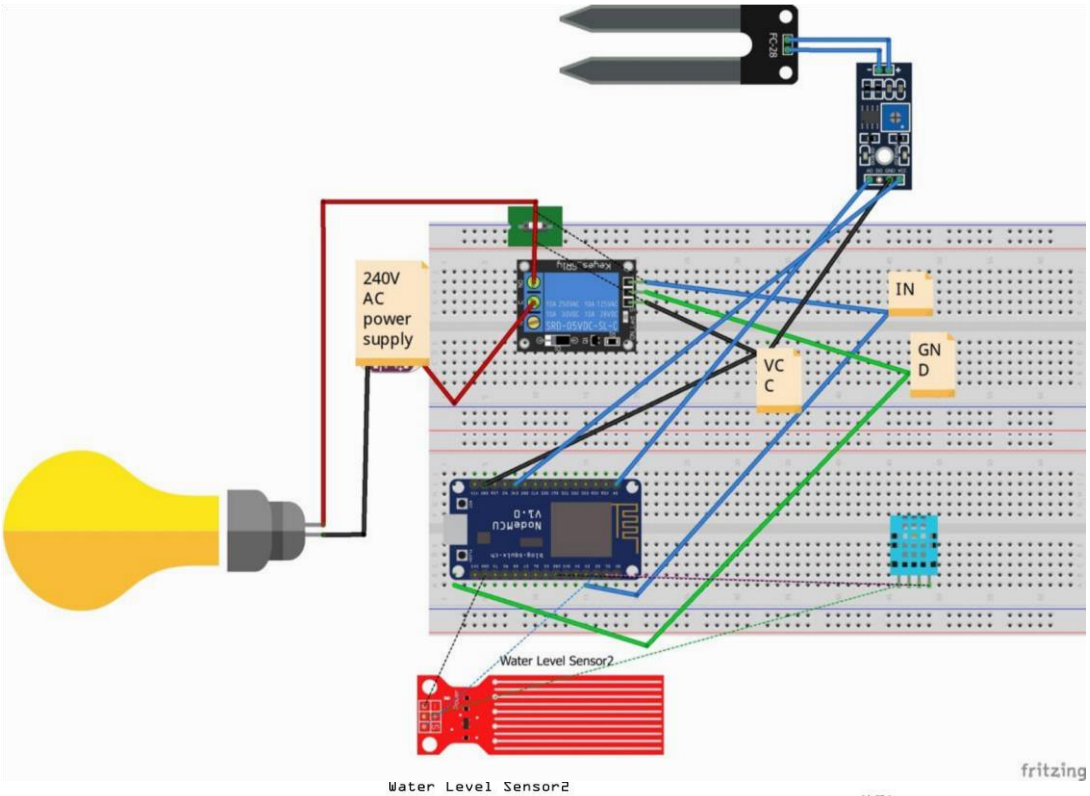
[30] [32]An introduction to neural networks: Part 1 https://codeburst.io/an-introduction-to-neural-networkspart-1-47ba9cfcdd88

APPENDIX A

Design

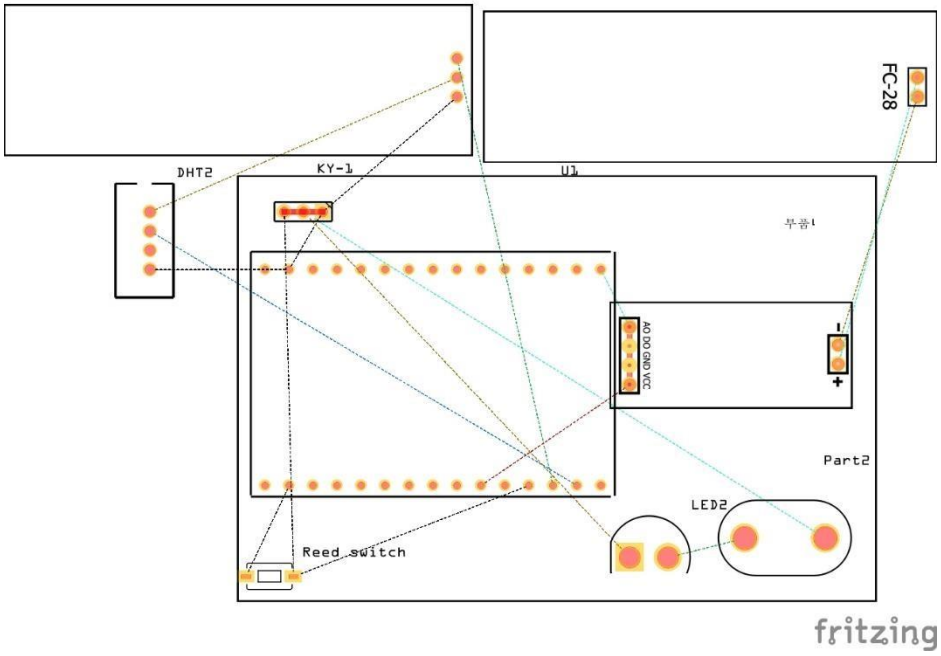


Circuit design



Water Level Sensor2

부품1



APPENDIX B

Codes