A Mini Project Report on

Implementation of Autonomous Vehicles using Deep Learning

Submitted to the Department of Computer Science & Engineering, GNITS in the partial fulfillment of the academic requirement for the award of B.Tech (CSE) under JNTUH, Hyderabad

By

G. Chaitanya Deepti	**19251A05E3**
M. Harshitha	**19251A05G0**
K. Shalini	**20255A0515**
B. Baby	**20255A0518**

Under the guidance of
Mrs. K. Gnana Prasuna
Assistant Professor

Department of Computer Science & Engineering
G. Narayanamma Institute of Technology & Science
(For Women)
(Autonomous)
Approved by AICTE, New Delhi & Affiliated to JNTU, Hyderabad
Accredited by NBA & NAAC, an ISO 9001:2015 certified Institution
Shaikpet, Hyderabd-500104

August 2022

Department of Computer Science & Engineering
G. Narayanamma Institute of Technology & Science
(For Women)

(Autonomous)
Approved by AICTE, New Delhi & Affiliated to JNTU, Hyderabad
Accredited by NBA & NAAC, an ISO 9001:2015 certified Institution
Shaikpet, Hyderabd-500104

Certificate

This is to certify that the Mini Project report on " Implementation of Autonomous Vehicles using Deep Learning" is a bonafide work carried out by G. Chaitanya Deepti (19251A05E3), M. Harshitha (19251A05G0), K. Shalini (20255A0515), B.Baby (20255A0518) in the partial fulfillment for the award of B.Tech degree in Computer Science & Engineering, G. Narayanamma Institute of Technology & Science, Shaikpet, Hyderabad, affiliated to Jawaharlal Nehru Technological University, Hyderabad under our guidance and supervision.

The results embodied in the Mini project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide
Mrs. Gnana Prasuna
Assistant Professor

Head of the Department
Dr. M. Seetha
Professor and Head, CSE

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr K Ramesh Reddy**, **Principal** GNITS, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. M Seetha, Head and Professor**, Dept. of CSE, GNITS for all the timely support and valuable suggestions during the period of our project.

We would like to extend our sincere thanks and gratitude to **Dr. A. Sharada, Professor,** overall Mini Project coordinator, Dept. of CSE, GNITS for all the valuable suggestions and guidance during the period of our project.

We are extremely thankful to **Dr. Raghavender K.V, Assoc. Prof., Mrs. D.R Nandadevi, Asst.Prof.,**Dept. of CSE, GNITS for their encouragement and support throughout the project.

We are extremely thankful and indebted to our internal guide, **Mrs. Gnana Prasuna, Assistant Professor of Gnits , Department of CSE,** GNITS for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank all the faculty and staff of CSE Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

G. Chaitanya Deepti	**19251A05E3**
M. Harshitha	**19251A05G0**
K. Shalini	**20255A0515**
B. Baby	**20255A0518**

ABSTRACT

The rapid development of Artificial Intelligence has revolutionized the area of autonomous vehicles by incorporating complex models and algorithms. Self-driving cars are always one of the biggest inventions in computer science and robotic intelligence. Highly robust algorithms that facilitate the functioning of these vehicles will reduce many problems associated with driving such as the drunken driver problem.

In this paper, the author's aim is to build a Deep Learning model that can drive the car autonomously which can adapt well to the real-time tracks and does not require any manual feature extraction. This research work proposes a computer vision model that learns from video data. It involves image processing, image augmentation, behavioral cloning and convolutional neural network model. The neural network architecture is used to detect path in a video segment, linings of roads, locations of obstacles, and behavioral cloning is used for the model to learn from human actions in the video.

Keywords:

- Self-driving cars
- Deep Learning
- Computer Vision
- Behavioral Cloning
- Image Augmentation

Domain: Deep Learning

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

A self-driving car, also known as an autonomous car, driver-less car, or robotic car , is a car containing vehicular automation, that is, a terrestrial vehicle that is capable of sensing its environment and moving safely with little or no human input. We live in an exciting period where each day more and more technology developments are under process and automation is one of the blessings of it. Automation is a way or method to deliver the output with minimal intervention from humans and in some case, there is no human intervention. These automations provide an accuracy level which is quite difficult to achieve manually and are highly beneficial in fields such as automotive industry, Aerospace, and military where the possibility to have an error in calculation or output is negligible .With the rapid developments in technology, computers are leaving behind the old notion of 'Computers are dumb machines.' Among all the achievements obtained in various fields of computer science, an autonomous vehicle has been one of the biggest and most important invention. It has been a topic of research for many years and numerous algorithms involving advanced concepts of artificial intelligence. Popularity of these cars is increasing tremendously. Recent studies show that around 10 Million self-driving cars would be used on road in the near future. Companies such as Audi, Volvo, Ford, Google, General Motors, BMW, and Tesla incorporated this technology in their latest releases.

Figure 1.1: Waymo's autonomous cars

1.1 Background

In this section, key concepts that are used in the implementation of this project and the motivation behind using these concepts are described. Convolutional Neural Networks (CNN) CNN is a type of feed-forward neural network computing system that can be used to learn from input data. Learning is accomplished by determining a set of weights or filter values that allow the network to model the behavior according to the training data. The desired output and the output generated by CNN initialized with random weights will be different.

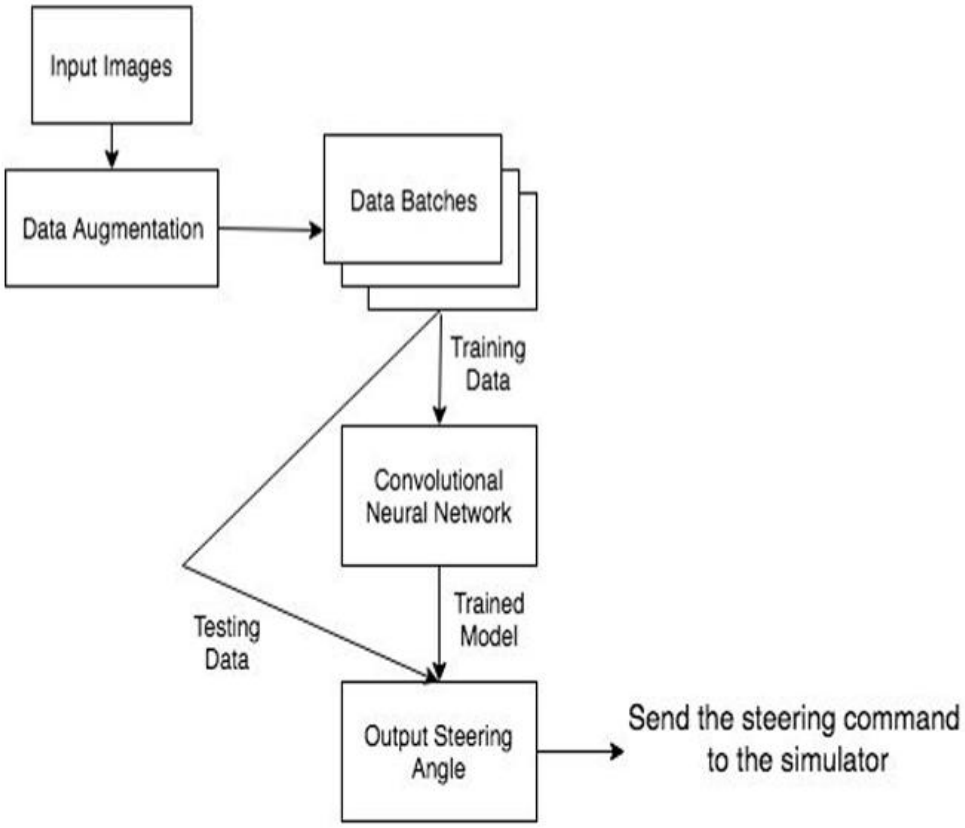


Figure 1.2: High Level Architecture

This difference (generated error) is backpropagated through the layers of CNN to adjust the weights of the neurons, which in turn reduces the error and allows us produce output closer to the desired one . CNN is good at capturing hierarchical and spatial data from images. It utilizes filters that look at regions of an input image with a defined window size and map it to some output. It then slides the window by some defined stride to other regions, covering the whole image.

Each convolution filter layer thus captures the properties of this input image hierarchically in a series of subsequent layers, capturing the details like lines in image, then shapes, then whole objects in later layers. CNN can be a good fit to feed the images of a dataset and classify them into their respective classes. Recurrent Neural Networks (RNN) RNN are a class of artificial neural

networks where connections between units form a directed cycle. Recurrent networks, unlike feedforward networks, have the feedback loop connected to their past decisions.

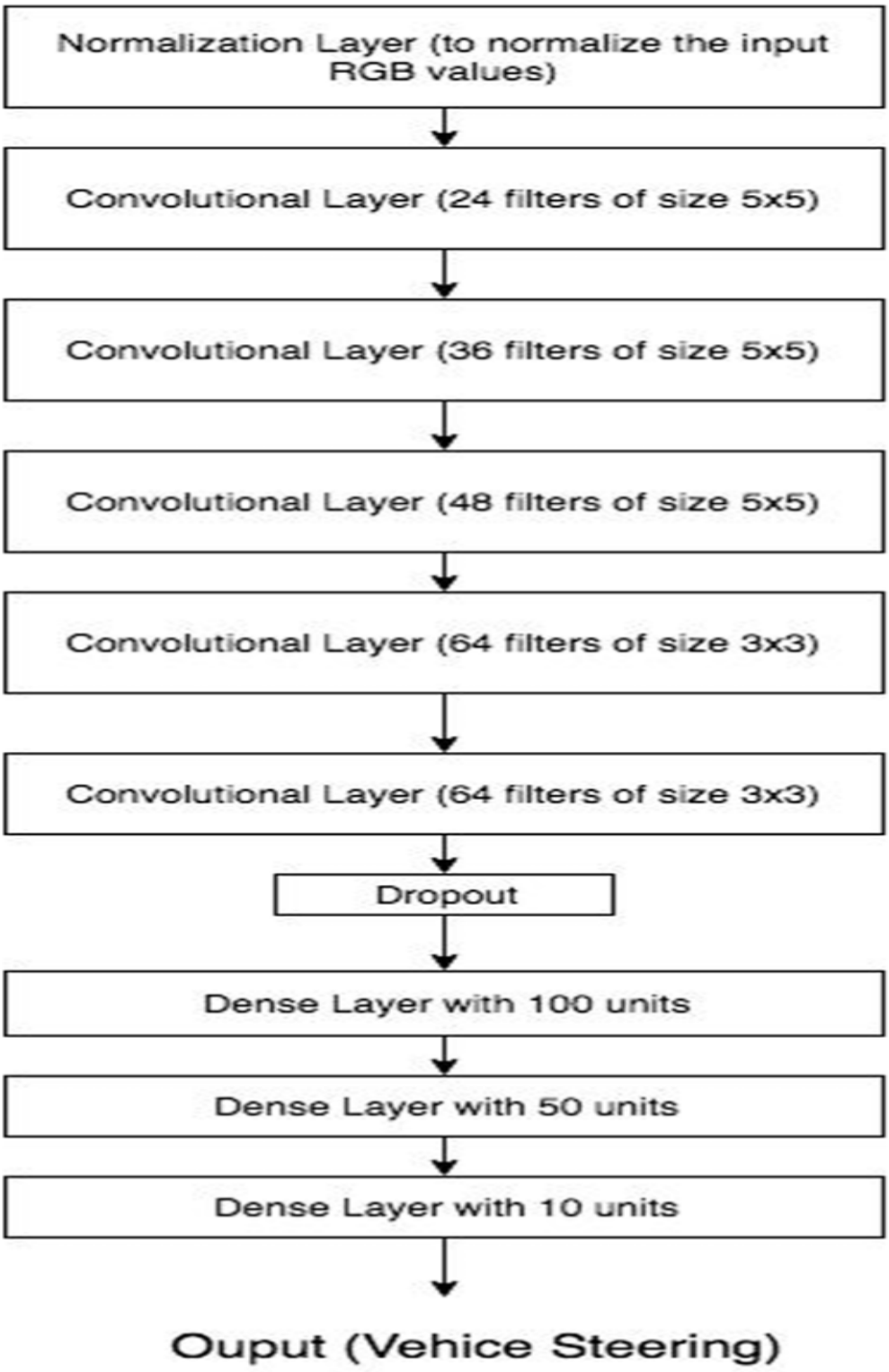


Figure 1.3: Deep Network Architecture

This memory (feedback) helps to learn sequences and predict subsequent values, thus being able to solve dependencies over time. For example, consider the case when the next word in a sentence is dependent on a previously occurring word or context. RNN will be an excellent choice for such scenarios. They are designed to recognize patterns in sequences of data, such as text, handwriting and so on. They are also applicable to images that can be separated (decomposed) into a sequence of patches [2].

Neural networks have activation functions to take care of the non-linearity and to squash the gradients or weights in certain range. Some of these functions are sigmoid, tanh, RELU and so on that are the building blocks of RNN. Though these are very powerful, there are some shortcomings in conventional RNN, such as the well-known problem of vanishing or exploding gradients [2]. Also, training might take a very long time. To overcome this, new classes of RNN implementations have been developed recently.

The deep learning model that has been prepared is based on the research done by NVIDIA for their autonomous vehicle .The model comprises of the following important layers.

1.2 Existing System

Self-driving cars combine a diverse variety of sensors to perceive their surroundings, such as thermographic cameras, radar, lidar, sonar, and GPS. Nowadays, many autonomous vehicles are found in busy towns such as Columbus and Ohio. However, these involve a supervising driver to assist the agent. According to level of human interference required, self-driving cars are categorized into 6 levels by the Society of Automotive Engineers .

These are:

Level 0:

Most vehicles on the road today are Level 0: manually controlled. The human provides the "dynamic driving task" although there may be systems in place to help the driver. An example would be the emergency braking system since it technically doesn’t "drive" the vehicle, it does not qualify as automation[30]. This vehicle is operated completely manually. Every single decision is taken by human.

Level 1:

This is the lowest level of automation. The vehicle features a single automated system for driver assistance, such as steering or accelerating (cruise control). Adaptive cruise control, where the vehicle can be kept at a safe distance behind the next car, qualifies as Level 1 because the human driver monitors the other aspects of driving such as steering and braking[30]. ADAS (Advanced Driver Assistance System) supports the human driver with either steering, braking or speed. ADAS provides rear-view cameras and moving seat warning options to alert drivers while driving off the road.

Level 2:

This means advanced driver assistance systems or ADAS. The vehicle can control both steering and accelerating/decelerating. Here the automation falls short of self-driving because a human sits in the driver's seat and can take control of the car at any time. Tesla Autopilot and Cadillac (General Motors) Super Cruise systems both qualify as Level 2.

Similar to ADAS, vehicles of this level provide the functionality of indicating car movements, detection of neighbouring vehicles, etc.

Level 3:

The jump from Level 2 to Level 3 is substantial from a technological perspective, but subtle if not negligible from a human perspective. Level 3 vehicles have "environmental detection" capabilities and can make informed decisions for themselves, such as accelerating past a slow-moving vehicle. But they still require human override. The driver must remain alert and ready to take control if the system is unable to execute the task.

Almost two years ago, Audi (Volkswagen) announced that the next generation of the A8 their flagship sedan would be the world's first production Level 3 vehicle. And they delivered. The 2019 Audi A8L arrives in commercial dealerships this Fall. It features Traffic Jam Pilot, which combines a lidar scanner with advanced sensor fusion and processing power plus built-in redundancies should a component fail.

However, while Audi was developing their marvel of engineering, the regulatory process in the U.S. shifted from federal guidance to state-by-state mandates for autonomous vehicles. So for the time being, the A8L is still classified as a Level 2 vehicle in the United States and will ship without key hardware and software required to achieve Level 3 functionality. In Europe, however, Audi will roll out the full Level 3 A8L with Traffic Jam Pilot (in Germany first).Vehicles of this level perform all driving activities independently. However, in restricted conditions, such as car parking, the human driver should be able to take over the control.

Level 4:

The key difference between Level 3 and Level 4 automation is that Level 4 vehicles can intervene if things go wrong or there is a system failure. In this sense, these cars do not require human interaction in most circumstances. However, a human still has the option to manually override[30].

Level 4 vehicles can operate in self-driving mode. But until legislation and infrastructure evolves, they can only do so within a limited area (usually an urban environment where top speeds reach an average of 30mph). This is known as geofencing. As such, most Level 4 vehicles in existence are geared toward ridesharing. For example: NAVYA, a French company, is already

building and selling Level 4 shuttles and cabs in the U.S. that run fully on electric power and can reach a top speed of 55 mph.

Alphabet's Waymo recently unveiled a Level 4 self-driving taxi service in Arizona, where they had been testing driverless cars without a safety driver in the seat for more than a year and over 10 million miles.

Canadian automotive supplier Magna has developed technology (MAX4) to enable Level 4 capabilities in both urban and highway environments. They are working with Lyft to supply high-tech kits that turn vehicles into self-driving cars.

Just a few months ago, Volvo and Baidu announced a strategic partnership to jointly develop Level 4 electric vehicles that will serve the robotaxi market in China. Cars categorized under level 4 can perform all driving tasks and in bound situations controlling the driving environment. These are accurate enough in most of the conditions and demand very less human intervention.

Level 5:

Level 5 vehicles do not require human attentionthe “dynamic driving task” is eliminated. Level 5 cars won’t even have steering wheels or acceleration/braking pedals. They will be free from geofencing , able to go anywhere and do anything that an experienced human driver can do. Fully autonomous cars are undergoing testing in several pockets of the world, but none are yet available to the general public. The future cars or those grouped under this level serve as virtual drivers and keep moving in all informed situations. They have the capability to take independent decisions in known or unknown scenarios.

Our approach is a deep learning model which is based on ‘Behavioural Cloning’ , a concept in which the agent learns from human behaviour. This can be recognized as a level 5 algorithm. In this paper development of Convolutional Neural Network model and training with the data using behavioural cloning is performed. Image pre-processing and image augmentation is also performed to provide more data to the model. A comparative analysis is also performed with the other deep learning models on the simulator provided by the Udacity.

1.3 Problem Definition

Udacity released an open source simulator for self-driving cars to depict a real-time environment. The challenge is to mimic the driving behavior of a human on the simulator with the help of a model trained by deep neural networks [1]. The concept is called Behavioral Cloning, to mimic how a human drives. The simulator contains two tracks and two modes, namely, training mode and autonomous mode. The dataset is generated from the simulator by the user, driving the car in training mode. This dataset is also known as the “good” driving data. This is followed by testing on the track, seeing how the deep learning model performs after being trained by that user data. Another challenge is to generalize the performance on different tracks. That means, training the

model using the dataset created on one of the tracks, and testing it on the other track of the simulator. Another method followed for implementing and testing selfdriving cars is a simulator and algorithm based automated testing, it assesses the performance of the self-driving car using simulator. The simulation system will create different randomly, manually modified scenarios which are generated based on an algorithm. The genetic algorithm will be used for augmenting both manual and random scenes to identify the failures across the system. Developing a model which maps raw images can also be used. The focus is to develop a model that can map different raw images captured to perform training and testing to a correct steering angle using the deep learning algorithm. The data is collected using a vehicle platform built using 1/10 scale RC Car, Raspberry pi 3 Model B computer and Front facing camera.Prototype based approach for testing Self-Driving cars includes creating a self-driving car prototype (using simulation) that uses cameras for navigation and generates a better output. For prototype purpose a 3D virtual city is designed which depicts a real environment with traffic cars, traffic signals and different type of obstacles. The sole target of our self-driving vehicle should navigate easily without violating any traffic regulations and hitting any unwanted obstacles. It should be quick efficient and comfortable so that fuel consumption is reduced, and minimum jerks are felt throughout the journey.

1.4 Proposed Work

The objective behind this approach is to leverage the power of Supervised Learning in conjunction with Deep Learning to achieve our end goal. We focused on a concept called ‘Behavioural Cloning’[3], which, essentially, is a technique of teaching the machine to learn from a human subject. By Behavioral cloning human car driving actions are captured and along with the situation that gave rise to the action. A log of these records is used as input to the machine learning [3]. The simulator Udacity [5] is opensource and available with Training Mode or Autonomous Mode. In the training mode, manual car drive is used to record the driving behavior. The recorded images and the behaviour are used to train deep learning model. Once trained the Autonomous Mode is used for testing the machine learning models to see how the model can drive on new road. The simulator is used and first training data is generate data using behaviour cloning. The image data and actions are used to build and train the model. Different deep learning models CNN, ResNet50, DenseNet 201, VGG16, VGG19 are used for the evaluation purpose. The proposed modified CNN has achieved the better result in comparison of ResNet50, DenseNet 201, VGG16, VGG19. Convolutional Neural Network (ConvNet/CNN) is discussed next in the paper.

1.5 Solution Approach

The high-level architecture of the implementation can be seen in Figure 1.2.

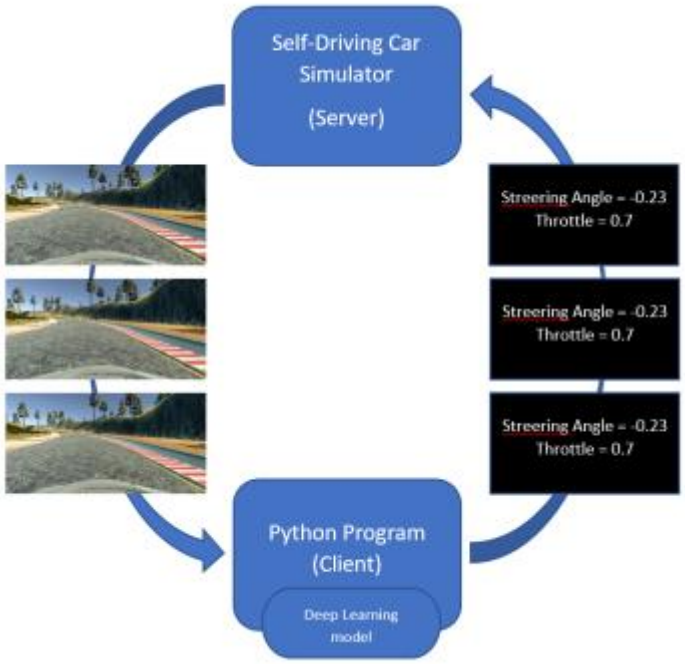


Figure 1.4: Implementation Architecture

The problem is solved in the following steps:

• The simulator can be used to collect data by driving the car in the training mode using a joystick or keyboard, providing the so called "good-driving" behavior input data in form of a driving_log (.csv file) and a set of images. The simulator acts as a server and pipes these images and data log to the Python client.

• The client (Python program) is the machine learning model built using Deep Neural Networks. These models are developed on Keras (a high-level API over Tensorflow). Keras provides sequential models to build a linear stack of network layers. Such models are used in the project to train over the datasets as the second 3 step. Detailed description of CNN models experimented and used can be referred to in the chapter on network architectures.

• Once the model is trained, it provides steering angles and throttle to drive in an autonomous mode to the server (simulator).

• These modules, or inputs, are piped back to the server and are used to drive the car autonomously in the simulator and keep it from falling off the track.

Prototype based approach for testing Self-Driving cars includes creating a self-driving car prototype (using simulation) that uses cameras for navigation and generates a better output. For

prototype purpose a 3D virtual city is designed which depicts a real environment with traffic cars, traffic signals and different type of obstacles. The sole target of our self-driving vehicle should navigate easily without violating any traffic regulations and hitting any unwanted obstacles. It should be quick efficient and comfortable so that fuel consumption is reduced, and minimum jerks are felt throughout the journey.

1.6 Concepts Involved

Convolutional Neural Network (ConvNet/CNN): Inspired by the organization of Visual Cortex, CNNs take an image as input, assigns importance to different features/objects of the image, thereby learning about its characteristics. It involves a series of convolution operations.

$$Dim(H_1, W_1, D_1) = \left(\frac{H+2Z_p-k_1}{Z_s} + 1, \frac{W+2Z_p-k_2}{Z_s} + 1, K_D\right) \qquad (1)$$

Where H1,W1 , and D1 are height ,width and number of kernels as output calculate by the input tensor H is height, W is width Zp is padding,K1and K2 are height and width of kernel, Zs is stride using equation (1). Flatten layer: In order to pass on information from a 2D network to a fully connected layer, a conversion to lower dimension is required. Flatten layer generally follows a convolution 2D layer and precedes a dense layer as an intermediate converter. Image Augmentation: Using the technique of Image Augmentation, a huge amount of training data can be generated through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc [19]. Loss function: Loss function is simply a measure of how wrong the predictions of the model are. There are several loss functions including Mean Square Error, Cross Entropy Loss etc. The Mean Square Error is measured as the Mean of the square of the difference between the actual observation and the predicted value.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \qquad (2)$$

Mathematically, Mean Square Error is defined using the equation (2). Activation functions: Activation functions are used to convert an input signal into an output signal. These are what empowers the Neural Networks by allowing them to understand linear as well as non-linear relationships in the data. There are several activation functions such as Sigmoid, ReLu [20], Tanh, Softmax etc. ReLU: ReLU (Rectified Linear Unit) is a half-rectified activation function that is most widely used in convolutional neural networks. Here, both the function and its derivative are monotonic. However, it converts negative values immediately to zero. This decreases the learnability of the model[20].

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases} \quad (3)$$

Equation (3) is used to define ReLU function mathematically. ELU: ELU (Exponential Linear Unit) activation function is similar to RELU [20] activation function, but helps the model to converge to zero much faster and produce more accurate results [21]. It can be mathematically stated as:

$$R(z) = \begin{cases} z & z > 0 \\ \alpha(e^z - 1) & z \leq 0 \end{cases} \quad (4)$$

ELU is defined mathematically using equation (4). Optimisers: Optimisers control learning of the model, by updating the model parameters in response to the output of the loss function. Adam: There are several optimisers but the most notable one is adam. The reason adam is preferred over classical stochastic gradient descent is that stochastic gradient descent maintains a single learning rate for all the weight updates whereas in adam the learning rate is unique for each weight update and updated separately [22]. Other bechmark deep learning algorithms are used in the experiment for the evaluation purpose are ResNet50, DenseNet 201, VGG16, VGG19. ResNet 50: ResNet is a powerful Deep Neural Network and won the first place on ImageNet [23] detection, ImageNet [23] localization, COCO (Common Objects in Context) [24] detection and COCO [24] segmentation in ILSVRC and COCO 2015. ResNet stands for Residual Network. Each layer feeds its output to the next layer and also to the layer 2-3 hops away from it. ResNet 50 is used in this paper, which is a 50-layer Residual Network [25]. Figure 1.5 represents the block diagram of Residual Network.

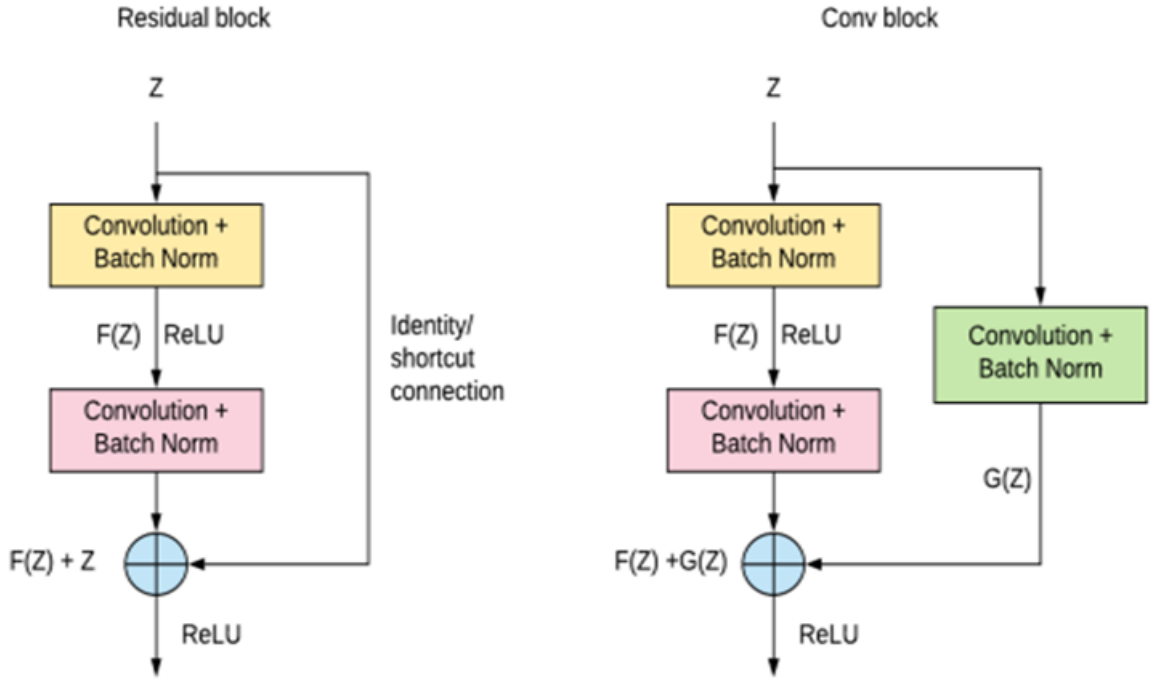


Figure 1.5: Residual Networks

DenseNet 201: DenseNet stands for Dense Convolutional Network. In DenseNet, each layer receives feature maps from all the previous layers. This makes the network quite compact. DenseNet 201 is a 201 layers deep network that is trained on more than a million images from Imagenet [23][26]. Concept behind DenseNet has been explained in Figure-1.6.

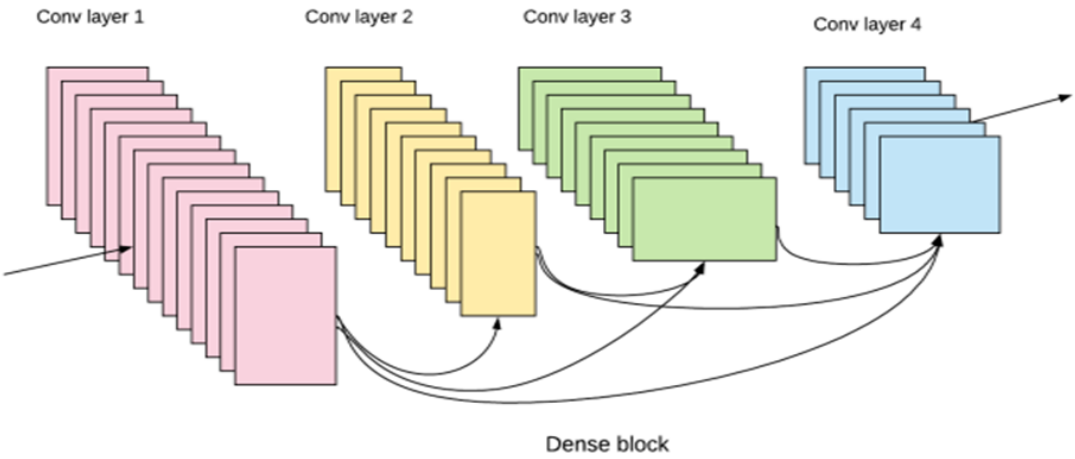


Figure 1.6: Dense Block

VGG: VGGNet is invented by Visual Geometry Group(VGG) from University of Oxford and was the 1st runner up in ILSVRC 2014. VGG16 is 16 layers deep while VGG19 is 19 layers deep. The notable characteristic of VGG networks is the use of 3x3 convolution filters in the Convolution layers, thereby allowing the network to be deeper [27].

1.7 Technologies Used

Technologies that are used in the implementation of this project and the motivation behind using these are described in this section. TensorFlow: This an open-source library for dataflow programming. It is widely used for machine learning applications. It is also used as both a math library and for large computation. For this project Keras, a high-level API that uses TensorFlow as the backend is used. Keras facilitate in building the models easily as it more user friendly. Different libraries are available in Python that helps in machine learning projects. Several of those libraries have improved the performance of this project. Few of them are mentioned in this section. First, “Numpy” that provides with high-level math function collection to support multi-dimensional metrices and arrays. This is used for faster computations over the weights (gradients) in neural networks. Second, “scikit-learn” is a machine learning library for Python which features different algorithms and Machine Learning function packages. Another one is OpenCV (Open Source Computer Vision Library) which is designed for computational efficiency with focus on real-time 4 applications. In this project, OpenCV is used for image preprocessing and augmentation techniques. The project makes use of MiniConda Environment which is an open source distribution for Python which simplifies package management and deployment. It is best for large scale data processing. The machine on which this project was built, is a personal computer with following configuration:

• Processor: Intel(R) Core i5-7200U @ 2.7GHz

• RAM: 8GB

• System: 64bit OS, x64 processor

2. LITERATURE SURVEY

2.1 Previous implementation methodologies:

In order to understand the development of research in autonomous driving in the last years, it is important to conduct a literature review to understand the different fields of application through which autonomous driving has evolved as well as to identify research gaps. Therefore in the next sections the research process, methodology and findings of the literature review are presented.

The research conducted focused in a systematic keyword search in the topic section of literature databases, including EBSCO, Science Direct, Emerald and ISI web of knowledge. The search conducted included the specific terms, “Autonomous driving”, “Self driving car” and “Driverless car” either in the title, keywords or abstract and including only academic journals listed in the mentioned databases. Hence, the aim of this research was not to find all literature regarding Autonomous Driving that because of its size would lead to an enormous amount of results due to the extensive applications, testing and research in other fields (robotics, underwater vehicles, military, aeronautics, space vehicles, etc.), but to achieve an overview and overall classification to identify current gaps in the scientific literature body. Therefore taking into consideration only the literature publications relevant for roads, traffic, crossroads and studies related to commuting, transportation or production, and including all relevant publications found related to the automotive industry, as well as also considering papers in other topics that acknowledge that the application could be relevant for self-driving cars. The search yielded 483 papers, 154 of which were found in EBSCO, 122 in Science Direct, 8 in Emerald and 199 in ISI Web of Knowledge. Furthermore, a thorough analysis of abstracts, titles and journals was conducted in order to eliminate duplicates and results that were found in more than one database, leading to the elimination of 63 repetitions, some of which were even found 3 times, reflecting the thoroughness of the search, delivering a total amount of 420 publications. Finally a last filtering process eliminated publications not fulfilling scientific standards nor a peer-review process lead to the exclusion of 21 further publications delivering a final number of 399 Publications. A set of analysis explained in the next sections were then conducted with the remaining 399 publications in order to understand of the current state of the literature, its focus and development over time.

Cites	Authors	Title	Year	*Source*
557	C. Urmson C. Baker et ál.	Autonomous driving in urban environments: Boss and the Urban Challenge	2008	*Journal of Field Robotics*
364	M. Bertozzi A. Broggi, A. Fascioli,	Vision-based intelligent vehicles: State of the art and perspectives	2000	*Robotics and Autonomous Systems*
330	P.A. Ioannou, C.C. Chien,	Autonomous intelligent cruise control	1993	*IEEE Transactions on vehicular technology*
311	U. Franke, D. Gavrila, et ál	Autonomous driving goes downtown	1998	*IEEE Intelligent systems and their applications*
230	P. Hidas	Modelling lane changing and merging in microscopic traffic simulation.	2002	*Transportation Research: Part C*
201	J. Leonard, J. How, et ál.	A Perception-Driven Autonomous Urban Vehicle	2008	*Journal of Field Robotics*
164	Y. Kuwata, S. Karaman, et ál.	Real-Time Motion Planning With Applications to Autonomous Urban Driving	2009	*IEEE Transactions on control systems technology*
163	W. Wijesoma, K.R.S. Kodagoda, et ál.	Road-Boundary Detection and Tracking Using Ladar Sensing.	2004	*IEEE Transactions on Robotics & Automation.*
157	K. Konolige, J. Bowman et ál.	View-based Maps.	2010	*International Journal of Robotics Research*
155	T.-H.S Li, C. Shih-Jie et ál.	Implementation of Human-Like Driving Skills by Autonomous Fuzzy Behavior Control on an FPGA-Based Car-Like Mobile Robot.	2003	*IEEE Transactions on Industrial Electronics.*

Table 2.1: Most cited publications in Autonomous Driving Field

2.2 Analysis of publications over time:

Through this method, good insight can be gained on the origin and development of a research field over time (Dahlander & Gann, 2010). As shown in Figure 2.1 we can observe the development of self-driving cars research throughout the last decades, demonstrating at the same time a continuous increase of the interest on the topic as well as important milestones that can also influence the topic development and research such as in this case the DARPA Grand Challenges in the

beginning of 2004 and end of 2005 or the Urban Challenge in the end of 2007 (DARPA, 2015), that were relatively important events in the field. Furthermore, from 2013 to 2014 a rapid increase of 60.8% can be observed, which can be related to the actual interest on the topic.

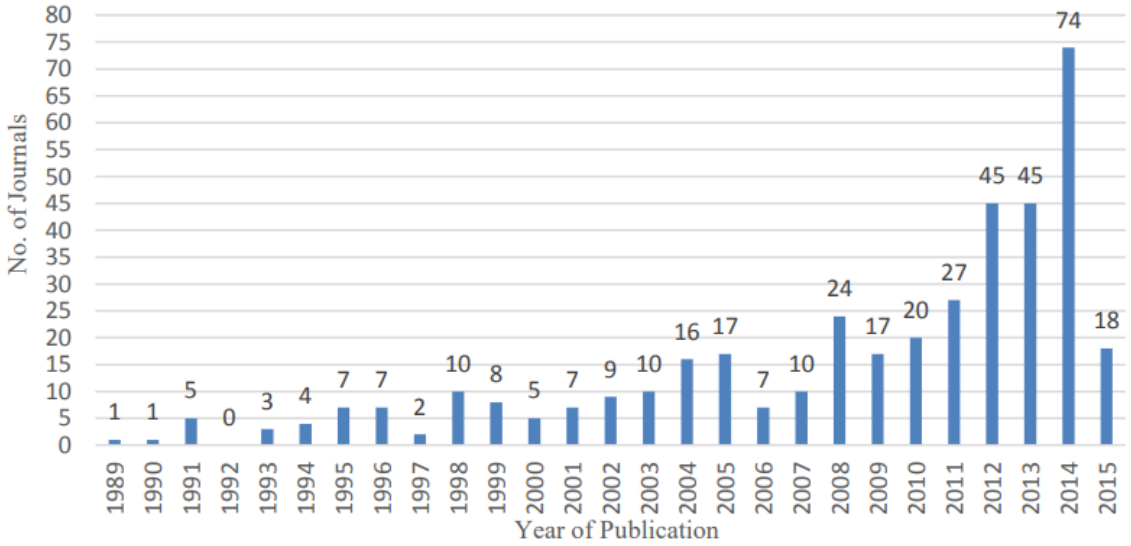


Figure 2.1: Analysis of publications over the years

With the rapid developments in computer technology, there has been a great increase in the capabilities of computers. Now, the computers are capable of performing Terabytes of calculation in just a few minutes. This has paved a path for the growth of Machine Learning amongst the community. An extensive study of the research done in the field of Self-Driving cars is conducted and explored. Reinforcement Learning and Deep Learning are the two main approaches involved in solving the problem of self-driving cars. However, this research work focuses on the Deep Learning based approach since it is capable of capturing the style and expertise of the driver through a concept known as Behavioural Cloning .

With the onset of powerful Graphical Processing Units (GPUs), Deep Learning has become a choice of technology for researchers as well as for developers working in the field of self-driving cars. With the combination of Deep Learning and OpenCV[6], there is a huge opportunity for building robust autonomous vehicles. Fujiyoshi et al. demonstrated how Deep Learning involving CNNs are being used in the tasks of Image Recognition, Object Detection and Semantic Segmentation .

Further, the paper presents how Deep Learning is being used currently to infer the control values of an autonomous car through technique known as end-to-end. It also presents visual explanation for such a technique which can boost the confidence of passengers of autonomous cars. Kulkarni et al. demonstrated how Faster Region based Convolutional Network (R-CNN) [8]

can be used for detecting traffic lights efficiently [9]. Bojarski et al. achieved an autonomy value of 90% in virtual environments using his endto- end approach consisting of a Convolutional Neural Network (CNN) .

This approach demonstrated how a single-camera can be used to feed images to CNN to get the steering angles accurately. Jain built a working model of the self-driving car using Raspberry Pi and Lidar sensor and showed how CNNs can be used to drive a real-world model of a car without any human intervention . Kim et al. collected the data from a game using end-to-end method and employed an autonomous driving technique announced by Nvidia . Further, authors have used AlexNet, one of the most popular CNN models.

Kang et al. did a thorough study of 37 publicly-available datasets and 22 virtual testing environments, which serves as a guide for researchers and developers involved in the designing of self-driving cars . It included various datasets like Berkeley DeepDrive Video dataset (BDDV) [14], Cambridge-driving Labeled Video Database (CamVid) [15]etc. The various virtual testing environments studied in the above mentioned paper included AirSim (Microsoft), CARLA , TORCS and several others.

3. IMPLEMENTATION

3.1 Dataset Generation:

This section consists of training a Deep-Learning model using the dataset gathered by our experiment as human subject, whose behaviour is to be cloned. This dataset was recorded using the Udacity Self-Driving Car Simulator [5]. The simulator consists of two modes:

1. Training Mode - It is used for recording the dataset
2. Autonomous Mode - It is used for testing the model.

The simulator is based on Unity Engine. The car has three cameras attached to it: one in the front, and two at each side of the car. Figures 3.4, 3.5 show the training track and testing track in Udacity simulator. Path provided in the simulator during the training phase involves a one-way track, which had multiple curvatures, obstacles, bridges, and rivers. Testing track is an unknown mountain area, which has up-down scenarios and non-uniform curvatures. The first mode, Training Mode, is used to gather the dataset. Therefore, they used this mode to gather the dataset. They drove multiple rounds each in forward as well as in backward direction on the first track.

Initially the dataset contained around 4053 frames along with the respective labels. Since, the dataset was highly skewed towards 0 degree, they under sampled it to get a more balanced dataset. They selected a threshold value of 350 i.e. corresponding to each angle we can have a maximum of 350 frames. This threshold was selected keeping in mind the data-imbalance condition as well as to maintain a decent count of the frames available for training our model. The dataset was then split into Training Set and Validation set in the ratio of 4:1.

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.

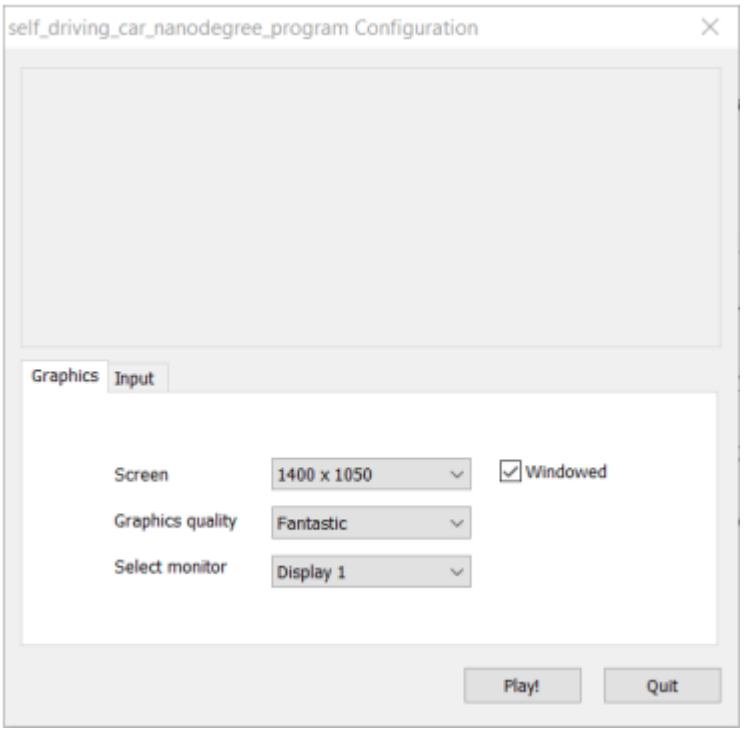


Figure 3.1: Configuration screen

The graphics and input configurations can be changed according to user preference and machine configuration as shown in Figure 3.1 . The user pushes the “Play!” button to enter the simulator user interface. We can enter the Controls tab to explore the keyboard controls, quite similar to a racing game which can be seen in Figure 3.2.

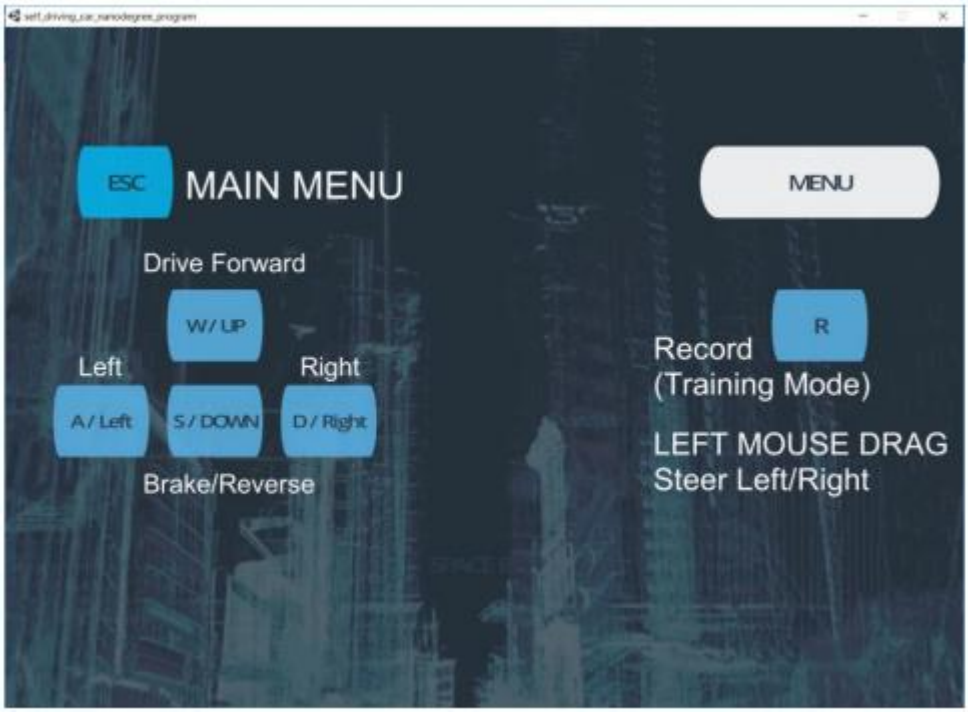


Figure 3.2: Controls Configuration

Figure 3.3: First Screen

The first actual screen of the simulator can be seen in Figure 3.3 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 3.4 and Figure 3.5. The word “simple” here just means that it has fewer curvy tracks and is easier to drive on, refer Figure.3.5.

Figure 3.4: Track_1

The “complex” track has steep elevations, sharp turns, shadowed environment, and is tough to drive on, even by a user doing it manually. Please refer Figure.3.6

Figure 3.5: 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 3.4 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. The mode screenshot can be as seen in Figure 3.6.

Figure 3.6: Autonomous mode

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 labeled 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 few images from the dataset are shown in Figures 3.7, 3.8, 3.9, 3.10, 3.11, 3.12.

Figure 3.7:Center0001

Figure3.8:Right0001

Figure3.9:Left0001

Figure3.10:Center0099

Figure3.11:Right0099

Figure3.12:Left0099

A sample of datalog.csv file is shown in Figure 3.13.

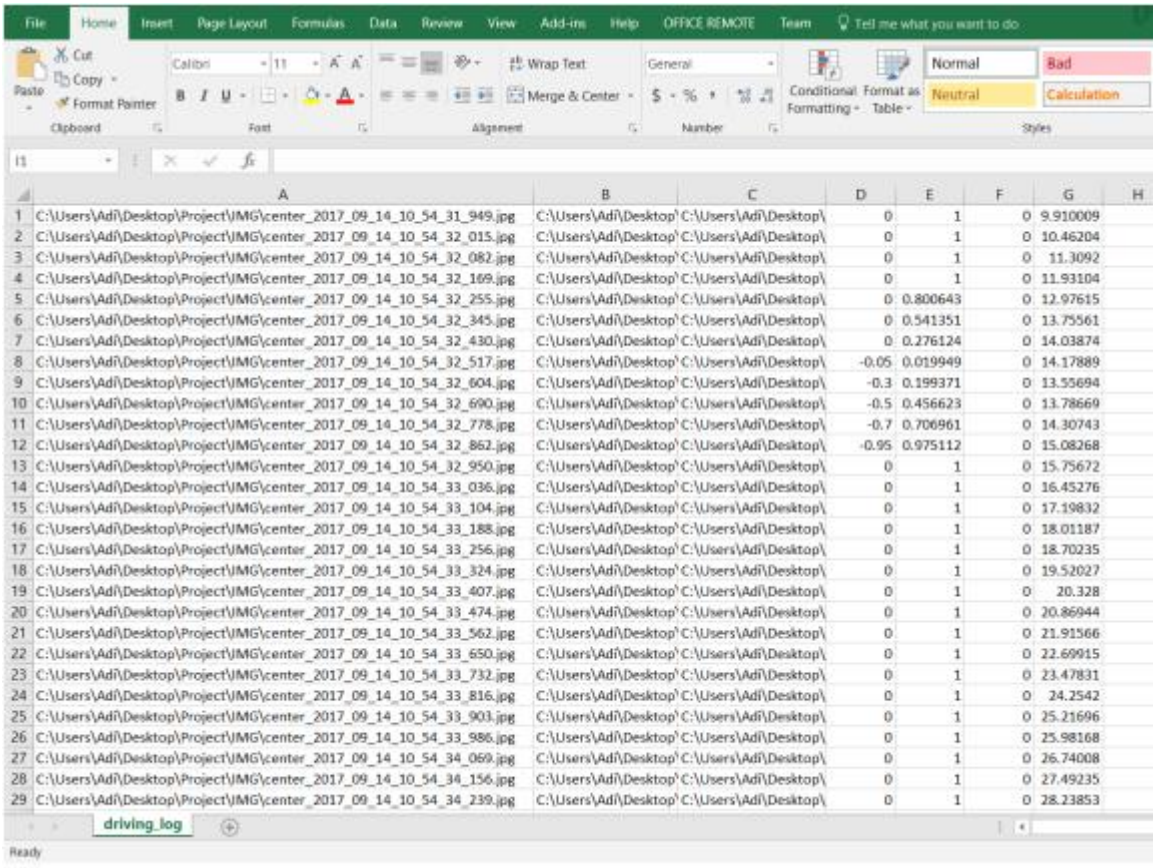
Column 1, 2, 3: contains paths to the dataset images of center, right and left respectively

Column 4: contains the steering angle Column value as 0 depicts straight, positive value is right turn and negative value is left turn.

Column 5: contains the throttle or acceleration at that instance

Column 6: contains the brakes or deceleration at that instance

Column 7: contains the speed of the vehicle



	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

Figure3.13:driving_log.csv

3.2 Preprocessing and Image Augmentation:

After having our dataset in place, the next step was to preprocess the data before providing it to CNN [4] based model. To preprocess the data we harnessed the power of OpenCV 2 [6] library. The preprocessing applied to the input data were cropping the image ,converting the Channel from RGB to YUV, applying Gaussian Blur, resizing the image and normalization. Therefore, each frame, that is passed to the model is first preprocessed as explained in the pseudo-code below:

Algorithm 1 Deep Learning Image Preprocessing

```
Require: img_matrix: Image as a matrix
    1.  Function preprocess(img_matrix)
    2.  img_matrix ← img_matrix[60:137, :, :]
    3.  img_matrix ← YUV(img_matrix)
    4.  img_matrix ← GaussianBlur(img_matrix, (3, 3), 0)
    5.  img_matrix ← resize(img_matrix), (200, 66))
    6.  img_matrix ← img_matrix/255
    7.  Return img_matrix
    8.  End function
```

Figure3.14: Image preprocessing algorithm

Conversion of the channel is performed to get the pixels in YUV format. This is performed because YUV channel is more efficient and reduces the bandwidth more than what RGB can capture [28]. In Gaussian Blur, the image is convolved using a Gaussian filter to minimise the noise in the image [19]. Gaussian filter basically uses Gaussian function, which is applied to each pixel in the image. The Gaussian function for two-dimensions is shown in below equation :

$$G(x,y) = \frac{e^{-(x^2+y^2)/2\sigma^2}}{2\Pi\sigma^2}$$

Further, normalization was performed to reduce each pixel to a similar data distribution which allows faster convergence of the model. The formula for Normalization is illustrated in equation shown below:

$$input\ para = input\ para/255$$

Our next step was to leverage the power of a concept known as Image Augmentation, which in the past has proved to be highly beneficial for any CNN [4] based model. Zooming() function is used for producing a new image from the given image by either zooming in or zooming out. Panning() function is used for producing translation in the image. brightness () function is used to multiply each intensity value of pixels in Image to brighten or darken the image. flipping () function flips the image Horizontally as well as the corresponding Steering Angle. In our model we used ELU activation function [21]. Adam optimizer [22] is used with default parameters along with Mean Square Error as the regression loss function. Batch size of 32 was selected. The pseudo-code for the process of Image Augmentation is described in algorithm 2:

Algorithm 2 Deep Learning Image Augmentation

Require: img_matrix: Image as a matrix; steer: Steering Angle

```
Function augment(img_matrix, steer)
number ← Random()
if number > 0.6 then
img_matrix← zooming(img_matrix)
if number > 0.6 then
img_matrix← panning(img_matrix)
if number > 0.6 then
img_matrix← brightness(img_matrix)
if number > 0.6 then
img_matrix, steer ← flipping(img_matrix, steer)
Return img_matrix, steer
End function
```

Figure 3.15: Deep Learning Image Augmentation Algorithm

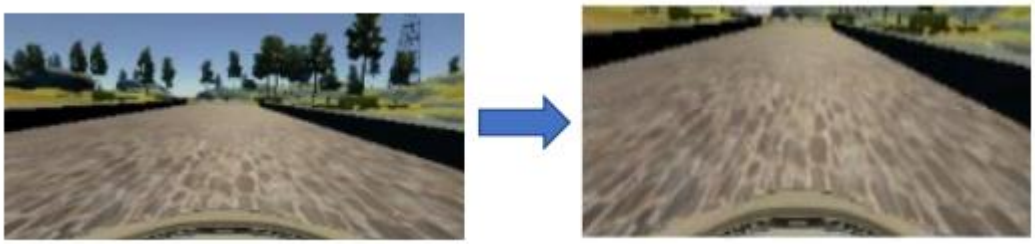


Figure 3.16: Crop Image

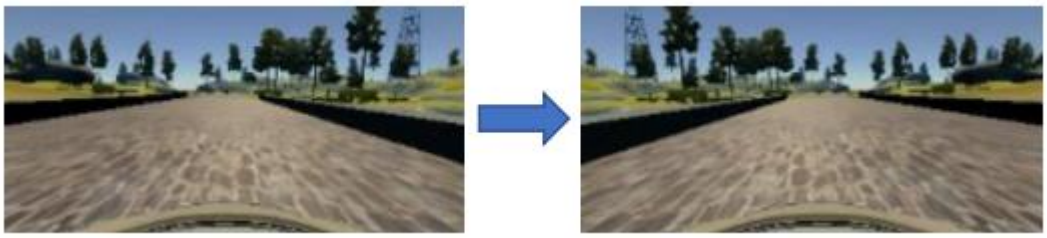


Figure 3.17: Flip Image

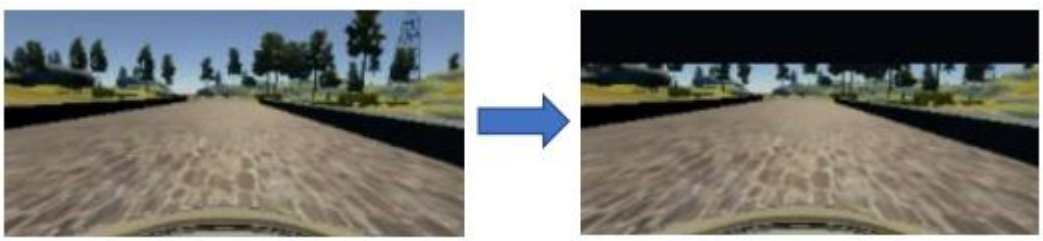


Figure 3.18: Shift Image vertical

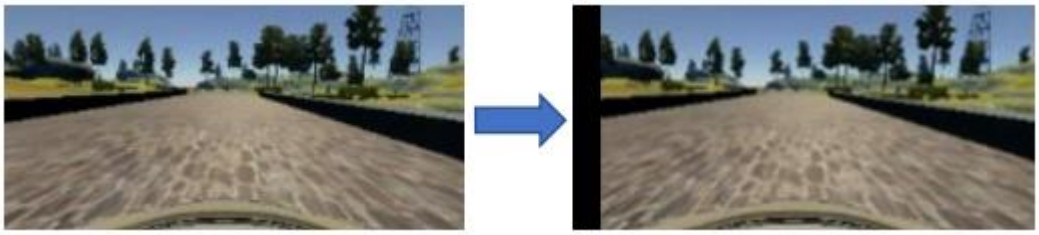


Figure 3.19: Shift Image horizontal

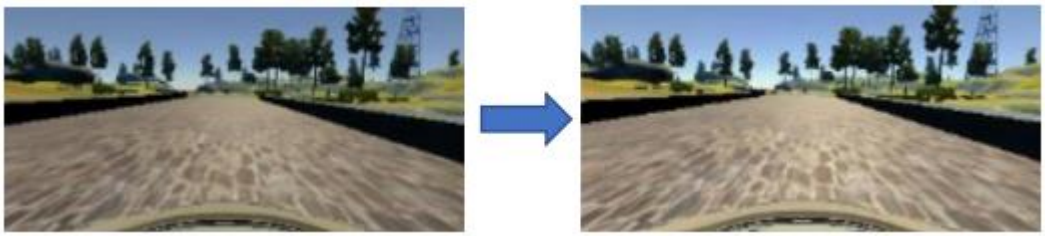


Figure 3.20: Brightness increase

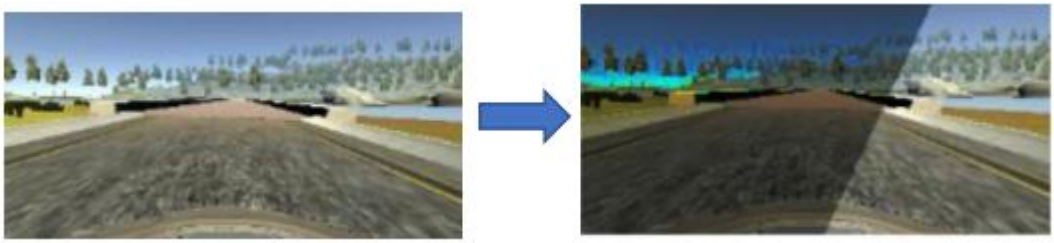


Figure 3.21: Random Shadows

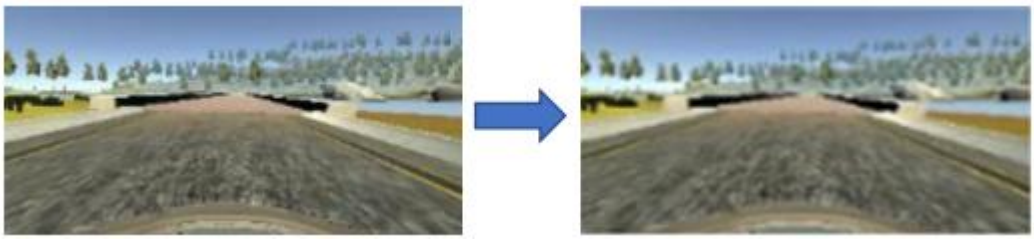


Figure 3.22: Random blur

Figure 3.23: Random Noise

3.3 Training and Validation:

This section consists of the configurations used to set up the models for training the Python Client to provide the Neural Network outputs that drive the car on the simulator. The tweaking of parameters and rigorous experiments were tried to reach the best combination. Though each of the models had their unique behaviors and differed in their performance with each tweak, the following combination of configuration can be considered as the optimal:

• The sequential models built on Keras with deep neural network layers are used to train the data.

• Models are only trained using the dataset from Track_1.

• 80% of the dataset is used for training, 20% is used for testing.

• Epochs = 50, i.e. number of iterations or passes through the complete dataset. Experimented with larger number of epochs also, but the model tried to "overfit". In other words, the model learns the details in the training data too well, while impacting the performance on new dataset.

• Batch-size = 40, i.e. number of image samples propagated through the network, like a subset of data as complete dataset is too big to be passed all at once.

• Learning rate = 0.0001, i.e. how the coefficients of the weights or gradients change in the network

• ModelCheckpoint() is the function provided in Keras to save checkpoints and to save the best epoch according to the validation loss. There are different combinations of Convolution layer, Time-Distributed layer, MaxPooling layer, Flatten, Dropout, dense and so on, that can be used to implement the Neural Network models. Out of around ten different architectures I tried, three of the best ones are discussed in the chapter on network architectures.

3.4 Testing:

After training the model with the proposed CNN architecture with the preprocessed image, the model is connected to the simulator to predict the steering angles in real-time. The frames were pre-processed, and the model passed back the Steering Angles and Throttle values back to the simulator in real-time. We trained our model using the frames obtained from the first track. For testing the model, we allowed the model to drive the car in a track it had never seen before, i.e different track .Throttle value is calculated using the equation. The entire process can be summarized using the workflow given in Figure 3.24 .

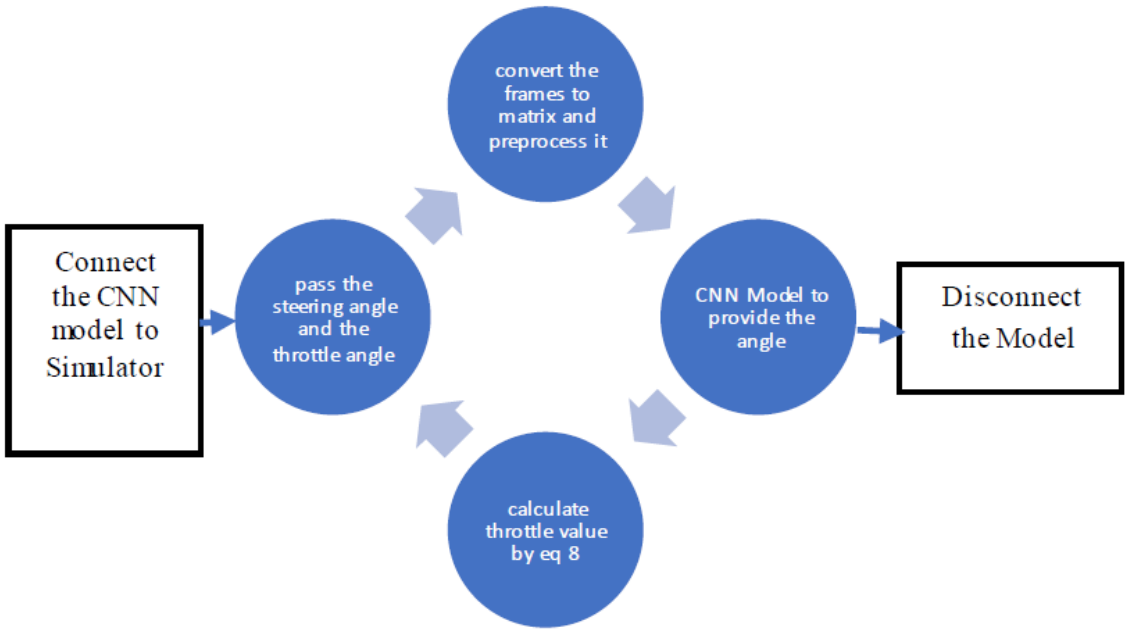


Figure 3.24: Workflow diagram of the model with simulator

4. EXPERIMENTAL RESULTS

The experimental results of proposed model are astonishing. The model is trained on the frames from the first track and tested by driving autonomously on an unseen second track.

Around 1500 images were passed to the generator function (for performing image augmentation during the training). The Deep Learning model was trained for 20 epochs using Mean Square Error as loss function and optimiser was selected to be adam optimiser [22]. It took around 3240.2 seconds for the entire process on Kaggle's powerful k80 GPU [29]. The Loss of training and validation is shown in the figure 4.1. The Validation loss keeps on decreasing and both the validation & Training loss seems to decrease as the epoch count increases

and converges at around 17 epoch, which is a good sign and indicates the model has learnt the correlations well and has not over-fit on the training set. The least MSE loss obtained is

0.029 on the validation set. The car remained stable for most of the duration and drove with perfection. This approach is indeed a good method for implementing Self-Driving Car in

simulation. It requires no separate feature extraction like identification of the edges of road or identification of the lanes etc.

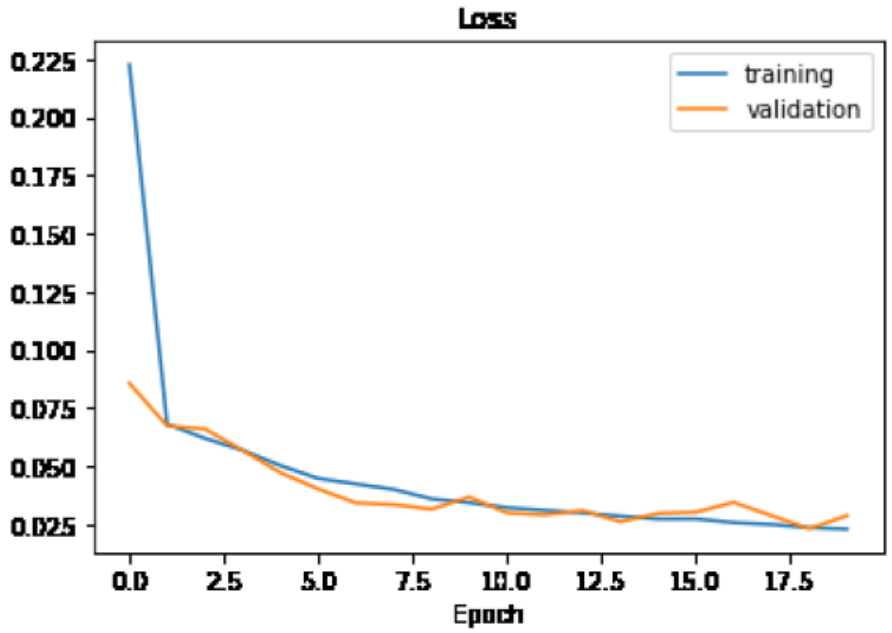


Figure 4.1: Loss vs. Epoch plot

Further, we experimented with different activation functions like ReLU [20], Leaky ReLU [20] and ELU [21], but ELU [21] seemed to perform better than the rest. So, ELU is used in the

experiment. Further standard models like Resnet 50 [25], DenseNet 201 [26], VGG16 [27]and VGG19 [27] as feature extractors followed by 1 or 2 dense layers are also trained and tested. The comparative analysis of the proposed model with other benchmark models is shown in Table 4.1. The Mean Square Error is measured as the difference between the actual observation and the predicted value for all the models. The proposed systems have limitations as the proposed model is trained with limited data on certain tracks. For real life we require no error and potential for working in different weather condition and situations.

MODEL	LOSS
Resnet50	0.03109
DenseNet 201	0.02590
VGG16	0.05303
VGG19	0.03153
Proposed Model	0.02914

Table 4.1: Comparative results analysis of models

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)

1.Value loss or Accuracy 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.

2 Generalization on Track_2 (Drive Performance) The second metric that was used to evaluate the results is generalization. This can be defined as how well the values are predicted by the models to drive over a different track it was not trained for. Here, the values are the predicted steering angle, brakes, and throttle. It is not something that can be plotted, but the evaluation can be done in terms of how far the car drives on the second track, without toppling down. The several factors affecting

this can be the speed, turns, conditions on track like elevations, shadows, and so on. As discussed, the models are only trained on Track_1 data which was simpler but tested on Track_2. Thus, it was significantly challenging when dealing with Track_2. Few upcoming observations were noted while experimenting, that supports this claim.

• Though it performed well on Track_1 and gave the best accuracy during training (loss over epochs), most architectures were not able to even take the first turn over the Track_2.

• The reason could be overfitting for the Track_1 dataset. Overfitting refers to a situation in which the program tries to model the training data too well. In general, the model trains for details and even the noise for the data it has passed through. This, in turn, negatively impacts the generalizing ability.

• Other reasons could be that, Track_2 starts with a road running parallel to the one the car starts at, with a barrier in between. . This can be one of such scenes that it can never encounter while training for Track_1 To work around this problem, the image preprocessing and augmentation techniques are used.

• Though the accuracy was highest for Architecture_2 (i.e. val_loss was the least), it could not perform that well on Track_2. In fact, Architecture_3 gave better results while driving on Track_2

• Ratings indicated 1 as the inferior performance, 5 being mediocre, and 10 being the best. Note that these ratings are purely qualitative and not precise measurements. There were instances where the car could not even drive a small distance and tried turning into the other parallel track right away and got stuck in the barrier right at the beginning. This scenario can be treated as rating 1.

• Increasing the number of convolution and max-pooling layers may get you better results. Thus, some architectures performed better than the others, but larger networks take more time to compute (train). Having said that, this might not be true that increasing the number of convolution layers will improve the results every time. When increasing the number further, very similar results were obtained, even with worse performances sometimes.

5 DISCUSSIONS

5.1 Real life autonomous vehicles:

Some of the real life implementations of Autonomous vehicles are as follows: Three CNN networks that are used by three companies pioneering self-driving cars are

1.HydraNet by Tesla

2.ChauffeurNet by Google Waymo

3.Nvidia Self driving car

HydraNet – semantic segmentation for self-driving cars

HydraNet was introduced by Raviet. It was developed for semantic segmentation, for improving computational efficiency during inference time.

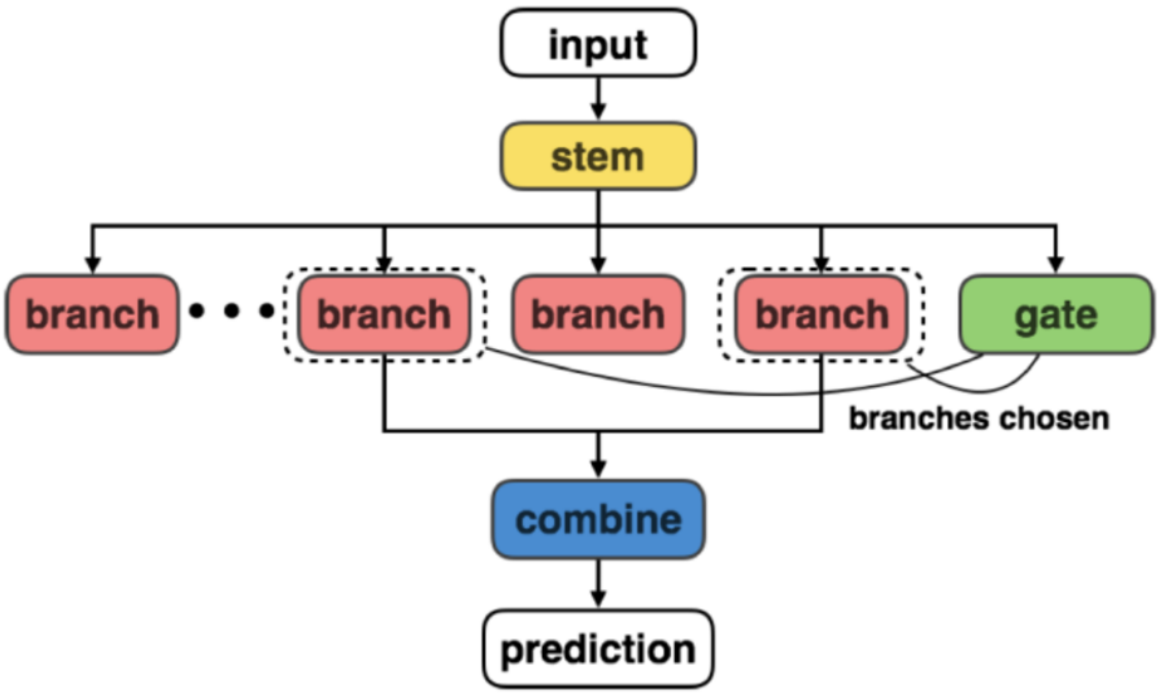


Figure 5.1: HydraNet Architecture

HydraNets is dynamic architecture so it can have different CNN networks, each assigned to different tasks. These blocks or networks are called branches. The idea of HydraNet is to get various inputs and feed them into a task-specific CNN network.

Take the context of self-driving cars. One input dataset can be of static environments like trees and road-railing, another can be of the road and the lanes, another of traffic lights and road, and so on. These inputs are trained in different branches. During the inference time,

the **gate** chooses which branches to run, and the combiner aggregates branch outputs and makes a final decision.

In the case of Tesla, they have modified this network slightly because it's difficult to segregate data for the individual tasks during inference. To overcome that problem, engineers at Tesla developed a shared backbone. The shared backbones are usually modified ResNet-50 blocks.

This HydraNet is trained on all the object's data. There are task-specific heads that allow the model to predict task-specific outputs. The heads are based on semantic segmentation architecture like the U-Net.

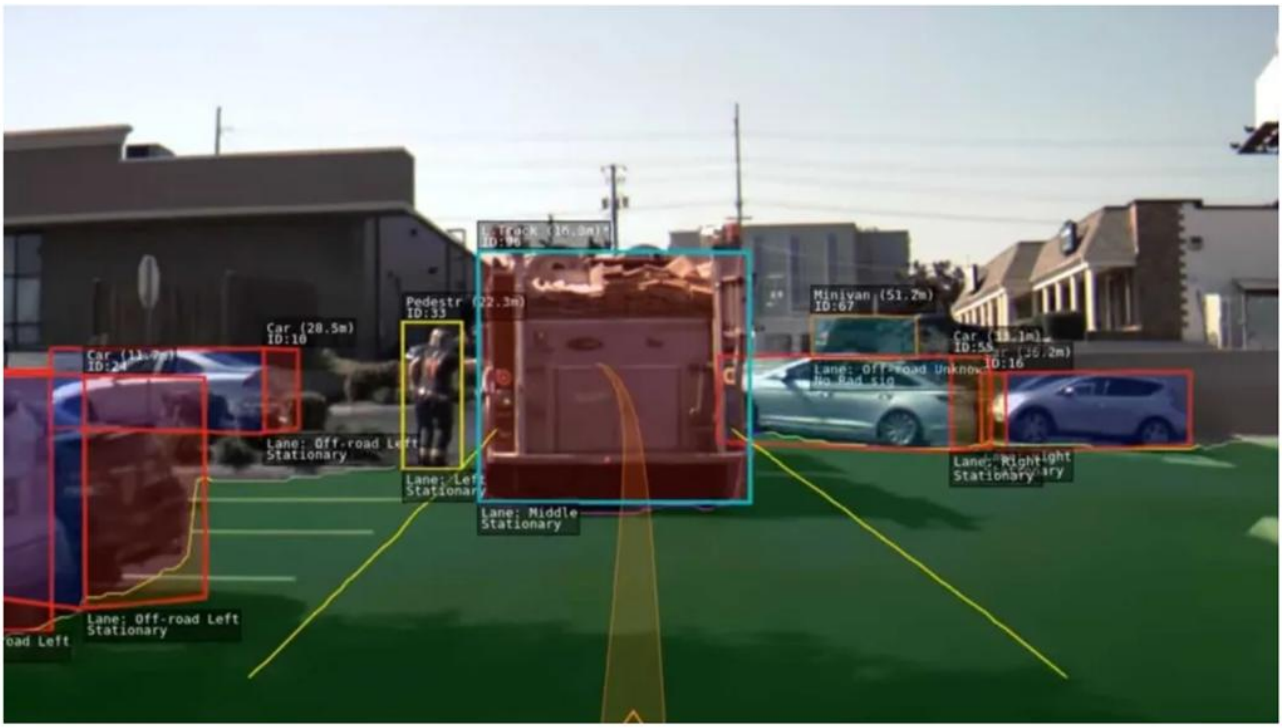


Figure 5.2: Object detection in HydraNet by Tesla

The Tesla HydraNet can also project a birds-eye, meaning it can create a 3D view of the environment from any angle, giving the car much more dimensionality to navigate properly. It's important to know that Tesla doesn't use LiDAR sensors. It has only two sensors, a camera and a radar. Although LiDAR explicitly creates depth perception for the car, Tesla's hydranet is so efficient that it can stitch all the visual information from the 8 cameras in it and create depth perception.

Figure 5.3: Self Driving car's camera

ChauffeurNet: training self-driving car using imitation learning

ChauffeurNet is an RNN-based neural network used by Google Waymo, however, CNN is actually one of the core components here and it's used to extract features from the perception system.

The CNN in ChauffeurNet is described as a convolutional feature network, or FeatureNet, that extracts contextual feature representation shared by the other networks. These representations are then fed to a recurrent agent network (AgentRNN) that iteratively yields the prediction of successive points in the driving trajectory.

The idea behind this network is to train a self-driving car using imitation learning. In the paper released by Bansal et al "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst", they argue that training a self-driving car even with 30 million examples is not enough. In order to tackle that limitation, the authors trained the car in synthetic data. This synthetic data introduced deviations such as introducing perturbation to the trajectory path, adding obstacles, introducing unnatural scenes, etc. They found that such synthetic data was able to train the car much more efficiently than the normal data.

Usually, self-driving has an end-to-end process as we saw earlier, where the perception system is part of a deep learning algorithm along with planning and controlling. In the case of ChauffeurNet, the perception system is not a part of the end-to-end process; instead, it's a mid-

level system where the network can have different variations of input from the perception system.

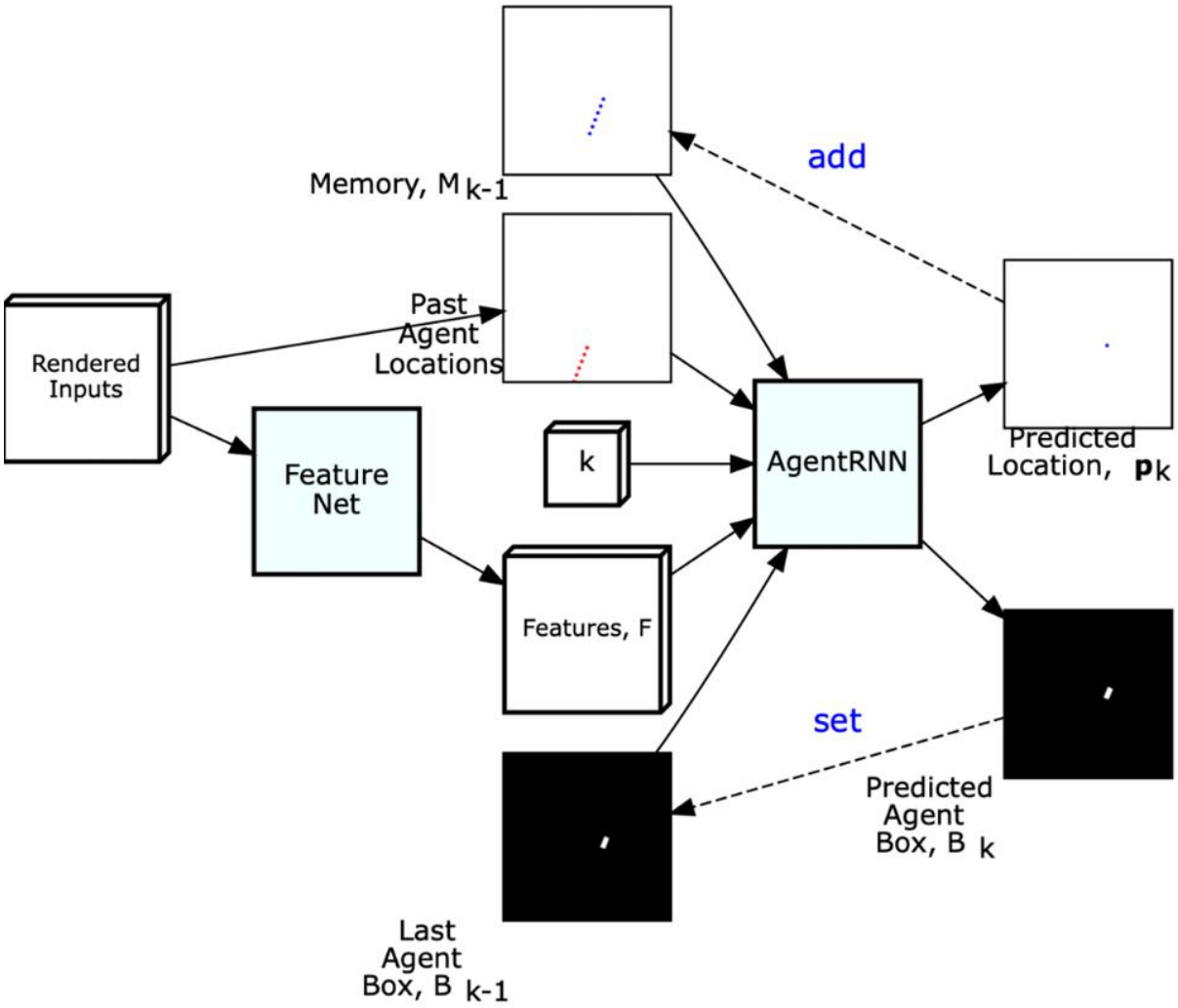


Figure 5.4: ChauffeurNet Architecture

ChauffeurNet yields a driving trajectory by observing a mid-level representation of the scene from the sensors, using the input along with synthetic data to imitate an expert driver.

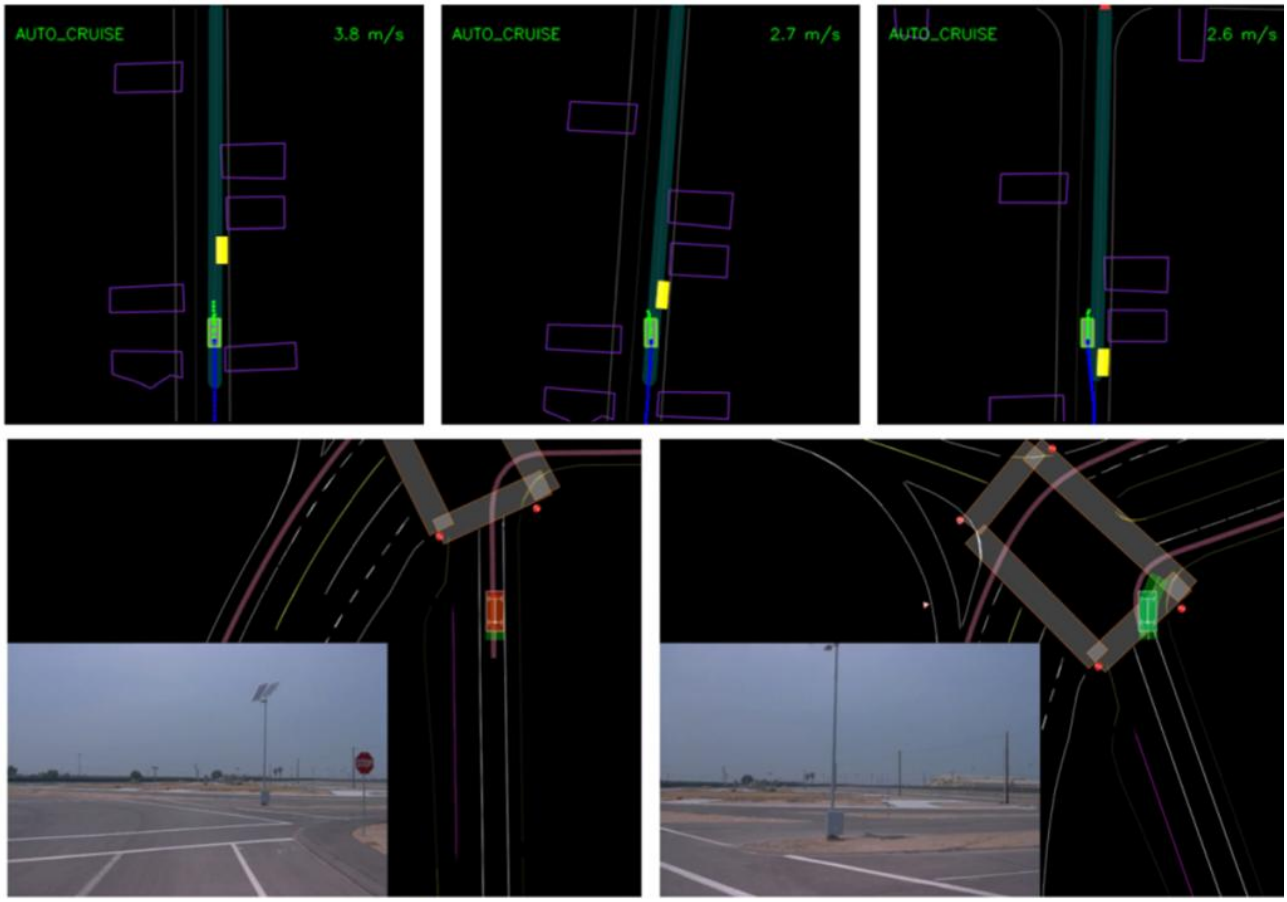


Figure 5.5: Object detection in ChauffeurNet

In the image above, the cyan path depicts the input route, green box is the self-driving car, blue dots are the agent's past route or position, and green dots are the predicted future routes or positions.

Essentially, a mid-level representation doesn't directly use raw sensor data as input, factoring out the perception task, so we can combine real and simulated data for easier transfer learning. This way, the network can create a high-level bird's eye view of the environment which ultimately yields better decisions. Nvidia also uses a Convolution Neural Network as a primary algorithm for its self-driving car. But unlike Tesla, it uses 3 cameras, one on each side and one at the front. See the Figure-5.6.

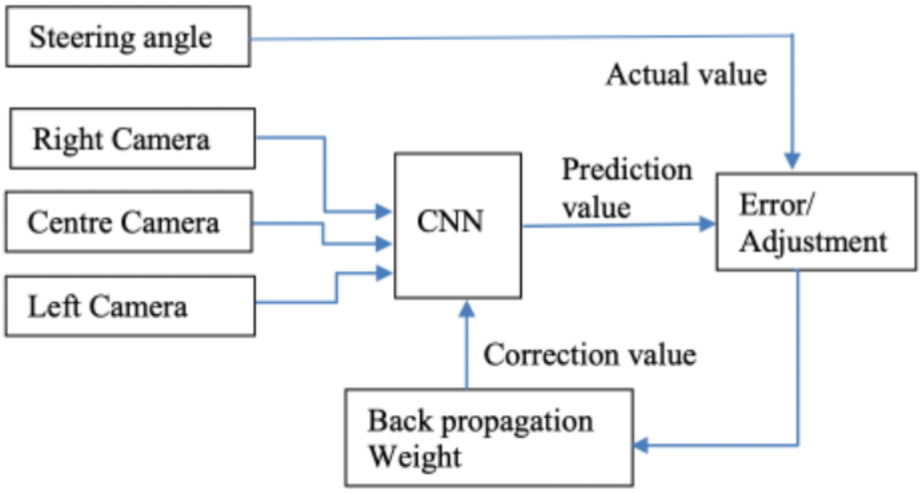


Figure 5.6: Layout of Waymo cars

The network is capable of operating inroads that don't have lane markings, including parking lots. It can also learn features and representations that are necessary for detecting useful road features.Compared to the explicit decomposition of the problem such as lane marking detection, path planning, and control, this end-to-end system optimizes all processing steps at the same time. Better performance is the result of internal components self-optimizing to maximize overall system performance, instead of optimizing human-selected intermediate criteria like lane detection. Such criteria understandably are selected for ease of human interpretation, which doesn't automatically guarantee maximum system performance. Smaller networks are possible because the system learns to solve the problem with a minimal number of processing steps.

5.2 Applications of Autonomous Vehicles:

The DARPA Urban challenge did provide an incentive in the quest towards a fully automated car. DARPA has been directly linked to successful development of autonomous vehicles by many research university teams as well as private sector. Commercial availability of a fully automatized vehicle appears to bejust around the corner. The last hurdles to commercial realization may be the necessary updates to road safety laws and, perhaps more difficult to define, the overcoming of individual and community fears to trust in the wheels of an automated machine on our roads. However, tests on existing automated cars are proving their reliability,with test outcomes better than human judgments made on the road. In the current marketplace, established car companies add autonomous parts when they comply with the law. Common examples of this emergence of automation in commercial cars include automated

parking, automatic correction of the car's trajectory if the driver crosses a continuous line, and alerts sounding to warn when other cars get too close.

5.3 Vehicle automation in different fields:

The industry of vehicle automation is much broader than commercial car automation. Freelance Robotics has worked on automation across many types of vehicles. For Instance, we have developed automation components for farming vehicles such as irrigators, tractors, and buggies, mining vehicles such as drilling rigs,and also industrial vehicles like forklifts and car crash testing vehicles.Civil engineering is an additional area of application, with successful robots having been developed for pipes inspection. In the wider context of the automated market, these applications are just a few examples. Given the potential variety and utility of applications, automated vehicles comprise a clear growth market.

GPS acquisition and processing:

When the environment is an open space, automated vehicles often use GPS technology to obtain their absolute position. As the antenna is generally at a certain height, and known to rock and tilt, we compensate for this difference of antenna position compared to the vehicle position on the ground using an additional sensor. This sensor is termed a Dynamic Measurement Unit (DMU) or Inertial Navigation Systems (INS) depending on the sensor output smartness. They consist of six to nine axis measurements generally composed of XYZ accelerometers, roll, pitch, yaw gyroscopes and XYZ magnetometers. A DMU commonly provides just the raw or noise filtered data, whereas INS provides a corrected output using Kalman filters to compensate for the defaults of each sensor, as some drift or have bias errors. An INS can also be loosely coupled to the GPS sensor position, velocity, and orientation data. Furthermore to improve performance the INS can be tightly coupled, merging the Doppler distance measurement from satellite triangulation with the less noisy data from INS to improve both GPS and INS accuracy. Also of interest, the Kalman filter can be used to filter the noisy heading from GPS using the heading data from the INS.

Another useful sensor frequently employed in vehicle automation is the Light Detection and Ranging (LIDAR) sensor. LIDAR allows distance to be mapped in the environment. This sensor operates independent from a GPS system, so can either be implemented with a GPS /INS system or by itself. Coupling LIDAR with a camera provides a powerful local positioning system. This local positioning system can be used to map the local environment and to identify landmarks. In this way, it is possible to know what is where remotely and also to locate the automated vehicle on the map. A commonly used technique to arrive at this data set is Simultaneous Localisation and Mapping (SLAM). As the name suggests, the technique corrects vehicle positioning from the LIDAR distance and map, creating shapes that can be recognized as a unique surrounding place on the map. The camera object/shape recognition may subsequently be applied as a complement to the map. The motor's encoder sensors are often used in conjunction with LIDAR to improve accuracy. When several sensors are involved,Particle filters can be useful. This statistical filtering approach operates on the assumption that automated vehicle location can be anywhere.Sensor data is merged from independent sources to converge upon a more accurate solution. The downfall of particle filters is that it becomes increasingly power hungry as the number of particles increase. Consequently, to process this type of algorithm in real-time,a powerful computer is required. Fortunately the necessary level ofcomputing power that can cope with an embedded system is becoming commonly available, making this technique of sensor data processing the current tool of choice for automation systems.

Another tool for automated vehicle localization is the use of landmarks. Landmarks can be natural or artificially added to the vehicle's environment. When natural, the vehicle identifies shapes that are supposed to be there, such as trees, barriers, posts, et cetera.Typically natural landmarks can be complicated to identify, decreasing system reliability. For example, a tree can change shape, or barriers appear different from different angles. In comparison, artificial landmarks provide an easy, consistent way for the automated vehicle to identify its position. Examples of suitable artificial landmarks include reflectors, RF IDS, bar codes, lines on the ground, panels with geometric shapes and colors, lights, Wi-Fi or IR beacons. When the vehicle has effectively identified positioning via landmarks, it can update and track its position over time. In this way, reliable correction of positioning on its map becomes feasible when the vehicle is a) in motion or b) moved in a known environment.

Once the automated vehicle has acquired knowledge of its position and orientation, it can use that information to drive. The vehicle will have a command in speed and bearing to move around. The initial objective is to make the automated vehicle control its motion along a specific AB line. To do so, a Proportional Integral Derivative (PID) is often used to

get the distance and orientation difference. Essentially the PID compares where the automated vehicle is against where it is supposed to be. The aim is to reduce this difference to the smallest amount possible. To do so, the automated vehicle applies a theoretical model to output a command. For example, to change vehicle curvature (angle of trajectory) a command may be given to the actuators that drive the vehicle. The PID modifies this command by boosting or damping the model reaction, which considers previous, present and predicted measurement difference to desired measurement. This process allows the automated vehicle to drive toward the AB line with both minimum overshoot and the quickest line acquisition the vehicle can managed. To optimize this line acquisition, tuning of the PID is required.

Once the automated vehicle controls motion along a line, the same principle can be easily extrapolated to control from one line to another. The vehicle can thereby create a path with any desired shape that the vehicle's model can conform to. This new line could be too small to be considered a curve; alternatively, Bezier curves may be used to create a path where curvature itself can be used to conform to a curved path. Thus, the vehicle can either follow a preregistered path entered by the user or determine its own path using decision rules.

Automated vehicles are required to navigate a number of path modifications for successful independent driving. One of the path modifications an automated vehicle often has to determine by itself is when an obstacle gets on the way of the original path. An obstacle could be stationary or in a collision path if it is in motion. To avoid the obstacle, the automated vehicle needs to take into consideration the vehicle model and the environment map it has to determine the shortest way to recover the initial path considering a minimum clearance to surrounding obstacles. To do so, decision algorithms are used to split the map into a grid. Decision algorithms operate with more or less efficiency, running tree exploration solutions, for example, may assist to evaluate which is the more or less optimal path to go around the obstacle.

6 FUTURE SCOPE

In the implementation of the project the deep neural network layers were used in sequential models. Use of parallel network of network layers to learn track specific behavior on separate branches can be a significant improvement towards the performance of the project. One of the branches can have CNN layers, the other with the RNN layers and combining the output with a dense layer at the end. There are similar problems that are solved using RESNET (Deep Residual networks) [8], a modular learning framework. RESNET are deeper than their 'plain' counterparts (state-of-art deep neural networks) yet require similar number of parameters (weights). Implementing Reinforcement Learning approaches for determining steering angles, throttle and brake can also be a great way of tackling such problems. Placing fake cars and obstacles on the tracks, would increase the level of challenges faced to solve this problem, however, it will take it much closer to the real-time environment that the self-driving cars would be facing in the real world. How well the model performs on real world data could be a good challenge. The model was tried with the real-world dataset, but there was no way of testing it on an environment like a simulator. The big players in the self-driving car industries must be already trying this on their autonomous vehicles. This would be a great experiment to see, how this model really works in the real time environment.

7 CONCLUSION

The Self-driving cars are the upcoming technology and will be the most successful in today's world with fuel prices hiking and pollution is increasing each day. The self-driving cars can help in not only cost-cutting but also reduce a lot of pollution from the surroundings as they are mostly hybrid or fully electric vehicles.

Self-driving cars aim to revolutionize car travel by making it safe and efficient. In this article, we outlined some of the key components such as LiDAR, RADAR, cameras, and most importantly – the algorithms that make self-driving cars possible.

While it's promising, there's still a lot of room for improvement

Few things need to be taken care of:

- The algorithms used are not yet optimal enough to perceive roads and lanes because some roads lack markings and other signs.
- The optimal sensing modality for localization, mapping, and perception still lack accuracy and efficiency.
- Vehicle-to-vehicle communication is still a dream, but work is being done in this area as well.
- The field of human-machine interaction is not explored enough, with many open, unsolved problems.
- Lidar is expensive and is still trying to strike the right balance between range and resolution.
- Weather conditions also have impact on performance of car. If there's a layer of snow on the road, lane dividers disappear.
- Level-5 car does not have a dashboard or a steering wheel, so a human passenger would not even have the option to take control of the vehicle in an emergency.

But the real promise of autonomous cars is the potential for dramatically lowering CO_2 emissions. In a recent study, experts identified three trends that, if adopted concurrently, would unleash the full potential of autonomous cars: vehicle automation, vehicle electrification, and ridesharing. By 2050, these "three revolutions in urban transportation" could:

- Reduce traffic congestion (30% fewer vehicles on the road)
- Cut transportation costs by 40% (in terms of vehicles, fuel, and infrastructure)
- Improve walkability and livability
- Free up parking lots for other uses (schools, parks, community centers)
- Reduce urban CO_2 emissions by 80% worldwide

Self driving cars have lots of benefits. Some of them are:

1. Prevention of car crashes

Of the 37,133 vehicle fatalities in 2017, 94% of the crashes were due to human error. Computers based on sophisticated systems and algorithms will essentially eliminate costly human error. Major causes of accidents, including drunk or distracted driving, will not be factors with self-driving cars. It's estimated self-driving cars can reduce accidents by up to 90%.

2. Societal cost-savings

One of the major factors when weighing the pros and cons of automated cars is the cost to society. Reports have shown that autonomous vehicles can help save society approximately $800 billion each year. The reduction in car crash-related costs, reduced strain on the healthcare system, more efficient transportation, better fuel savings, and more can all contribute to the overall societal cost-savings.

3. Traffic efficiency

One of the major benefits of self-driving cars is their ability to communicate with each other. With this ability to communicate in real-time, cars would be able to travel efficiently at optimized distances from each other. They'd also determine the best route for you to take, as to eliminate bumper-to-bumper traffic jams.

4. Better access and mode of transportation

For those who cannot or choose not to drive, self-driving cars could be a safe and reliable mode of transportation. Those with a disability or the elderly would be able to get into a self-driving car without putting others at risk.

Cities with limited public transit coverage would also benefit from self-driving cars. Self-driving cars can easily reach areas where infrastructure is lacking.

5. Environmentally friendly

Another significant factor in the self-driving cars pros and cons debate is the environment. Autonomous cars will likely be electric rather than utilizing internal-combustion engines. Furthermore, the consistent speeds self-driving cars will be traveling at will reduce constant braking and accelerating. These factors will all contribute to reducing emissions and becoming more environmentally sustainable.

8 REFERENCES:

[1] “ 10 Million Self-Driving Cars Will Hit The Road By 2020 -- Here’s How To Profit .” .

[2] W. Societ y for International Engineers, “SAE International Releases Updat ed Visual Chart for It s ‘Levels of Driving Aut omat ion’ Standard for Self-Driving Vehicles,” *SAE.org News*, 2018. .

[3] F. T orabi, G. Warnell, and P . St one, “ Behavioral cloning from observat ion,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2018, vol. 2018-July, pp. 4950–4957, doi: 10.24963/ijcai.2018/687.

[4] K. O’Shea and R. Nash, “ An Introduction to Convolut ional Neural Networks,” pp. 1–11, 2015.

[5] “ GitHub - udacity/self-driving-car-sim: A self-driving car simulator built wit h Unit y.” .

[6] “ OpenCV.” .

[7] H. Fujiyoshi, T . Hirakawa, and T . Yamashita, “Deep learning-based image recognit ion for aut onomous driving,” *IATSS Research*. Elsevier B.V., Dec. 2019, doi: 10.1016/j.iatssr.2019.11.008.

[8] S. Ren, K. He, R. Girshick, and J. Sun, “ Fast er R-CNN: Towards Real-T ime Object Detection wit h Region P roposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031. [9] R. Kulkarni, S. Dhavalikar, and S. Bangar, “Traffic Light Detection and Recognit ion for Self Driving Cars Using Deep Learning,” *Proc. - 2018 4th Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2018*, pp. 1–4, 2019, doi: 10.1109/ICCUBEA.2018.8697819.

[10] M. Bojarski *et al.*, “ End t o End Learning for Self-Driving Cars,” Apr. 2016.

[11] A. K. Jain, “Working model of Self-driving car using Convolutional Neural Network, Raspberry P i and Arduino,” in *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, Sep. 2018, pp. 1630–1635, doi: 10.1109/ICECA.2018.8474620.

[12] J. Kim, G. Lim, Y. Kim, B. Kim, and C. Bae, “ Deep Learning Algorithm using Virtual Environment Data for Self-driving Car,” in *1st International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2019*, Mar. 2019, pp. 444–448, doi: 10.1109/ICAIIC.2019.8669037. [13] Y. Kang, H. Yin, and C. Berger, “ T est Your Self –Driving Algorithm: An Overview of Publicly Available Driving Datasets and Virt ual T est ing Environments,” *IEEE Trans. Intell. Veh.*, vol. 4, no. 2, pp. 171–185, Mar. 2019, doi: 10.1109/t iv.2018.2886678.

[14] F. Yu *et al.*, " BDD100K: A Diverse Driving Video Dat abase with Scalable Annot at ion T ooling," May 2018. [15] G. J. Brost ow, J. Fauqueur, and R. Cipolla, " Semantic object classes in video: A high-definit ion ground t rut h dat abase," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, Jan. 2009, doi: 10.1016/j.pat rec.2008.04.005.

[16] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High –Fidelity Visual and Physical Simulat ion for Autonomous Vehicles," 2018, pp. 621–635.

[17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, " CARLA: An Open Urban Driving Simulat or," Nov. 2017.

[18] B. Wymann, C. Dimit rakakis, A. Sumner, E. Espié, and C. Guionneau, " T ORCS: T he open racing car simulat or," 2015.

[19] E. S. Gedrait e and M. Hadad, " Invest igat ion on t he effect of a Gaussian Blur in image filt ering and segmentation," *Proc. Elmar - Int. Symp. Electron. Mar.*, no. August , pp. 393–396, 2011.

[20] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Funct ions: Comparison of t rends in Practice and Research for Deep Learning," Nov. 2018.

[21] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," Nov. 2015.

[22] D. P . Kingma and J. L. Ba, " Adam: A met hod for st ochast ic opt imizat ion," 2015.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, " ImageNet: A large-scale hierarchical image dat abase," Mar. 2010, pp. 248–255, doi: 10.1109/cvpr.2009.5206848.

[24] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, May 2014, vol. 8693 LNCS, no. PART 5, pp. 740– 755, doi: 10.1007/978-3-319-10602-1_48.

[25] K. He, X. Zhang, S. Ren, and J. Sun, " Deep residual learning for image recognit ion," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, " Densely connected convolut ional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Nov. 2017, vol. 2017-January, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.

[27] K. Simonyan and A. Zisserman, " Very deep convolut ional networks for large-scale image recognit ion," 2015.

[28] M. P odpora, G. P aweł, K. Korba´s, and A. Kawala-Janik, " YUV vs RGB-Choosing a Color Space for Human-Machine Int eract ion," doi: 10.15439/2014F206.

[29] " Kaggle: Your Home for Data Science."

[30] The 6 Levels of Vehicle Autonomy Explained “https://www.synopsys.com/automotive/autonomous-driving-levels.html”