

AUTONOMOUS SELF DRIVING CAR

A mini-project report submitted in partial fulfillment of the Academic requirements for the award of the Degree of

BACHELOR OF ENGINEERING IN INFORMATION TECHNOLOGY

By

S HRUTURAJ

(2451-19-737-040)

G.OM PRAKASH

(2451-19-737-027)

H.KUSHAL

(2451-19-737-030)

Under the guidance of
Dr. D . Shanthi Professor,
Dept of IT



DEPARTMENT OF INFORMATION TECHNOLOGY
MATURI VENKATA SUBBA RAO (MVSR) ENGINEERING
COLLEGE

(An Autonomous Institution)

(Affiliated to Osmania University, Hyderabad. Recognized by AICTE)
Nadargul, Saroornagar Mandal, Hyderabad-501510

2022-2023

MATURI VENKATA SUBBA RAO (MVSRR)
ENGINEERING COLLEGE
(An Autonomous Institution)
(Affiliated to Osmania University, Hyderabad. Recognized by AICTE)
Nadargul, Saroornagar Mandal, Hyderabad-501510



DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the mini project work entitled “**AUTONOMOUS SELF DRIVING CAR**” is a bonafide work carried out by **Mr. S HRUTURAJ (2451-19-737-040)**, **Mr. G.OM PRAKASH (2451-19-737-027)**, **Mr. H.KUSHAL (2451-19-737-030)** in partial fulfillment of the requirements for the award of degree of **Bachelor of Engineering in Information Technology** from **Maturi Venkata Subba Rao (M.V.S.R.) Engineering College**, affiliated to **OSMANIA UNIVERSITY**, Hyderabad, during the Academic Year 2022-23. under our guidance and supervision.

The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Signature of Project Coordinator

Signature of Guide

Signature of Head, ITD

Signature of External Examiner

DECLARATION

This is to certify that the work reported in the present mini-project entitled “AUTONOMOUS SELF DRIVING CAR” is a record of bonafide work done by us in the Department of Information Technology, M.V.S.R. Engineering College, Osmania University. This report is based on the project work done entirely by us and not copied from any other source.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

<u>Roll Number</u>	<u>Student Name</u>	<u>Signature</u>
2451-19-737-040	S HRUTURAJ	
2451-19-737-027	G.OM PRAKASH	
2451-19-737-030	H.KUSHAL	

ACKNOWLEDGEMENT

We with extreme jubilation and deepest gratitude, would like to thank our guide, **Dr. D.Shanthi, Associate Professor**, Department of Information Technology, Maturi Venkata Subba Rao (MVSR) Engineering College, for her constant encouragement to us to complete our work in time.

With immense pleasure, we record our deep sense of gratitude to our beloved Head of the department **Dr.K.VenuGopal Rao** Dean-Academics & HOD, Department of Information Technology, Maturi Venkata Subba Rao Engineering College, for permitting and providing facilities to carry out this project.

We would like to extend our gratitude to **Dr. D.Shanthi**, Assoc.Prof. & **Project Coordinator**, **K.Devaki** and **V.Swarna Kamalam** **Section coordinators**, Department of Information Technology, Maturi Venkata Subba Rao Engineering College, for their valuable suggestions and timely help during the course of the project.

Finally, we express, from the bottom of our heart and deepest gratitude to the entire faculty, my parents and family for the support, dedication, comprehension and love.

S HRUTURAJ(2451-19-737-040)
OM PRAKASH(2451-19-737-027)
H.KUSHAL(2451-19-737-030)

MVSR Engineering College
Department of Information Technology

COURSE NAME: PROJECT WORK I

COURSE CODE: PW 653 IT

VISION

To impart technical education to produce competent and socially responsible engineers in the field

of Information Technology.

MISSION

M1. To make the teaching-learning process effective and stimulating.

M2. To provide adequate fundamental knowledge of sciences and Information Technology with positive attitude.

M3. To create an environment that enhances skills and technologies required for industry.

M4. To encourage creativity and innovation for solving real world problems.

M5. To cultivate professional ethics in students and inculcate a sense of responsibility towards society

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Bachelor's program in Information Technology is aimed at preparing graduates who will:

I. Apply knowledge of mathematics and Information Technology to analyze, design and implement solutions for real world problems in core or in multidisciplinary areas.

II. Communicate effectively, work in a team, practice professional ethics and apply knowledge

of computing technologies for societal development.

III. Engage in Professional development or postgraduate education to be a life-long learner.

PROGRAM OUTCOMES (POs)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and

write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of

the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOS):

(1) Hardware design: An ability to analyze, design, simulate and implement computer hardware/software and use basic analog/digital circuits, VLSI design for various computing and communication system applications.

(2) Software design: An ability to analyze a problem, design algorithm, identify and define the

computing requirements appropriate to its solution and implement the same.

COURSE OBJECTIVES:

1. To enhance practical & Professional skills.

2. To familiarize the tools and techniques of symmetric literature survey and documentation.

3. To expose students to industry practices and teamwork.

4. To encourage students to work with innovative and entrepreneurial ideas.

COURSE OUTCOMES:

On successful completion of this course students will be able to:

1. Define a problem of the recent advancements with applications towards society.

2. Outline requirements and perform requirement analysis for solving the problem.

3. Design and develop a software and/or hardware-based solution within the scope of project using contemporary technologies and tools.

4. Test and deploy the applications for use.

5. Develop the Project as a team and demonstrate the application, with effective written and oral communications.

ABSTRACT

Autonomous driving has developed rapidly over the last few years. Predicting the steering angle for a self-driving car according to different road conditions is very important. There are some endeavors for this topic, including lane detection, object detection on roads, 3-D reconstruction etc., but in our work, we focus on a vision-based model that directly maps raw input images to steering angles using deep networks and this model doesn't depend on specifying the features to learn. In this project, we propose an end-to-end steering angle prediction model based on deep transfer learning and it can accurately predict steering angles based on input image sequences which are from an onboard camera. This prediction model is based on the Behavioral Cloning Technique and other techniques which include Deep Neural Networks, Feature Extraction with Convolution Neural Networks as well as Continuous Regression. The CNN model we use is based on transfer learning techniques, and Image manipulation with good performance. This network is used to extract spatial features of the input image sequences. The model is used to fit the nonlinear relationship well between the input images and the steering angles. In order to train and validate the proposed model, the experimental study is conducted using the Udacity Self-driving Car simulator. We create our own training data for our model by driving a car on the training track inside the simulator recording the images at each instance of drive representing our dataset and evaluating its performance on a completely different testing track. Experimental results show that the proposed model in this project can efficiently predict the steering angles and clone humans' driving behaviors, and our model has better performance, higher accuracy, and less training time.

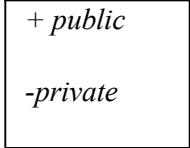
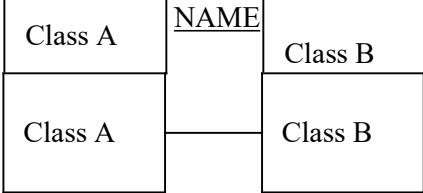
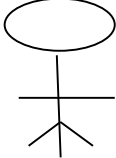
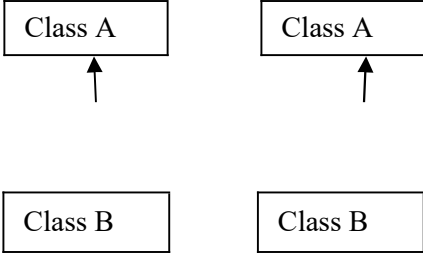
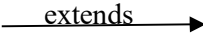
LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO.
4.1	System Architecture	9
4.1.2	Flow of the project	10
4.2.1	Use Case Diagram	11
4.2.2	Class Diagram	12
4.2.3	Sequence Diagram	12
4.2.4	Activity Diagram	13
5.1	Python installation	14
5.1.1	Downloading visual code	15
5.1.2	Visualcode installation finished	16
5.1.3	Downloading udacity simulator	17
5.2.1	Final neural network for predicting steering angles	20
6.1	Establishing connection between simulator and CNN	27
6.2	Running autonomously on lake track	27
6.3	Running autonomously on jungle track	28

LIST OF TABLES

TABLE NO	NAME OF THE TABLE	PAGE NO.
2.1	LITERATURE SURVEY	7

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Association represents static relationships between classes. Role represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single class.
4.	Aggregation		Interaction between the system and external environment
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.



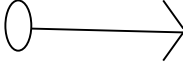
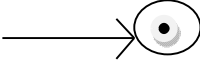
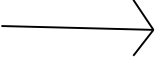
7.	Communication		Communication between various use cases.
8.	State		State of the processes.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states

TABLE OF CONTENTS

<u>CONTENTS</u>	<u>PAGE NO.</u>
TITLE	i
CERTIFICATE	ii
DECLARATION	iii
ACKNOWLEDGEMENT	iv
VISION	v
MISSION	vi
PROGRAM OUTCOMES	vii
COURSEOBJECTIVES	viii
COURSE OUTCOMES	ix
ABSTRACT	x
LIST OF FIGURES	xi
LIST OF TABLES	xii
LIST OF SYMBOLS	xiii
TABLE OF CONTENTS	xiv

1.	CHAPTER 1: INTRODUCTION	2
2.	1.1 PROBLEM STATEMENT	3
3.	1.2 OBJECTIVES	3
4.	1.3 MOTIVATION	3
5.	1.4 EXISTING SYSTEM	4
6.	1.5 PROPOSED SYSTEM	5
7.	1.6 SCOPE	5
8.	CHAPTER 2: LITERATURE SURVEY	7
9.	CHAPTER 3: SYSTEM REQUIREMENT SPECIFICATION	8
10.	3.1 SOFTWARE REQUIREMENTS	8
11.	3.2 HARDWARE REQUIREMENTS	8
12.	CHAPTER 4: SYSTEM DESIGN	9
13.	4.1 SYSTEM ARCHITECTURE	9
14.	4.1.1 ARCHITECTURE DESCRIPTION	9
15.	4.1.2 FLOW OF THE PROJECT	10
16.	4.2 UML DIAGRAMS	10
	a. 4.2.1 USE CASE DIAGRAM	11
	b. 4.2.2 CLASS DIAGRAM	12
	c. 4.2.3 SEQUENCE DIAGRAM	12
	d. 4.2.4 ACTIVITY DIAGRAM	13
17.	CHAPTER 5: IMPLEMENTATION	14
18.	5.1 ENVIRONMENTAL SETUP	14
19.	5.2 ALGORITHM	18
20.	5.3 MODULE DESCRIPTION	22
21.	CHAPTER 6: RESULTS	27
22.	CHAPTER 7: TESTS	23
23.	7.1 TESTING	23
24.	7.1.1 UNIT TESTING	23
25.	7.1.2 INTEGRATION TESTING	23
26.	7.1.3 ACCEPTANCE TESTING	30
27.	7.2 TEST CASES	30
28.	CHAPTER 8: CONCLUSION AND FUTURE ENHANCEMENTS	31
29.	REFERENCES	32
30.	APPENDIX	33

CHAPTER 1

INTRODUCTION

Automated vehicles have been generating significant attention and discussion, recently with almost every automobile company trying to develop their respective autonomous vehicle concept and are successful in achieving some levels of autonomy and are planning to start production of driverless vehicles in few years. Going by means of vehicle is at present one of the most perilous kinds of transportation with over a million passing every year around the world. As nearly all car crashes (particularly fatal ones) are caused by driver error, driverless vehicles would viably dispose of about all dangers related to driving just as driver fatalities, road safety, mobility for everyone, and injuries. The self-driving vehicle is the dormant beast that can make a huge difference. The self driving vehicle is a vehicle outfitted with an autopilot framework and is equipped for driving without the help of human administrator. This innovation was fabricated initially using autonomy approach yet with the progress in the field of PC vision and AI we can utilize the deep learning approach. There are fundamentally two approaches to tackle this problem viz. the old school robotics-based approach and the modern learning-based approach.

The traditional robotics-based approach splits the autonomous driving task into subsequent modules, namely perception, planning and control. Although this somewhat simplifies the overall process, precise implementation of these individual fragments is an arduous task in itself. Furthermore, the complex algorithms pertaining to perception, planning and control are computationally expensive and often struggle from real-time processing; not to mention they are generally scenario-specific and need to be retuned before being deployed to handle dissimilar situations. With the advent of machine learning strategies, some of the aspects of robotics approach were substituted using alternative data driven algorithms. Particularly, convolutional neural networks (CNNs) completely revolutionized the way perception stack was implemented. Nonetheless, the notion of end-to-end learning truly turned the tables by defining the entire task of autonomous driving as a machine learning problem. End-to-end learning, in the context of autonomous driving, allows an agent to directly map the perception data to the appropriate actuator commands using neural network as a non-linear function approximator. This eliminates the need of designing and implementing specialized perception, planning and control stacks, which not only

simplifies the development phase but also gives an added advantage of real-time computation during the deployment phase.

1.1 PROBLEM STATEMENT

To propose a trained CNN model which can autonomously drive on new tracks. This model is implemented using the behavioral cloning technique by learning from our behavior as manual driver. The pretrained model then can adjust the steering angle to an appropriate degree based on the situation that it finds itself in. The model can adapt to new tracks and different road conditions. The performance of model will be good to handle the car at sharp turns and hilly regions.

1.2 OBJECTIVES

The objective of this project:

- To identify the track and run autonomously.
- To maintain speed accordingly.
- To handle the car well at sharp turns.
- To predict the steering angle based on the image given.

1.3 MOTIVATION

The self-driving car is made up of numerous sensors, such as LiDAR and RADAR systems, working concurrently to carry out operations automatically without the help of drivers. By performing situational analysis, motion planning, and trajectory control, these sensors help in the process of navigation. There is an increasing demand for self-driving cars for road safety measures. The number of road accidents is increasing daily; the

primary cause of accidents is human error that occurs in the uncertainty of events, for instance, head-on collisions due to misinterpretation of the driver. The self-driving vehicle is driven by a computer. Thus, it makes fewer errors than humans, such as accidents due to distracted driving, alcohol, not wearing seatbelts, speeding, and drowsy driving. Self-driving vehicles are machines that optimally perform the task of driving and aren't tired or bored, or wouldn't be distracted. Thus, the traveler is safer in an autonomous vehicle than in a human-driven car.

1.4 EXISTING SYSTEM

Throughout the past decade, we have witnessed one of the greatest strides in automobile technology with the focus on autonomous cars. Advancements have accelerated with the participation of the leading car developers including BMW, Ford, Audi, Google and Tesla. Self-driving cars combine a variety of sensors to perceive their surroundings, such as cameras, radar, lidar, sonar, GPS. For example, Cameras will receive visual information to help the car's software determine traffic signals and cross traffic warning. Ultrasound will be used to inform about the immediate environment for parking assistance. Each of these technologies will collect different information for the vehicle, and based on the information collected the car will make interpretations and make the appropriate response for that situation. Autonomy in vehicles is often categorized in six levels, according to a system developed by SAE International. The SAE levels can be roughly understood as Level 0 - no automation; Level 1 - hands on/shared control; Level 2 - hands off; Level 3 - eyes off; Level 4 - mind off, and Level 5 - steering wheel optional. At the moment, fully autonomous vehicles don't exist, and though some automakers and technology companies are getting closer and closer to coming out with fully autonomous vehicles. The main two technologies are lidar vs camera. Almost every single company working on self-driving cars right now uses LIDAR. Uber, Waymo, and Toyota all use it, but not Tesla. Waymo the industry leader in self-driving cars. Waymo's cars, which are fairly typical of other self-driving cars, use high-resolution cameras and lidar (light detection and ranging), a way of estimating distances to objects by bouncing light and sound off things. Tesla has been heavily relying on Vision and going against LIDAR sensors. At the same time, all the other companies use Lidar.

DRAWBACKS OF EXISTING SYSTEM

- Expensive setup of LIDAR sensors which also increases the weight of car decreasing the overall efficiency.
- Existing Systems are being used only in ideal conditions with limited miles and rely heavily on high-accuracy maps with road lines.
- LIDAR is not able to detect how they are moving or even what those objects are.

1.5 PROPOSED SYSTEM

We are developing a model which will focus on a very useful technique called Behavioral Cloning. This model is implemented using other techniques such as deep neural network, feature extraction with convolutional neural networks as well as continuous Regression. In summary we are essentially going to be downloading a self-driving car simulator provided to us Open-Source by Udacity. We are then going to use the simulator to create our very own training data for our model by driving a car through the training track inside the simulator as we drive the car through the simulator. We are going to be taking images at each instance of the drive. These images are going to represent our training dataset. The dataset is then showed to our CNN model and allow it to learn how to drive autonomously. We are going to evaluate its performance on a completely different testing track where the car will be made to run autonomously if we are able to train the car properly it will perform very well on our second track and will drive on its own. This behavioral cloning technique is incredibly useful and plays a big role in real life self-driving cars as well.

1.6 SCOPE

- The scope of this project is unlimited and are increasingly present in modern society. The industry of vehicle automation is much broader than commercial car automation. They 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.
- The advantages of a change like this are significant for both drivers and individuals in need of the cargo being transported. Widely known challenges of trucking include the long hours and risk involved in such a crucial position. Autonomous and self-driving trucks can help with these challenges and more.
- The Lane and attitude determination could also benefit from GNSS if the accuracy is good enough. If the location information is shared among cars, GNSS could be a part of a short-range situation awareness system (awareness of other vehicles in the road and collision avoidance) although it is not expected that GNSS is the sole means of information for short-range situation awareness.

CHAPTER 2**LITERATURE SURVEY**

S.NO	AUTHOR NAME	PROBLEM IDENTIFIED	TECHNIQUES USED	ACCURACY	DRAWBACKS
1.	Ali sagarhaji, Priyam shah, B.Srinivas	Making autonomous cars using deep learning	Behavioral cloning	84.5	Cannot deal with dynamically changing scenarios
2.	Tanmay vilas, Chinmay vilas, shivanathan	Behavioral cloning of human driver	End To End Imitation Learning	86.3	Not immune to corner cases and turns
3.	Mariusz Bojarski, Prasoon Goyal	Training a CNN to map to direct steering commands	End To End Learning	85.8	Fails at unpaved roads and forest areas

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

- CPU : i5 9th or 10th Gen.
- GPU : 8 GB
- RAM : 4GB 1050Ti

3.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 7/8/10
- Text Editor : VS Code Editor
- PYHTON : 3.7+
- ANACONDA : Latest Version imported with required libraries.
- UDACITY : 64-bit

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

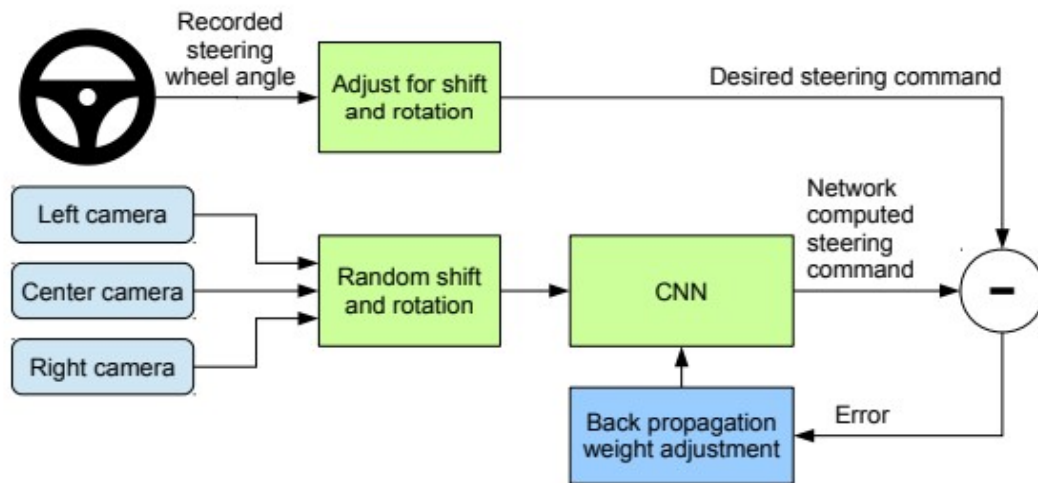


Fig-4.1 SYSTEM ARCHITECTURE

4.1.1 Architecture Description

- In the first step the Udacity simulator will capture images from the left camera, center camera, right camera..
- The captured image will be pre-processed.
- The pre-processed images are then given to the CNN.
- Network computes the steering angle and send command to the simulator.
- Error in computation of the steering angle will alert the network to adjust its weight through back propagation.

4.1.2 Flow Of The Project

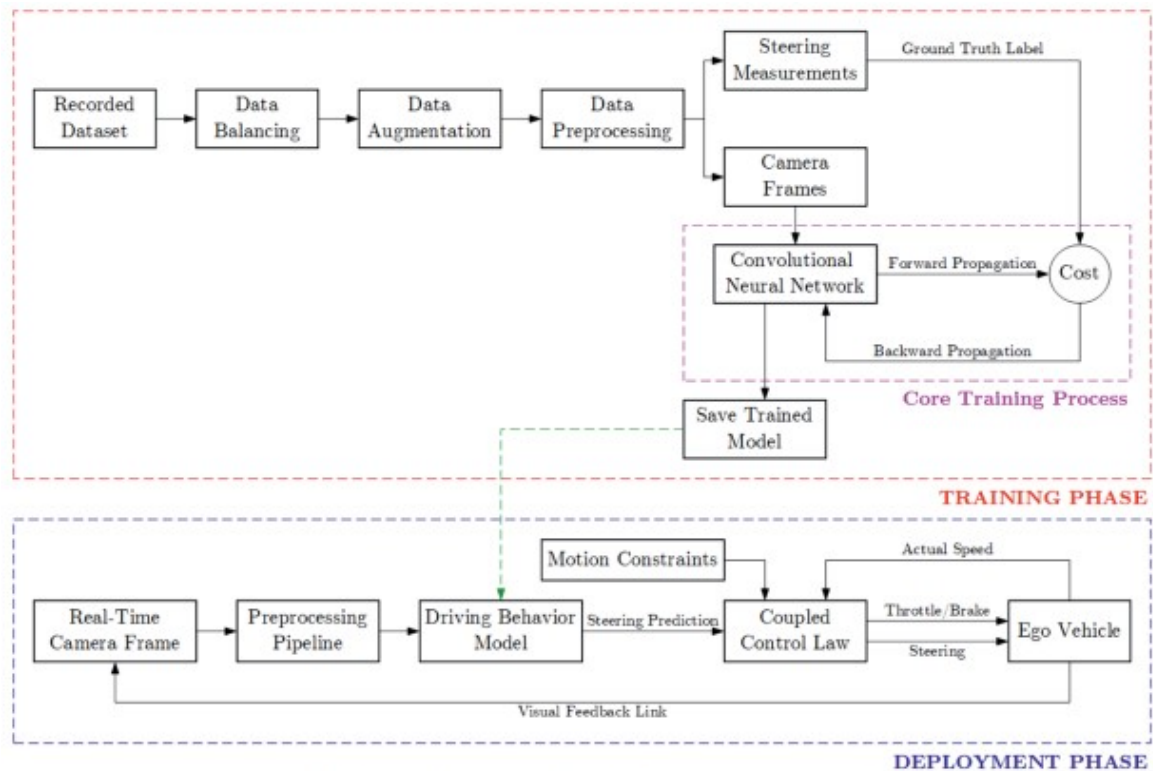


Fig-4.1.2 FLOW OF THE PROJECT

4.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

4.2.1 Use Case Diagram

- A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

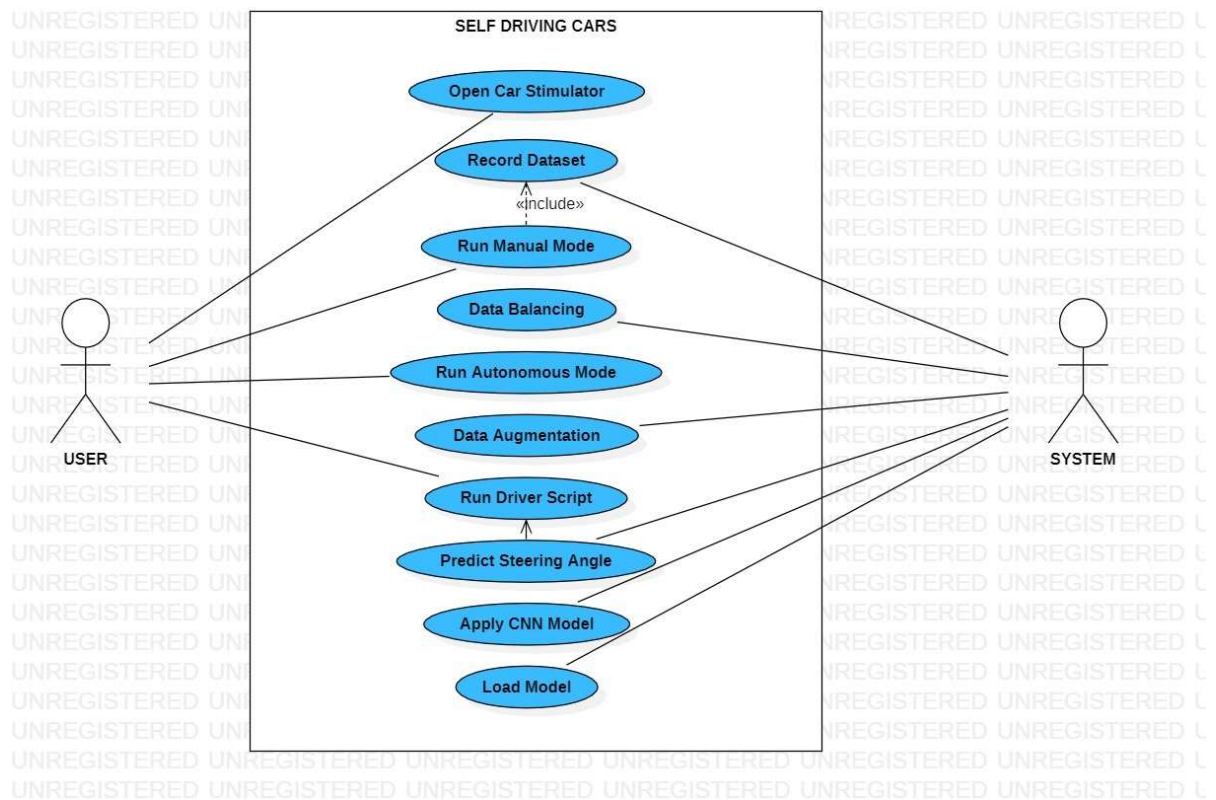


Fig-4.2.1 USE CASE DIAGRAM

4.2.2 Class Diagram

- In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by

showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains which information.

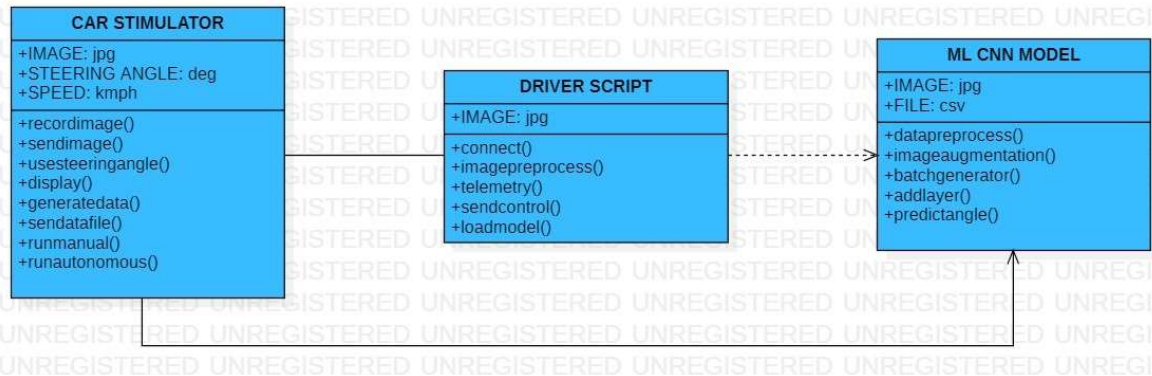


Fig-4.2.2 CLASS DIAGRAM

4.2.3 Sequence Diagram

- A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

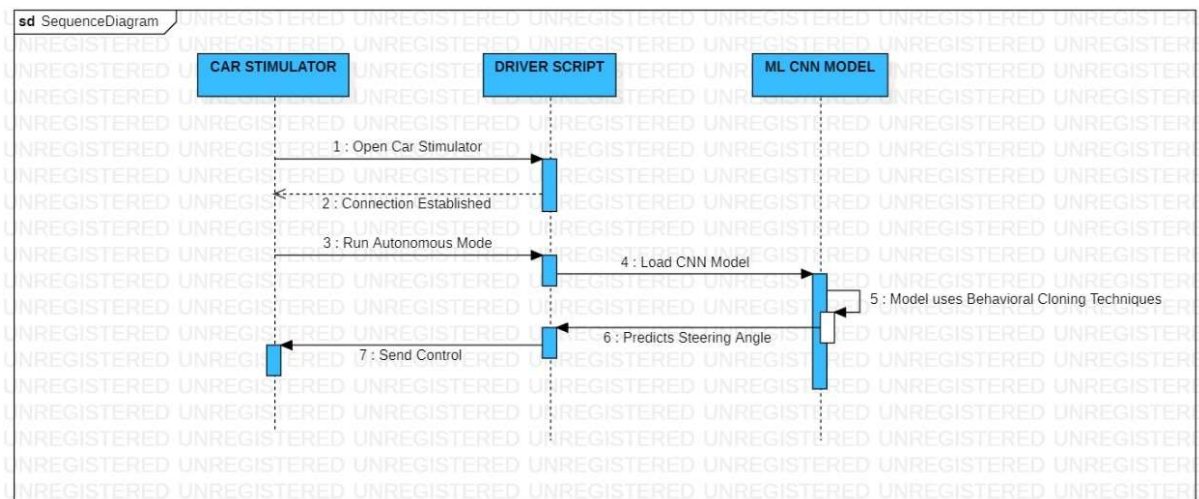


Fig-4.2.3 SEQUENCE DIAGRAM

4.2.4 Activity Diagram

- Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

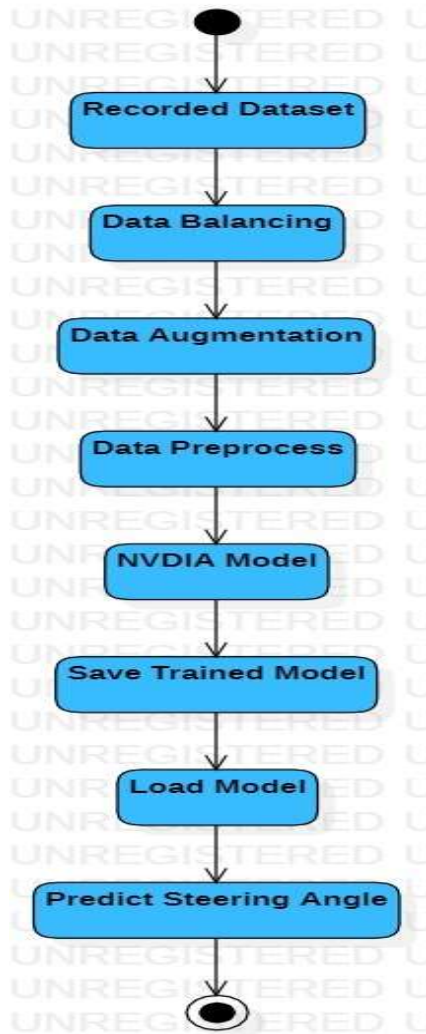


Fig-4.2.6 ACTIVITY DIAGRAM

CHAPTER 5

IMPLEMENTATION

5.1 ENVIRONMENTAL SETUP

Installing Anaconda:

1. Go to the Anaconda Website and choose a Python 3.x graphical installer.
2. Locate your download and double click it

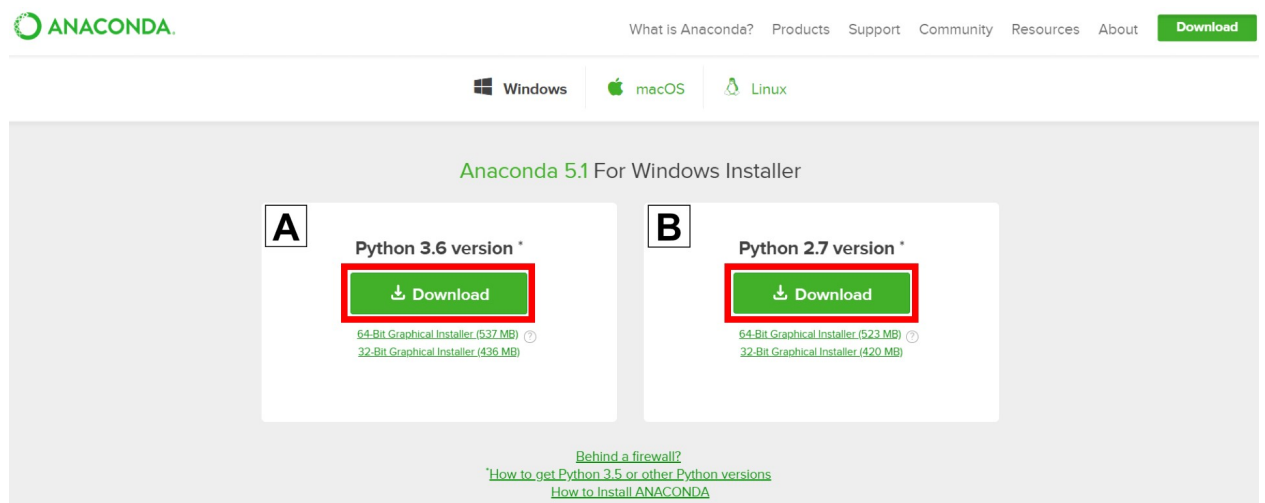


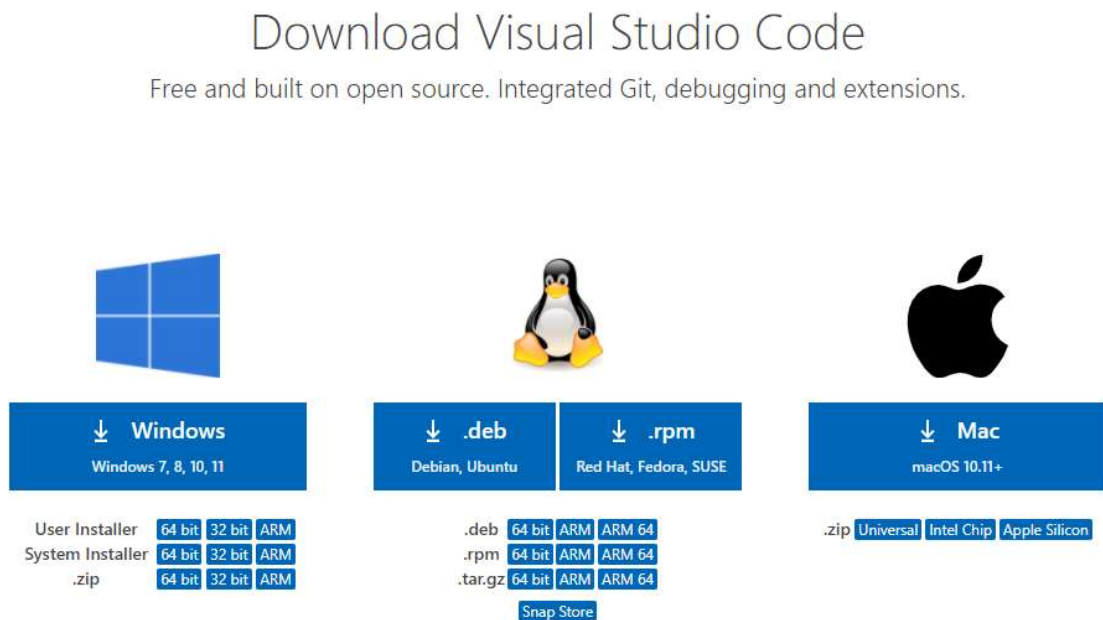
FIG-5.1 PYTHON INSTALLATION

3. Begin with the installation process:
 - Getting through the License Agreement
 - Select Installation Type: Select Just Me if you want the software to be used by a single User
 - Choose Installation Location
 - Finishing up the Installation
4. Open Anaconda Command Prompt. Try typing `conda --version` and `python --version` into the Command Prompt to check to see if everything went well.

Installing Visual Studio :

Step 1: Visit the official website of the Visual Studio Code using any web browser like Google Chrome, Microsoft Edge, etc.

Visit-the-official-website-of-the-Visual-Studio-Code



Step 2: Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application.

Press-the-Download-for-Windows-button

Step 3: When the download finishes, then the Visual Studio Code icon appears in the downloads folder.

Visual-Studio-Code-Installer

Step 4: Click on the installer icon to start the installation process of the Visual Studio Code.

Step 5: After the Installer opens, it will ask you for accepting the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.

Step 6: Choose the location data for running the Visual Studio Code. It will then ask you

for browsing the location. Then click on Next button.

Option-for-Visual-Studio-Code

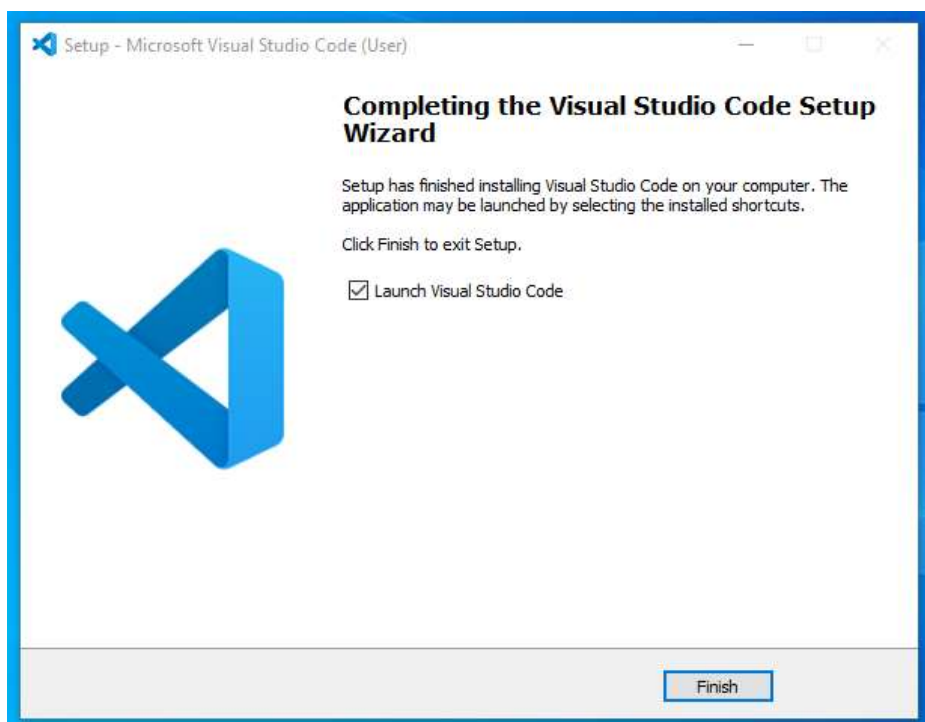
Step 7: Then it will ask for beginning the installing setup. Click on the Install button.

Click-on-the-Install-button

Step 8: After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.

Installing-Visual-Studio-Code

Step 9: After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the “Launch Visual Studio Code” checkbox and then click Next.

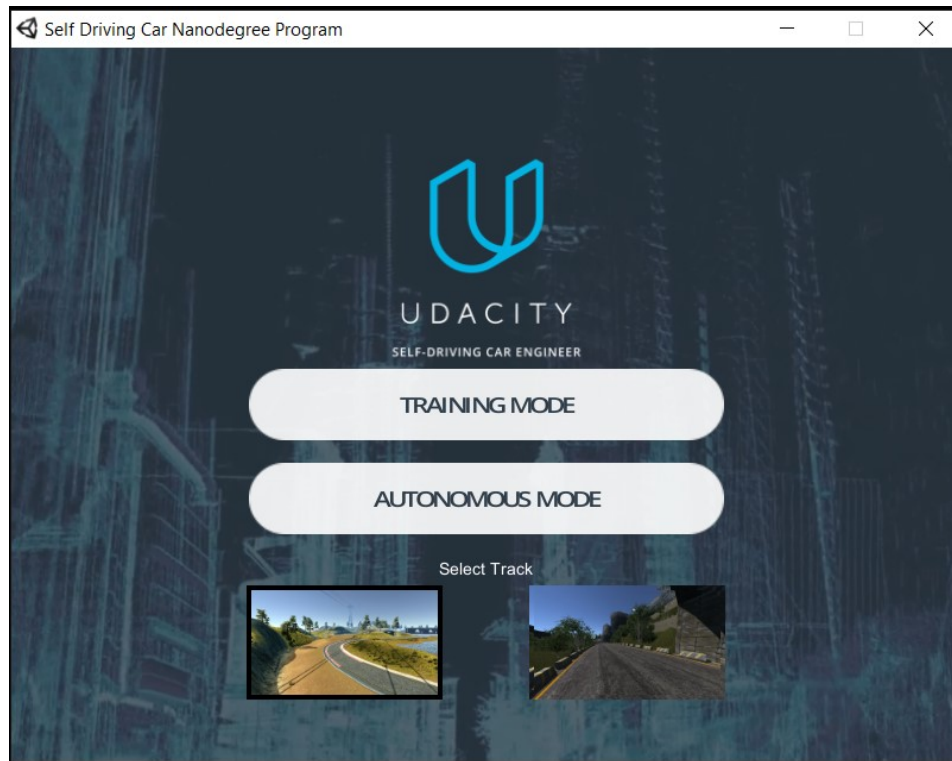


Visual-Studio-Code-installation-is-finished

Step 10: After the previous step, the Visual Studio Code window opens successfully.

Installing Udacity simulator:

1. Download the zip file, extract it and run the executable file.
2. Install the free game making engine Unity, if you don't already have it. Unity is necessary to load all the assets.



1. System

1.1 Record Dataset:

The dataset which consists of images from left, right, center images taken from car point of view, along with the steering angle.

1.2 Image Preprocessing:

- Images are resized and preprocessed for feature extraction.
- Data Balancing is applied to remove unnecessary data from all the collected datasets were heavily unbalanced towards zero-steering owing to the fact that the steering angle was reset to zero whenever the control keys were released.
- Data Augmentation of images are required to enhance convergence, minimize overfitting and speed up the training process. In this work, a total of five

augmentation techniques, zoom, pan, brightness, flip were applied to the dataset during the training phase.

- CNN model is built and steering angles are predicted using the build CNN model which are sent to the car simulator.

2.User:

- Open the Car simulator provided by the open source software Udacity.
- User Runs the Manual mode to record the data set and provide the dataset for the training phase.
- Runs Autonomous Mode after the connection is established between the Udacity car simulator and model.

- **2.1 View Results:**

The car runs autonomously on both of the tracks i.e Lake track and Jungle rack. Lake track is used for training where as Jungle tracks is a new track for the pretrained model.

5.2 ALGORITHM

The implementation of proposed pipeline can be divided into two phases, viz. training phase and deployment phase.

1. Training Phase

Training phase comprised of data collection, balancing, augmentation, preprocessing and training neural network.

1.1. Data Collection

The simulator has three cameras: a center, right and left camera. To capture good driving behavior, I recorded three laps on track 1 using center lane driving. In the training stage, I use all three cameras as training inputs. This is because we need to handle the issue of recovering from being off-center driving. The datasets for simplistic and rigorous driving behaviors included timestamped frames from the center, left and right cameras onboard the vehicle and the normalized steering angle measurement corresponding to each timestamp.

1.2. Data Segregation

The collected datasets were randomly split into training and validation subsets in the ratio of 4:1 (i.e. 80% training data and 20% validation data). The random state of splitting each dataset was chosen specifically such that the training and validation dataset would have minimal variation w.r.t. the steering measurements.

1.3. Data Balancing

The original training datasets were skewed towards either left or right steering since the ego vehicle traversed the track in a single direction. Additionally, all the collected datasets were heavily unbalanced towards zero-steering owing to the fact that the steering angle was reset to zero whenever the control keys were released. In order to minimize these unbalances, the dataset was balanced by deletion of random portion of the dataset containing exactly zero steering angle measurements above the threshold of 400.

1.4. Data Augmentation

Preprocessing and Augmentation of images are required to enhance convergence, minimize overfitting and speed up the training process. The image processing pipeline was implemented in python using Computer Vision Library. In this work, a total of five augmentation techniques, zoom, pan, brightness, flip were applied to the dataset during the training phase.

1.5. Data Preprocessing

Data preprocessing was aimed at faster and efficient training as well as deployment. This work describes a two step preprocessing function, which performs resizing and normalization (with mean centering) operations on the input images.

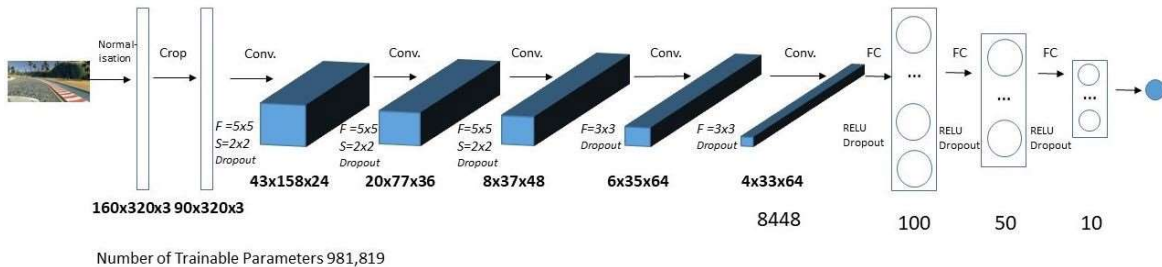
1.6. Training Model

The entire network is implemented and trained using Keras and Tensor flow as Backend .The Neural Network Architecture i used, is based on the [paper](#) published by NVIDIA, developed to map the raw pixels to steering commands. The network consists of 9 layers, including a normalization layer, 5 convolution layers and 3 fully connected layers.

The Image is normalized in Normalization layer. According to the [Paper](#), performing normalization in the network allows the normalization scheme to be altered with the network architecture and to be accelerated via GPU processing.

There are 5 convolution layers designed to extract the features from the images. The first 3 Convolutions are performed with strided convolutions with 2x2 strides and a 5x5 Kernel. The last two convolution layers are obtained through non-strided convolutions performed with 3x3 kernel.

The convolution layers are followed by 3 full connected layers leading to the output steering value. The model is implemented in Keras with Tensorflow as backend(*line 212:237*). My final model architecture is shown in the image below.



Final neural network for predicting steering angles

Adam Optimizer was used to update the network weights. This optimizer required no tuning as it adapts the learning rate during training.

3. Convolutional Neural Network

Step1: Convolutional operation

The first building block in our plan of attack is convolution operation. In this step, we will touch on feature detectors, which basically serve as the neural network's filters. We will also discuss feature maps, learning the parameters of such maps, how patterns are detected,

the layers of detection, and how the findings are mapped out.

Step (1b): Relu Layer

The second part of this step will involve the Rectified Linear Unit or ReLU. We will cover ReLU layers and explore how linearity functions in the context of Convolutional Neural Networks.

Not necessary for understanding CNN's, but there's no harm in a quick lesson to improve your skills.

Step 2: Pooling Layer

In this part, we'll cover pooling and will get to understand exactly how it generally works. Our focus here, however, will be a specific type of pooling; max pooling. We'll cover various approaches, though, including mean (or sum) pooling. This part will end with a demonstration made using a visual interactive tool that will definitely sort the whole concept out for you.

Step 3: Flattening

This will be a brief breakdown of the flattening process and how we move from pooled to flattened layers when working with Convolutional Neural Networks.

Step 4: Full Connection

In this part, everything that we covered throughout the section will be merged together. By learning this, you'll get to envision a fuller picture of how Convolutional Neural Networks operate and how the "neurons" that are finally produced learn the classification of images.

Summary

In the end, we'll wrap everything up and give a quick recap of the concept covered in the section. If you feel like it will do you any benefit (and it probably will), you should check out the extra tutorial in which Softmax and Cross-Entropy are covered. It's not mandatory for the course, but you will likely come across these concepts when working with Convolutional Neural Networks and it will do you a lot of good to be familiar with them.

5.3 MODULE DISCRIPTION

- OpenCV
- Numpy
- VS Code

OpenCV:

OpenCV is a library of programming capacities mostly focused on ongoing PC vision. Initially created by Intel, it was later bolstered by Willow Garage then Itseez . The library is cross-stage and free for use under the open-source BSD permit.

OpenCV underpins a few models from profound learning structures like TensorFlow, Torch, PyTorch (in the wake of changing over to an ONNX model) and Caffè as indicated by a characterized rundown of upheld layers. It advances OpenVisionCapsules. , which is a versatile configuration, perfect with every other organization. Authoritatively propelled in 1999 the OpenCV venture was at first an Intel Research activity to propel CPU-concentrated applications, some portion of a progression of undertakings including constant beam following and 3D show dividers. The principle supporters of the undertaking remembered various improvement specialists for Intel Russia, just as Intel's Performance Library Team.

OpenCV is written in C++ and its essential interface is in C++, yet it despite everything holds a less far reaching however broad more seasoned C interface. There are ties in Python, Java and MATLAB/OCTAVE. Since variant 3.4, OpenCV.js is a JavaScript official for chose subset of OpenCV capacities for the web stage.

The entirety of the new turns of events and calculations in OpenCV are presently evolved in the C++ interface.

OpenCV runs on the accompanying work area working frameworks: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. OpenCV runs on the accompanying portable working frameworks: Android, iOS, Maemo, BlackBerry 10. The client can get official discharges from SourceForge or take the most recent sources from GitHub. OpenCV utilizes CMake.

Advantages:

- OpenCV is accessible liberated from cost.

- As OpenCV library is written in C/C++ it is very quick.
- Low RAM use .
- It is versatile as OpenCV can run on any gadget which runs on C.

As a human, handling pictures is a characteristic procedure as we cooperate with numerous individuals and it is simple for us to perceive in our day by day lives. PCs, then again, need to follow by their own interesting procedures so as to break down huge measures of media information. All together for profound learning PC vision to flourish, thousands on a great many photographs, recordings, and different pictures should be incorporated for a viable AI to get valuable.

The PC arranges a picture dependent on pixel esteems, it isolates a picture into an enormous framework of boxes - also called pixels - and allots a number to each case.

- OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.
- The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding [18 million](#). The library is used extensively in companies, research groups and by governmental bodies.
- Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together,

detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

- It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, [Android](#) and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured [CUDA](#) and [OpenCL](#) interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

NumPy:

- NumPy is a Python library, including support for enormous, multi-dimensional clusters and frameworks, alongside a huge assortment of significant level numerical capacities to work on these exhibits. The predecessor of NumPy, Numeric, was initially made by Jim Hugunin with commitments from a few different designers. In 2005, Travis Oliphant made NumPy by consolidating highlights of the contending Numarray into Numeric, with broad alterations. NumPy is open-source programming and has numerous givers.
- NumPy focuses on the CPython reference execution of Python, which is a non-streamlining bytecode translator. Numerical calculations composed for this rendition of Python regularly run much more slow than assembled counterparts. NumPy addresses the gradualness issue incompletely by giving multidimensional clusters and capacities and administrators that work productively on exhibits, requiring changing some code, generally internal circles utilizing NumPy.
- Utilizing NumPy in Python gives usefulness practically identical to MATLAB since they are both deciphered, and the two of them permit the client to compose quick projects as long as most activities take a shot at exhibits or frameworks

rather than scalars. In examination, MATLAB brags a huge number extra tool compartments, remarkably Simulink, while NumPy is inherently coordinated with Python, an increasingly present day and complete programming language. Additionally, reciprocal Python bundles are accessible; SciPy is a library that includes more MATLAB-like usefulness and Matplotlib is a plotting bundle that gives MATLAB-like plotting usefulness. Inside, both MATLAB and NumPy depend on BLAS and LAPACK for effective direct variable based math calculations.

- Python ties of the broadly utilized PC vision library OpenCV use NumPy exhibits to store and work on information. Since pictures with various channels are essentially spoken to as three-dimensional clusters, ordering, cutting or concealing with different exhibits are exceptionally proficient approaches to get to explicit pixels of a picture. The NumPy cluster as widespread information structure in OpenCV for pictures, extricated include focuses, channel pieces and a lot more limitlessly streamlines the programming work process and troubleshooting.
- The center usefulness of NumPy is its "ndarray", for n-dimensional exhibit, information structure. Rather than Python's worked in list information structure, these clusters are homogeneously composed: all components of a solitary exhibit must be of a similar kind.
- Such clusters can likewise be sees into memory cradles distributed by C/C++, Cython, and Fortran expansions to the CPython mediator without the need to duplicate information around, giving a level of similarity with existing numerical libraries. This usefulness is misused by the SciPy bundle, which wraps various such libraries . NumPy has worked in help for memory-mapped ndarrays.

VSCode:

Visual Studio Code is a distribution of the Code – OSS repository with Microsoft-specific customizations released under a traditional Microsoft product license.

Visual Studio Code combines the simplicity of a code editor with what developers need for their core edit-build-debug cycle. It provides comprehensive code editing, navigation, and understanding support along with lightweight debugging, a rich extensibility model, and lightweight integration with existing tools.

Visual Studio Code is updated monthly with new features and bug fixes. You can download it for Windows, macOS, and Linux on Visual Studio Code's website. To get the latest releases every day, install the Insiders build.

Visual Studio Code, also commonly referred to as VS Code,[9] is a source-code editor made by Microsoft for Windows, Linux and macOS.[10] Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.



Fig:8.VSCode

CHAPTER 6

RESULTS

6.1 Establishing a connection between simulator and CNN model:

```

drive1.py X
drive1.py > img_preprocess
11 sio = socketio.Server()
12
13 app = Flask(__name__)
14 speed_limit = 10
15
16 def img_preprocess(img):
17     img = img[60:135, :, :]
18     img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
19     img = cv2.GaussianBlur(img, (3, 3), 0)
20     img = cv2.resize(img, (200, 66))
21     img = img/255
22     return img
23
24 @sio.on('telemetry')
25 def telemetry(sid, data):
26     speed = float(data['speed'])
27     image = Image.open(BytesIO(base64.b64decode(data['image'])))
28     image = np.asarray(image)
29     image = img_preprocess(image)
30     image = np.array(image)

```

PS E:\Autonomous Car> C:\Users\hrutu\conda\envs\model\python.exe "e:/Autonomous Car/drive1.py"

2022-07-19 09:59:04.688477: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2 To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

2022-07-19 09:59:06.219595: I tensorflow/core/common/runtime/gpu_device.cc:1532] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 2 145 MB memory: -> device: 0, name: NVIDIA GeForce GTX 1650, pci bus id: 0000:01:00:0, compute capability: 7.5

(14932) wsgi starting up on http://0.0.0.0:4567

(14932) accepted ('127.0.0.1', 63253)

Connected

2022-07-19 09:59:25.019466: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] loaded cuDNN version 8401

1/1 [=====] - 7s 7s/step

0.88477940309702757 1.0 0.0

1/1 [=====] - 0s 44ms/step

0.88477940309702757 1.0 0.0

1/1 [=====] - 0s 27ms/step

0.88477940309702757 0.9562 0.438

1/1 [=====] - 0s 34ms/step

0.8889366641640631 0.99999 0.8001

FIG-6.1 Connected

6.2 Running Autonomously on LAKE TRACK

```

drive1.py X
drive1.py > img_preprocess
11 sio = socketio.Server()
12
13 app = Flask(__name__)
14 speed_limit = 10
15
16 def img_preprocess(img):
17     img = img[60:135, :, :]
18     img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
19     img = cv2.GaussianBlur(img, (3, 3), 0)
20     img = cv2.resize(img, (200, 66))
21     img = img/255
22     return img
23
24 @sio.on('telemetry')
25 def telemetry(sid, data):
26     speed = float(data['speed'])
27     image = Image.open(BytesIO(base64.b64decode(data['image'])))
28     image = np.asarray(image)
29     image = img_preprocess(image)
30     image = np.array(image)

```

1/1 [=====] - 0s 50ms/step

-0.822046249359846115 0.07820000000000000 9.218

1/1 [=====] - 0s 47ms/step

-0.024634387344121933 0.07820000000000000 9.218

1/1 [=====] - 0s 44ms/step

-0.03900588676334274 0.07820000000000000 9.218

1/1 [=====] - 0s 48ms/step

-0.04847369584441185 0.07821 9.2179

1/1 [=====] - 0s 52ms/step

-0.044828858226537704 0.07821 9.2179

1/1 [=====] - 0s 31ms/step

-0.04996531829237938 0.07821 9.2179

1/1 [=====] - 0s 52ms/step

-0.052292224019765854 0.07822999999999999 9.2177

1/1 [=====] - 0s 42ms/step

-0.03951684758067131 0.07822999999999999 9.2177

1/1 [=====] - 0s 46ms/step

-0.029976408928632736 0.07824000000000000 9.2176

FIG-6.1 LAKE TRACK

6.2 Running Autonomously on JUNGLE TRACK

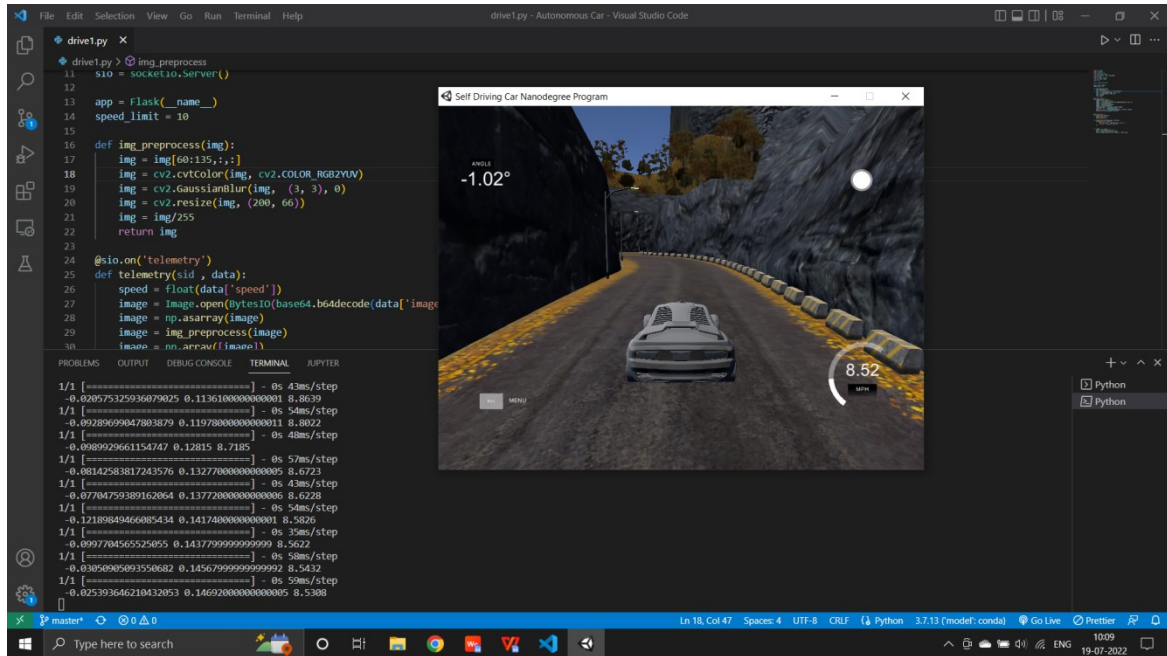


FIG-6.1 JUNGLE TRACK

CHAPTER 7

TESTS

7.1 TESTING

7.1.1 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- All links should take the user to the correct page.

7.1.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects

encountered.

7.1.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.2TEST CASES

- KNOWN USER

FIG-7.2.1 KNOWN USER

- UNKNOWN USER

FIG-7.2.2 UNKNOWN USER

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

This work presented a lightweight pipeline for training and deploying robust driving behavior models on autonomous vehicles using end-to-end imitation learning. The model, which includes 5 layers of convolution and 3 more fully connected layers, was able to learn by cloning human behaviour, and was able to generalize response to a new test track. Though the model only controls the steering angle, it can also be extended to control throttle and brake. The proposed work can be extended to real scenarios as well. This can be done by collecting good driving behavior on real roads and then using that data and feeding it in the model for the predictions. The dataset used for training can be Waymo Open Dataset or Udacity's Open Source dataset. The model can further be made more complex or tweaked to increase accuracy on real road scenarios as well. Also, this model can be trained for real road images from several cameras on the hood of the car i.e. center, left and right camera images can be used as input. Hence, the model once trained can be deployed in a real self-driving car as well.

REFERENCES

- End to End Learning for Self-Driving Cars
<https://arxiv.org/abs/1604.07316>
- Self-Driving Car Steering Angle Prediction Based On Deep Neural Network An Example Of CarND Udacity Simulator
<https://ieeexplore.ieee.org/document/8747006>
- Comparison Review on LiDAR vs Camera in Autonomous Vehicle
<https://www.irjet.net/archives/V7/i8/IRJET-V7I8731.pdf>
- Learning a Driving Simulator <https://arxiv.org/pdf/1608.01230v1.pdf>

APPENDIX

Source code:

```

!git clone https://github.com/rsllim087a/track
!ls track
!pip3 install imgaug
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import keras
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Convolution2D, MaxPooling2D, Dropout, Flatten,
Dense
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from imgaug import augmenters as iaa
import cv2
import pandas as pd
import ntpath
import random
datadir = 'track'
columns = ['center', 'left', 'right', 'steering', 'throttle', 'reverse',
'speed']
data = pd.read_csv(os.path.join(datadir, 'driving_log.csv'), names =
columns)
pd.set_option('display.max_colwidth', -1)
data.head()
def path_leaf(path):
    head, tail = ntpath.split(path)
    return tail
data['center'] = data['center'].apply(path_leaf)
data['left'] = data['left'].apply(path_leaf)
data['right'] = data['right'].apply(path_leaf)
data.head()
num_bins = 25
samples_per_bin = 400
hist, bins = np.histogram(data['steering'], num_bins)
center = (bins[:-1] + bins[1:]) * 0.5
plt.bar(center, hist, width=0.05)
plt.plot((np.min(data['steering']), np.max(data['steering'])),
(samples_per_bin, samples_per_bin))
print('total data:', len(data))
remove_list = []
for j in range(num_bins):
    list_ = []
    for i in range(len(data['steering'])):
        if data['steering'][i] >= bins[j] and data['steering'][i] <=
bins[j+1]:
            list_.append(i)
    list_ = shuffle(list_)
    list_ = list_[samples_per_bin:]
    remove_list.extend(list_)

print('removed:', len(remove_list))
data.drop(data.index[remove_list], inplace=True)

```

```

print('remaining:', len(data))

hist, _ = np.histogram(data['steering'], (num_bins))
plt.bar(center, hist, width=0.05)
plt.plot((np.min(data['steering']), np.max(data['steering'])),
(samples_per_bin, samples_per_bin))

print(data.iloc[1])
def load_img_steering(datadir, df):
    image_path = []
    steering = []
    for i in range(len(data)):
        indexed_data = data.iloc[i]
        center, left, right = indexed_data[0], indexed_data[1],
indexed_data[2]
        image_path.append(os.path.join(datadir, center.strip()))
        steering.append(float(indexed_data[3]))
        # left image append
        image_path.append(os.path.join(datadir, left.strip()))
        steering.append(float(indexed_data[3])+0.15)
        # right image append
        image_path.append(os.path.join(datadir, right.strip()))
        steering.append(float(indexed_data[3])-0.15)
    image_paths = np.asarray(image_path)
    steerings = np.asarray(steering)
    return image_paths, steerings

image_paths, steerings = load_img_steering(datadir + '/IMG', data)
X_train, X_valid, y_train, y_valid = train_test_split(image_paths,
steerings, test_size=0.2, random_state=6)
print('Training Samples: {}\nValid Samples: {}'.format(len(X_train),
len(X_valid)))
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
axes[0].hist(y_train, bins=num_bins, width=0.05, color='blue')
axes[0].set_title('Training set')
axes[1].hist(y_valid, bins=num_bins, width=0.05, color='red')
axes[1].set_title('Validation set')

def zoom(image):
    zoom = iaa.Affine(scale=(1, 1.3))
    image = zoom.augment_image(image)
    return image
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
zoomed_image = zoom(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title('Original Image')

axs[1].imshow(zoomed_image)
axs[1].set_title('Zoomed Image')

```

```

def pan(image):
    pan = iaa.Affine(translate_percent= {"x" : (-0.1, 0.1), "y": (-0.1,
0.1)})
    image = pan.augment_image(image)
    return image
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
panned_image = pan(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title('Original Image')

axs[1].imshow(panned_image)
axs[1].set_title('Panned Image')
def img_random_brightness(image):
    brightness = iaa.Multiply((0.2, 1.2))
    image = brightness.augment_image(image)
    return image
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
brightness_altered_image = img_random_brightness(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title('Original Image')

axs[1].imshow(brightness_altered_image)
axs[1].set_title('Brightness altered image ')

def img_random_flip(image, steering_angle):
    image = cv2.flip(image,1)
    steering_angle = -steering_angle
    return image, steering_angle
random_index = random.randint(0, 1000)
image = image_paths[random_index]
steering_angle = steerings[random_index]

original_image = mpimg.imread(image)
flipped_image, flipped_steering_angle = img_random_flip(original_image,
steering_angle)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title('Original Image - ' + 'Steering Angle:' +
str(steering_angle))

axs[1].imshow(flipped_image)

```



```

    axs[1].set_title('Flipped Image - ' + 'Steering Angle:' +
str(flipped_steering_angle))
def random_augment(image, steering_angle):
    image = mpimg.imread(image)
    if np.random.rand() < 0.5:
        image = pan(image)
    if np.random.rand() < 0.5:
        image = zoom(image)
    if np.random.rand() < 0.5:
        image = img_random_brightness(image)
    if np.random.rand() < 0.5:
        image, steering_angle = img_random_flip(image, steering_angle)

    return image, steering_angle
ncol = 2
nrow = 10

fig, axs = plt.subplots(nrow, ncol, figsize=(15, 50))
fig.tight_layout()

for i in range(10):
    randnum = random.randint(0, len(image_paths) - 1)
    random_image = image_paths[randnum]
    random_steering = steerings[randnum]

    original_image = mpimg.imread(random_image)
    augmented_image, steering = random_augment(random_image,
random_steering)

    axs[i][0].imshow(original_image)
    axs[i][0].set_title("Original Image")

    axs[i][1].imshow(augmented_image)
    axs[i][1].set_title("Augmented Image")

def img_preprocess(img):
    img = img[60:135,:,:]
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
    img = cv2.GaussianBlur(img, (3, 3), 0)
    img = cv2.resize(img, (200, 66))
    img = img/255
    return img
image = image_paths[100]
original_image = mpimg.imread(image)
preprocessed_image = img_preprocess(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()
axs[0].imshow(original_image)
axs[0].set_title('Original Image')
axs[1].imshow(preprocessed_image)
axs[1].set_title('Preprocessed Image')
def batch_generator(image_paths, steering_ang, batch_size, istraining):

    while True:
        batch_img = []
        batch_steering = []

        for i in range(batch_size):
            random_index = random.randint(0, len(image_paths) - 1)

```

```

    if istraining:
        im, steering = random_augment(image_paths[random_index],
steering_ang[random_index])

    else:
        im = mpimg.imread(image_paths[random_index])
        steering = steering_ang[random_index]

    im = img_preprocess(im)
    batch_img.append(im)
    batch_steering.append(steering)
    yield (np.asarray(batch_img), np.asarray(batch_steering))
x_train_gen, y_train_gen = next(batch_generator(X_train, y_train, 1, 1))
x_valid_gen, y_valid_gen = next(batch_generator(X_valid, y_valid, 1, 0))

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(x_train_gen[0])
axs[0].set_title('Training Image')

axs[1].imshow(x_valid_gen[0])
axs[1].set_title('Validation Image')
def nvidia_model():
    model = Sequential()
    model.add(Convolution2D(24, 5, 5, subsample=(2, 2), input_shape=(66,
200, 3), activation='elu'))
    model.add(Convolution2D(36, 5, 5, subsample=(2, 2), activation='elu'))
    model.add(Convolution2D(48, 5, 5, subsample=(2, 2), activation='elu'))
    model.add(Convolution2D(64, 3, 3, activation='elu'))

    model.add(Convolution2D(64, 3, 3, activation='elu'))
#    model.add(Dropout(0.5))

    model.add(Flatten())

    model.add(Dense(100, activation = 'elu'))
#    model.add(Dropout(0.5))

    model.add(Dense(50, activation = 'elu'))
#    model.add(Dropout(0.5))

    model.add(Dense(10, activation = 'elu'))
#    model.add(Dropout(0.5))

    model.add(Dense(1))

    optimizer = Adam(lr=1e-3)
    model.compile(loss='mse', optimizer=optimizer)
    return model
model = nvidia_model()
print(model.summary())
history = model.fit_generator(batch_generator(X_train, y_train, 100, 1),
                             steps_per_epoch=300,
                             epochs=10,

validation_data=batch_generator(X_valid, y_valid, 100, 0),
                             validation_steps=200,
                             verbose=1,
                             shuffle = 1)

```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training', 'validation'])
plt.title('Loss')
plt.xlabel('Epoch')
model.save('model.h5')
from google.colab import files
files.download('model.h5')
```

Course code	Statement Student will be able to	Cognitive Level	PO / PSO addressed
CO1	Define a problem of the recent advancements with applications towards society.	R	po2,po3,po8,po9,po10,po11,psol,psol2
CO2	Outline requirements and perform requirement analysis for solving the problem.	U,An	Po2,po3,po4,po9,po10,po11,psol2
CO3	Design and develop a software and/or hardware based solution within the scope of project using contemporary technologies and tools.	Ap,C	Po1,Po2,po3,po4,po5,po8,po9,po10,po11,psol,psol2
CO4	Test and deploy the applications for use.	E	po3,po4,po5,po8,po9,po10,po11,psol2
CO5	Develop the Project as a team and Demonstrate the application, with effective written and oral communications.	U,C	Po9,p10

Table 1: Course Outcomes - Cognitive levels
Cognitive Levels: R-Remember; U-Understand; Ap-Apply; An=Analyze; E-Evaluate; C-Create

Course Code	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
No. of PIs addressed by course for a given PO	4	4	4	4	4	4	4	5	5	7	4	6	5	6
CO1		1	2			2		1	2	2	1		1	2
CO2		2	3	1					1	2	1			2
CO3	1	3	3	2	1	2	2	3	2	2	2	2	3	4
CO4			3	2	1	1	2	2	2	2	1			2
CO5									3	4				

Table 2: Relevance of CO - PO/PSO wrt to performance indicators

Note: Entry in each row of CO in table 2 represents number of PIs addressed by CO for a given PO

- Calculate each CO contribution to PO as follows:

Levels	>= 60% - level 3
	>=40 % to <60% - level 2
	>=20 % to <40% - level 1
	<20% – no correlation

$$\text{Contribution (CO)} = \frac{\text{No of key elements of a particular PO satisfied by the CO}}{\text{* Total number of key elements of the same PO}} \times 100$$

- Calculate overall course contribution to PO as follows:

Contribution (course) = Rounded average of levels where it is addressed (in column) of a particular PO

	PO1		PO2		PO3		PO4		PO5		PO6		PO7		PO8		PO9		PO10		PO11		PO12		PSO1		PSO2	
PW861IT	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level	%	Level
CO1			25	1	50	2					50	2			20	1	40	2	50	2	25	1			25	1	34	1
CO2			50	2	75	3	25	1									20	1	50	2	25	1					34	1
CO3	25	1	75	3	75	3	50	2			50	2	50	2	60	3	40	2	50	2	50	2	34	1	75	3	67	3
CO4					75	3	50	2	25	1	50	2	50	2	40	2	40	2	50	2	25	1					34	1
CO5																	60	3	100	3								
No of Cos mapped & total	1	1	3	6	4	11	3	5	1	1	3	6	2	4	3	6	5	10	5	11	4	5	1	1	2	4	4	6
Average of Level	1/1=1		6/3=2		11/4=2.75		5/3=1.6		1/1=1		6/3=2		4/2=2		6/3=2		10/5=2		11/5=2.2		5/4=1.25		1/1=1		4/2=2		6/4=1.5	
Rounded average level	1		2		3		2		1		2		2		2		2		2		1		1		2		2	

Table 3: Calculation of CO-PO/PSO correlation levels

PW861IT	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2
CO1		1	2			2		1	2	2	1		1	2
CO2		2	3	1					1	1	1			2
CO3	1	3	3	2	1	2	2	3	2	2	2	2	3	4
CO4			3	2	1	2	2	2	2	1	1			2
CO5									3	4				
PC 433 IT	1	2	3	2	1	2	2	2	2	2	1	1	2	2

Table 4: Course Articulation Matrix

Project Name	Domain	In-house/ Industry	PO/PSO addressed	Internal Guide

Table 5: PO/PSO addressed by the Project

PO/PSO	PO1,PO2,PO6,PO7				PO3	PO4,PO5, PSO1	PO4,PO5, PSO2	PO8	PO9				PO10			PO11	PO12
Rubrics	R1				R2	R3	R4	R5	R6				R7			R8	R9
Roll. No.	CI	CII	CIII	Total	CIV	CV	CVI	CVII	CVIII	CIX	CX	Total	CXI	CXII	Total	CXIII	CIV
	4	4	4	12	4	4	4	4	4	4	4	12	4	4	8	4	4

Table 6: Rubrics Evaluation

Rubrics for project

Focus Areas:

1. **Problem Formulation (PO1,PO2, PO6,PO7)**
2. **Project Design (PO3)**
3. **Build (PO4,PO5,PSO1)**
4. **Test& Deploy (PO4, PO5,PSO2)**
5. **Ethicalresponsibility (PO8)**
6. **Team Skills (PO9)**
7. **ProjectPresentation (P10)**
8. **Projectmanagement (PO11)**
9. **LifelongLearning (PO12)**

Focus Areas	Criterion [c]	Exemplary 4	Satisfactory 3	Developing 2	Unsatisfactory 1
Problem Formulation (PO1,PO2, PO6, PO7)	I - Identify/Define Problem Ability to identify a suitable problem and define the project objectives.	Demonstrates a skillful ability to identify / articulate a problem and the objectives are well defined and prioritized.	Demonstrates ability to Identify / articulate a problem and All major objectives are identified.	Demonstrates some ability to identify / articulate a problem that is partially connected to the issues and most major objectives are identified but one or two minor ones are missing or priorities are not established.	Demonstrates minimal or no ability to identify / articulate a problem and many major objectives are not identified.
	II - Collection of Background Information: Ability to gather background Information (existing knowledge, research, and/or indications of the problem)	Collects sufficient relevant background information from appropriate sources, and is able to identify pertinent/critical information;	Collects sufficient relevant background information from appropriate sources;	Collects some relevant background information from appropriate Sources.	Minimal or no ability to collect relevant background information
	III- Define scope of the problem Ability to identify problem scope suitable to the degree considering the impact on society and environment	Demonstrates a skillful ability to define the scope of problem accurately mentioning the relevant fields of engineering precisely. Considers, explains and evaluates the impact of engineering interventions on society and environment.	Demonstrates ability to define problem scope mentioning the relevant fields of engineering broadly. Considers and explains the impact of engineering interventions on society and environment	Demonstrates some ability to define problem scope mentioning some of the relevant fields. Some consideration of the impact of engineering interventions on society and environment.	Demonstrates minimal or no ability to define problem scope and fails to mention relevant fields of engineering. Minimal or no consideration of the impact of engineering interventions on society and environment
Project Design (PO3)	IV- Understanding the Design Process and Problem Solving: Ability to explain the design process including the importance of needs, specifications, concept generation and to develop an approach to solve a problem.	Demonstrates a comprehensive ability to understand and explain a design process. Considers multiple approaches to solving a problem, and can articulate reason for choosing solution	Demonstrates an ability to understand and explain a design process. Considers multiple approaches to solving a problem, which is justified and considers consequences.	Demonstrates some ability to understand and explain a design process. Considers a few approaches to solving a problem; doesn't always consider consequences.	Demonstrates minimal or no ability to understand and explain a design process. Considers a single approach to solving a problem. Does not consider consequences.

Build (PO4,PO5, PSO1)	V- Implementing Design Strategy: Ability to execute a solution taking into consideration design requirements using appropriate tool (software/hardware);	Demonstrates a skillful ability to execute a solution taking into consideration all design requirements using the most relevant tool.	Demonstrates an ability to execute a solution taking into consideration design requirements using relevant tool.	Demonstrates some ability to execute a solution but not using most relevant tool.	Demonstrates minimal or no ability to execute a solution. Solution does not directly attend to the problem.
Test & Deploy (PO4,PO5, PSO2)	VI- Evaluating Final Design: To evaluate/confirm the functioning of the final design. To deploy the project on the target environment	Demonstrates a skillful ability to evaluate/confirm the functioning of the final design skillfully, with deliberation for further improvement after deployment.	Demonstrates an ability to evaluate/confirm the functioning of the final design. The evaluation is complete and has sufficient depth.	Ability to evaluate/confirm the functioning of the final design, but the evaluation lacks depth and/or is incomplete.	Demonstrates minimal or no ability to evaluate/confirm the functioning of the final design.
Ethical responsibility (PO8)	VII - Proper Use of Others' Work: Ability to recognize, understand and apply proper ethical use of intellectual property, copyrighted materials, and research.	Always recognizes and applies proper ethical use of intellectual property, copyrighted materials, and others' research.	Recognizes and applies proper ethical use of intellectual property, copyrighted materials, and others' research.	Some recognition and application of proper ethical use of intellectual property, copyrighted materials, and others' research.	Minimal or no recognition and/or application of proper ethical use of intellectual property, Copyrighted materials, or others' research.
Team Skills (PO9)	VIII - Individual Work Contributions and Time Management: Ability to carry out individual Responsibilities and manage time (estimate, prioritize, establish deadlines/ milestones, follow timeline, plan for contingencies, adapt to change).	Designated jobs are accomplished by deadline; completed work is carefully and meticulously prepared and meets all requirements.	Designated jobs are accomplished by deadline; completed work meets requirements.	Designated jobs are accomplished by deadline; completed work meets most requirements.	Some Designated jobs are accomplished by deadline; completed work meets some requirements.
	IX - Leadership Skills: Ability to lead a team. (i) Mentors and accepts mentoring from others. (ii) Demonstrates capacity for initiative while respecting others' roles. (iii) Facilitates others' involvement. (iv) Evaluates team Effectiveness and plans for improvements	Exemplifies leadership skills.	Demonstrates leadership skills.	Demonstrates some leadership skills at times.	Demonstrates minimal or no Leadership skills.
	X - Working with Others: Ability to listen to, collaborate with, and champion the efforts of others.	Skillfully listens to, collaborates with, and champions the efforts of others.	Listens to, collaborates with, and champions the efforts of others.	Sometimes listens to, collaborates with, and champions others' efforts.	Rarely listens to, collaborates with, or champions others' efforts.

Project Presentation (P10)	XI - Technical Writing Skills Ability to communicate the main idea with clarity. Ability to use illustrations properly to support ideas (citations, position on page etc)	Main idea is clearly and precisely stated. Materials are seamlessly arranged in a logical sequence. Illustrations are skillfully used to support ideas	Main idea is understandable. Material moves logically forward, Illustrations are properly used to support ideas	Main idea is somewhat Understandable. Material has some logical order and is somewhat coherent or easy to follow. Illustrations are for the most part properly used to support ideas	Main idea is difficult to understand. Material has little logical order, and is often unclear, incoherent. Illustrations are used, but minimally support ideas. (not properly cited etc)
	XII - Communication Skills for Oral Reports Ability to present strong key ideas and supporting details with clarity and concision. Maintain contact with audience, and ability to complete in the allotted time	Presentation logically and skillfully structured. Key ideas are compelling, and articulated with exceptional clarity and concision. Introduction, supporting details and summary are clearly evident and memorable, and ascertain the credibility of the speaker. Presentation fits perfectly within time constraint.	Presentation has clear structure and is easy to follow. Key ideas are clearly and concisely articulated, and are interesting. There is sufficient detail to ascertain speaker's authority, and presentation includes an introduction and summary. Presentation fits within time constraint, though presenter might have to subtly rush or slow down.	Presentation has some structure. Key ideas generally identifiable, although not very remarkable. Introduction, supporting details and/or summary may be too broad, too detailed or missing. Credibility of the speaker may be questionable at times. Presentation does not quite fit within time constraint; presenter has to rush or slow down at end	Presentation rambles. Not organized; key ideas are difficult to identify, and are unremarkable. No clear introduction, supporting details and summary. Speaker has no credibility. Presentation is unsuitably short or unreasonably long.
Project management (PO11)	XIII Monitoring and Controlling the Project	Monitors timelines and progress toward project goals on a daily basis. Provides accurate, complete reports of project progress.	Monitors timelines and progress toward project goals most of the time. Provides relatively accurate, complete reports of project progress with only minor errors or omissions	Seldom monitors timelines and progress toward project goals. Provides relatively accurate, yet clearly incomplete, reports of project progress	Does not monitor timelines and progress toward project goals. Provides inaccurate, incomplete reports of project progress
Lifelong Learning (PO12)	XIV - Extend Scope of Work: Ability to extend the project through implementation in other study areas	Demonstrates a skillful ability to explore a subject/topic thoroughly, discusses the road map to extend the project in other areas.	Demonstrates an ability to explore a subject/topic, and shows possible areas in which project can be extended	Demonstrates some ability to explore a subject/topic, providing some knowledge of areas in which project can be extended	Demonstrates minimal or no ability to explore a subject/topic, and does not discuss future work clearly mentioning other areas