|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENT** | **PAGE NO** |
|  |  |  |
| **I** | **ABSTRACT** | **i** |
| **II** | **LIST OF ABBREVIATIONS** | **ii** |
| **1** | **INTRODUCTION** | **1** |
| **2** | **FEASIBILITY STUDY**   * 1. Technical Feasibility   2. Economical Feasibility   3. Social Feasibility | **5** |
| **3** | **LITERATURE REVIEW** | **7** |
| **4**  **5** | **SYSTEM ANALYSIS**   * 1. System Study   2. Auto Copilot DAVE-2 System 5 Methodology   3. System Architecture   4. Data Collection   5. Network Architecture   6. Training Details   7. Agumentation   8. Simulation   9. Evaluation   10. Simulation Test   11. On-Road Test   12. Visualization of CNN   13. Advantages   14. Disadvantages   15. System Requirements | **9** |
| **6** | **SOFTWARE ENVIRONMENT**   * 1. Introduction to Python   2. Installation of Python | **23** |

|  |  |  |
| --- | --- | --- |
| **7** | **SYSTEM DESIGN**   * 1. System Architecture   2. Model Prediction   3. UML Diagrams      1. Use Case Diagram      2. Data flow Diagram      3. Sequence Diagram      4. State diagram | **42** |
| **8** | **IMPLEMENTATION**   * 1. Driving Model   2. Dataset Preparation   3. Train and Test Model   4. Test | **48** |
| **9** | **SYSTEM TESTING**   * 1. Unit Testing   2. Integration Testing   3. Acceptance Testing   4. Risk Analysis   5. Test Cases | **54** |
| **10** | **RESULTS & OUTPUTS** | **58** |
| **11** | **CONCLUSION** | **62** |
| **11** | **BIBILOGRAPHY** | **64** |

# ABSTRACT

We trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved surprisingly powerful. With minimum training data from humans the system learns to drive in traffic on local roads with or without lane markings and on highways. It also operates in areas with unclear visual guidance such as in parking lots and on unpaved roads. The system automatically learns internal representations of the necessary processing steps such as detecting useful road features with only the human steering angle as the training signal. We never explicitly trained it to detect, for example, the outline of roads. Compared to explicit decomposition of the problem, such as lane marking detection, path planning, and control, our end-to-end system optimizes all processing steps simultaneously. We argue that this will eventually lead to better performance and smaller systems. Better performance will result because the internal components self-optimize to maximize overall system performance, instead of optimizing human-selected intermediate criteria, e. g., 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 the minimal number of processing steps. We used an NVIDIA DevBox and Torch 7 for training and an NVIDIA DRIVETM PX self-driving car computer also running Torch 7 for determining where to drive. The system operates at 30 frames per second (FPS).

# LIST OF ABBREVATIONS

1. CNN - Convolution neural network
2. DAVE-2 - Digital application engineering
3. Tensorflow - Famous Library
4. OpenCV - Open Computer Vision

# INTRODUCTION

CNNs [1] have revolutionized pattern recognition [2]. Prior to the widespread adoption of CNNs, most pattern recognition tasks were performed using an initial stage of hand-crafted feature extraction followed by a classifier. The breakthrough of CNNs is that features are learned automatically from training examples. The CNN approach is especially powerful in image recognition tasks because the convolution operation captures the 2D nature of images. Also, by using the convolution kernels to scan an entire image, relatively few parameters need to be learned compared to the total number of operations.

While CNNs with learned features have been in commercial use for over twenty years [3], their adoption has exploded in the last few years because of two recent developments. First, large, labeled data sets such as the Large Scale Visual Recognition Challenge (ILSVRC) [4] have become available for training and validation. Second, CNN learning algorithms have been implemented on the massively parallel graphics processing units (GPUs) which tremendously accelerate learning and inference.

In this paper, we describe a CNN that goes beyond pattern recognition. It learns the entire processing pipeline needed to steer an automobile. The groundwork for this project was done over 10 years ago in a Defense Advanced Research Projects Agency (DARPA) seedling project known as DARPA Autonomous Vehicle (DAVE) [5] in which a sub-scale radio control (RC) car drove through a junk-filled alley way. DAVE was trained on hours of human driving in similar, but not identical environments. The training data included video from two cameras coupled with left and right steering commands from a human operator.

In many ways, DAVE-2 was inspired by the pioneering work of Pomerleau [6] who in 1989 built the Autonomous Land Vehicle in a Neural Network (ALVINN) system. It demonstrated that an end- toend trained neural network can indeed steer a car on public roads. Our work differs in that 25 years of advances let us apply far more data and computational power to the task. In addition, our experience with CNNs lets us make use of this powerful technology. (ALVINN used a fully-connected network which is tiny by today’s standard.)

1

While DAVE demonstrated the potential of end-to-end learning, and indeed was used to justify starting the DARPA Learning Applied to Ground Robots (LAGR) program [7], DAVE’s performance was not sufficiently reliable to provide a full alternative to more modular approaches to off-road driving. DAVE’s mean distance between crashes was about 20 meters in complex environments.

Nine months ago, a new effort was started at NVIDIA that sought to build on DAVE and create a robust system for driving on public roads. The primary motivation for this work is to avoid the need to recognize specific human-designated features, such as lane markings, guard rails, or other cars, and to avoid having to create a collection of “if, then, else” rules, based on observation of these features.This paper describes preliminary results of this new effort.

The current could be manipulated to tell the vehicle to move the steering wheel left or right. Japanese people followed this work but they used a camera system that relayed data to a computer to process images of the road in 1977. However, this vehicle could only travel at speeds below 20 mph. Contribution of the Germans in the automobile industry is noticeable and here again they proved, a decade later in the form of the VaMoRs, a vehicle outfitted with cameras that could drive itself safely at 56 mph. As technology improved, so did self-driving vehicles’ ability to detect and react to their environment [34]. In 2002, DARPA announced the Urban Grand Challenge, a $1 million prize if a vehicle is able to navigate 142 miles through the Mojave Desert.

In the year 2004, competition won by the Stanford team, under Sebastian Thrun observation. He became project leader in Google’s Waymo project in 2009 [35]. For an automobile to be autonomous, it needs to be continuously aware of its surroundings—first, by perceiving (identifying and classifying information) and then acting on the information through the autonomous/computer control of the vehicle. Autonomous vehicles require safe, secure, and highly responsive solutions which need to be able to make split-second decisions based on a detailed understanding of the driving environment. The National Highway Traffic Safety Administration categorized driving in 6 levels based on automation level . By 2013, many big companies including BMW, Audi, Cruse, Ford, and Mercedes Benz were all working on their own self driving vehicle.

There were many autonomous car fatalities in the past, many people lost their lives in it. Tesla first launched autopilot mode in cars. Nowadays, millions of dollars are invested in these projects

1

throughout the world. If our cars took their own wheels, we could text, tweet, and even play driving games as we lived the lifestyle only a few chauffeured executives can afford today.

But beyond that, we would have to change the ways we think about driving and transport. Tesla autopilot-2015 is the most significant example of the semi-autonomous vehicle and it succeeds on the road. Autopilot mode enabled hands-free control for highway and freeway driving.

For the past decade, self-driving algorithms have drawn growing research efforts from both industry and academia using low-cost vehicle-mounted cameras. In a self-driving car, various automation levels have been described. At **level 0**, there is no automation. The car is controlled by a human driver. **Level 1** and **Level 2** are specialized driver assistance systems where the system is still controlled by a human driver, but a few functions are automated, such as brake, stability control, etc. **Level 3** vehicles are autonomous although it requires a human driver to intervene and monitor. **Level 4** vehicles are completely autonomous, but the automation is restricted to the vehicle’s operating architecture environment i.e. not all driving situations are covered, **Level 5** vehicles are assumed to be fully autonomous and their efficiency should be equal to that of a human driver. In the near future, we are still far from reaching **level 5** self-driving vehicles. However, **level 3/4** self-driving vehicles will theoretically become a reality.

Excellent research and technical breakthroughs in the area of machine learning and computer vision, as well as low-cost vehicle-mounted cameras that can either independently deliver actionable information or supplement other sensors, are the key reason for key technical achievements in these fields. In modern cars, many vision-based driver assistance features have been widely supported. Some of these features include the identification of pedestrians/bicycles, crash avoidance by measuring the width of the front driver, lane departure warning, etc. However, in this project, we focused on the [“An Intelligent Autopilot System that Learns Drive”](https://graspcoding.com/an-intelligent-autopilot-system-that-learns-drive-ai-project/) i.e. autonomous steering largely unexplored activity in the area of machine learning and computer vision.

An autonomous car is also known as a robotic-car, a driverless-car, and a self-driving car capable of detecting and controlling its environment without human input. An autonomous car perceives their surrounding by combining a variety of techniques including laser light, radar, odometry, GPS, and Computer Vision. Sensory information is interpreted by advanced control systems to define suitable navigation routes, as well as relevant signage and obstacles.

1

There is a rise in curiosity about self-driving vehicles. This is because of breakthroughs in deep learning, where deep neural networks are learned to perform tasks that usually need human intervention. To recognize patterns and characteristics in images, CNN applies models, making them helpful in the field of computer vision. Some of the examples of these are image classification, object detection, object detection, etc.

In this **autopilot – *AI Project***, we are implementing a Convolution Neural Network (CNN) to map raw pixels for a self-driving vehicle from the collected images to the steering commands. With minimal human training input, with or without the lane markers, the machine learns unique features to steer on the road. The inspiration is taken from **Udacity Self-driving car** and from **End to End Learning for Self-driving Cars from NVIDIA**. The dataset provided by Udacity was used for testing purposes and planning the dataset. The **End to End Learning for Self-Driving Cars** uses [convnets](https://en.wikipedia.org/wiki/Convolutional_neural_network) to predict steering angle according to the road.

An autonomous car is also known as a robotic-car, a driverless-car, and a self-driving car capable of detecting and controlling its environment without human input. An autonomous car perceives their surrounding by combining a variety of techniques including laser light, radar, odometry, GPS, and Computer Vision. Sensory information is interpreted by advanced control systems to define suitable navigation routes, as well as relevant signage and obstacles.

**autopilot – *AI Project***, we are implementing a Convolution Neural Network (CNN) to map raw pixels for a self-driving vehicle from the collected images to the steering commands. With minimal human training input, with or without the lane markers, the machine learns unique features to steer on the road. The inspiration is taken from **Udacity Self-driving car** and from **End to End Learning for Self-driving Cars from NVIDIA**. The dataset provided by Udacity was used for testing purposes and planning the dataset. The **End to End Learning for Self-Driving Cars** uses [convnets](https://en.wikipedia.org/wiki/Convolutional_neural_network) to predict steering angle according to the road.

1

# FEASIBILITY STUDY

With the advancement of internet technology and the change in the mode of communication, it is found that much first-hand news have been discussed in Internet forums well before they are reported in traditional mass media. Also, this communication channel provides an effective channel for illegal activities such as dissemination of copyrighted movies, threatening messages and online gambling etc. Our proposed System will analyze online plain text sources from selected discussion forums and will classify the text into different groups and system will decide which post is legal and illegal.

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* TECHNICAL FEASIBILITY
* ECONOMICAL FEASIBILITY
* SOCIAL FEASIBILITY

#### Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

This system is more reliable, maintainable, affordable and producible. These are the parameters which are considered during design and development of this project. During design and development phase of this project there was appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters.

1

#### Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

This system can be used by many websites such as social networking sites and other websites in which discussion forums are available. This system will reduce illegal activities held on internet. This system will provide economic benefits for many websites. It includes quantification and identification of all the benefits expected.

#### Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

6

# LITERATURE REVIEW

Self-driving cars are software based cars that runs own their own. Self-driving cars are an advance technological development in the field of automotive industry that provides both comfort and safety features for drivers. Basic module of this vehicle are lane detection and object detection. Many popular self-driving vehicles have features like they take control when they observe miss mindedness sleepiness of drivers. According to a research by **ASIRT** (Association for Safe international road travel) on average 3,700 people lose their lives every day on the roads. An additional 20-50 million suffer non-fatal injuries, often resulting in long- term disabilities (WHO, 2021). This mostly happens due to human mistakes. To avoid such mistakes self-driving cars come to the ground.

According to Gringer Bonnie’s debate on “History of the Autonomous Car” In titlemax.com the first self-driving model that was proposed by General motors in 1939 that was guided by radio-controlled electromagnetic fields generated with magnetized metal spikes embedded in the road. In 1958 GM’s design was produced on commercial level and car’s front was embedded by coils that were used as sensors. In 1969 John McCarthy put forward his thoughts on self- driving vehicle where he proposed an Idea of la detection using camera. (Gringer, 2021. )

Prof. B. Wang, V. Frémont and S. A. Rodríguez tells in his paper on 'Color-based road detection and its evaluation' tells that Self-driving cars works on image processing techniques on the basis of feature extraction. Every processing is done on each frame of the video. Machines does not take video as a stream of events it takes it as a stream of frames captured at each minimum instant of time. And all processing is done using those frames. For a camera every image is a 2d mesh of colored pixels. And it’s our sense that how we can interpret those colored pixels and turn the frames according to our need. (Wang, 2014)

There are many approaches to implement lane detection. We will discuss all we have seen or learned and then propose ours. First approach that C. Ma and M. Xie, told in their research paper is detecting the lanes by thresholding the given image and calculating left or right on the basis of white pixels. In this method each frame is warped and it is calculated in what direction is the more white pixels. As value of white in colors is 255 and of black is zero. So values off all pixels is calculated column vise and checked for the highest number of value. The side having the highest value means there are more white pixels and vehicle turns that way. This technique

7

is only implementable in a controlled environment where we have a path created with white papers or stuff like that, it cannot work on roads because we have to adjust values of threshold for every environment and it’s not possible on commercial production level. There can be improvements in this method by implementing various steps after it or by using this method (Ma, 2010)

Before canny edge detection to improve edges, another approach proposed by Ammar N. Abbas, Muhammad Asad Irshad is by applying machine learning algorithms that detects the feature sets of raw frames to create dataset of image and steering angle and train data using that dataset. On the basis of trained model, the values of steering are predicted by comparing each image with the path.

This method can only do prediction for a path that is already followed and trained and cannot run on new areas. We need to put in every single road to train model that we need to use in our daily life because it can only predict the steering angle according to image data. Also it makes it slower as it has to compare matrices of arrays to train a model. And also it takes in a lot of data to train so its loss of storage also. So this method is totally not applicable according to our perspective. (Abbas, 2021). Some developers are using the method of detecting lanes by using Hough transformation where raw image is converted into edges using cv2’s built-in method of canny and lines are created using collinear points collected by the method of HoughlinesP and after that they are drawn over the picture.

If we want to conclude the knowledge gained through this literature study we can make a model plan we will use the first method to convert the image to threshold so we can get a clear edge planning so edge detection can be easier by using canny edge detection. One unique thing we have added is cropping the color mapped parts of the image then thresholding. Detailed method will be discussed below.

After thresholding canny edge detection will be applied and to get rid of non-necessary tiny edges we have applied Gaussian blur. At the end Hough transformation is applied Hough transform will generate lines information of collinear and adjacent pixels that will be used to draw lines. (Deng, 2018)

Now let us discuss how the processing works in a self-driving car. It takes raw frame captured at each instant and converts it into threshold of black and white color then it detects the edges. Edges are observed under Hough-lines method where collinear points are collected in to array of arrays and each array is then used as drawing points to draw straight lines.

8

# SYSTEM ANALYSIS

#### System Study

System Analysis is first stage according to System Development Life Cycle model. This System Analysis is a process that starts with the analyst.

Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system. One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. During analysis, data are collected on the available files, decision points, and transactions handled by the present system.

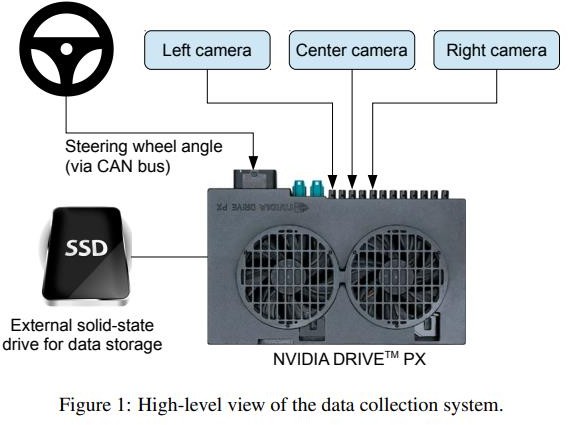
In the near future, we are still far from reaching **level 5** self-driving vehicles. However, **level 3/4** self- driving vehicles will theoretically become a reality. Excellent research and technical breakthroughs in the area of machine learning and computer vision, as well as low-cost vehicle-mounted cameras that can either independently deliver actionable information or supplement other sensors, are the key reason for key technical achievements in these fields. In modern cars, many vision-based driver assistance features have been widely supported. Some of these features include the identification of pedestrians/bicycles, crash avoidance by measuring the width of the front driver, lane departure warning, etc. However, in this project, we focused on the [“An Intelligent Autopilot System that Learns](https://graspcoding.com/an-intelligent-autopilot-system-that-learns-drive-ai-project/) [Drive”](https://graspcoding.com/an-intelligent-autopilot-system-that-learns-drive-ai-project/) i.e. autonomous steering largely unexplored activity in the area of machine learning and computer vision.

#### Auto Copilot DAVE-2 System

Figure 1 shows a simplified block diagram of the collection system for training data for DAVE-2. Three cameras are mounted behind the windshield of the data-acquisition car. Time-stamped video from the cameras is captured simultaneously with the steering angle applied by the human driver. This steering command is obtained by tapping into the vehicle’s Controller Area Network (CAN) bus. In order to make our system independent of the car geometry, we represent the steering command as 1/r where r is the turning radius in meters. We use 1/r instead of r to prevent a singularity when driving straight (the turning radius for driving straight is infinity). 1/r smoothly transitions through zero from left turns (negative values) to right turns (positive values).

Training data contains single images sampled from the video, paired with the corresponding steering 9

command (1/r). Training with data from only the human driver is not sufficient. The network must learn how to recover from mistakes. Otherwise the car will slowly drift off the road. The training data is therefore augmented with additional images that show the car in different shifts from the center of the lane and rotations from the direction of the road.

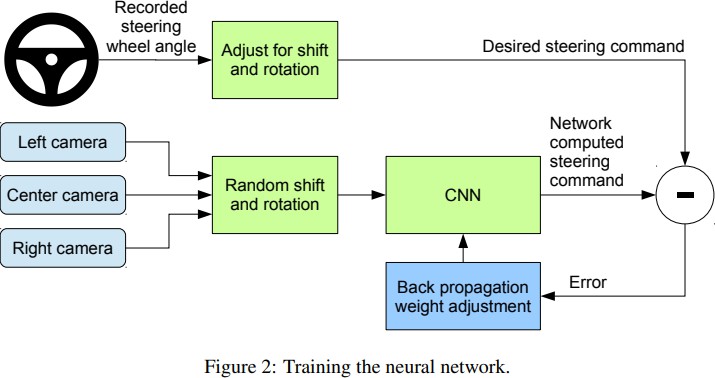


Images for two specific off-center shifts can be obtained from the left and the right camera. Additional shifts between the cameras and all rotations are simulated by viewpoint transformation of the image from the nearest camera. Precise viewpoint transformation requires 3D scene knowledge which we don’t have. We therefore approximate the transformation by assuming all points below the horizon are on flat ground and all points above the horizon are infinitely far away. This works fine for flat terrain but it introduces distortions for objects that stick above the ground, such as cars, poles, trees, and buildings. Fortunately these distortions don’t pose a big problem for network training. The steering label for transformed images is adjusted to one that would steer the vehicle back to the desired location and orientation in two seconds.

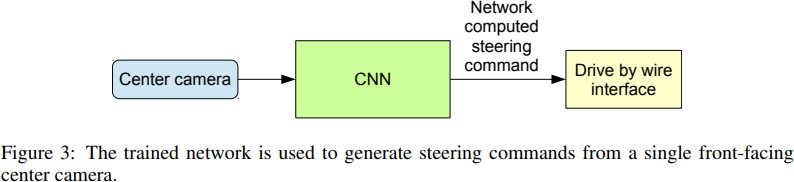
A block diagram of our training system is shown in Figure 2. Images are fed into a CNN which then computes a proposed steering command. The proposed command is compared to the desired command for that image and the weights of the CNN are adjusted to bring the CNN output closer to the desired

10

output. The weight adjustment is accomplished using back propagation as implemented in the Torch.



Once trained, the network can generate steering from the video images of a single center camera. This configuration is shown in Figure 3.



11

# Methodology

#### System Architecture

Self-driving cars works on image processing techniques on the basis of feature extraction. Every processing is done on each frame of the video. Machines does not take video as a stream of events it takes it as a stream of frames captured at each minimum instant of time. And all processing is done using those frames. For a camera every image is a 2d mesh of colored pixels. It takes raw frame captured at each instant and converts it into threshold of black and white color then it detects the edges. Edges are observed under Hough-lines method where collinear points are collected in to array of arrays and each array is then used as drawing points to draw straight lines. Orientations and

length of line to be detected can be set in Hough-Lining. Then the angle of drawn lines is calculated

and left or right decision is taken accordingly

*Figure 1 General Block Diagram for system architecture*

Raw frame



GreyScale

Thresholding



Hough lines



Gausian blur



Canny edge detection



Line creation



Angle

calculation



Movement

methods

Figure 4 General Block Diagram for system architecture

For all these things python has made it possible. We have used the following libraries to do these steps

12

#### Data collection

Training data was collected by driving on a wide variety of roads and in a diverse set of lighting and weather conditions. Most road data was collected in central New Jersey, although highway data was also collected from Illinois, Michigan, Pennsylvania, and New York. Other road types include two- lane roads (with and without lane markings), residential roads with parked cars, tunnels, and unpaved roads. Data was collected in clear, cloudy, foggy, snowy, and rainy weather, both day and night. In some instances, the sun was low in the sky, resulting in glare reflecting from the road surface and scattering from the windshield.

Data was acquired using either our drive-by-wire test vehicle, which is a 2016 Lincoln MKZ, or using a 2013 Ford Focus with cameras placed in similar positions to those in the Lincoln. The system has no dependencies on any particular vehicle make or model. Drivers were encouraged to maintain full attentiveness, but otherwise drive as they usually do. As of March 28, 2016, about 72 hours of driving data was collected.

#### Network Architecture

We train the weights of our network to minimize the mean squared error between the steering command output by the network and the command of either the human driver, or the adjusted steering command for off-center and rotated images (see Section 5.2). Our network architecture is shown in Figure 4. The network consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers. The input image is split into YUV planes and passed to the network.

The first layer of the network performs image normalization. The normalizer is hard-coded and is not adjusted in the learning process. Performing normalization in the network allows the normalization scheme to be altered with the network architecture and to be accelerated via GPU processing.

The convolutional layers were designed to perform feature extraction and were chosen empirically through a series of experiments that varied layer configurations. We use strided convolutions in the first three convolutional layers with a 2×2 stride and a 5×5 kernel and a non-strided convolution with a 3×3 kernel size in the last two convolutional layers.

We follow the five convolutional layers with three fully connected layers leading to an output control value which is the inverse turning radius. The fully connected layers are designed to function as a

controller for steering, but we note that by training the system end-to-end, it is not possible to make a 13

clean break between which parts of the network function primarily as feature extractor and which serve as controller

#### Training Details (Data Collection)

The first step to training a neural network is selecting the frames to use. Our collected data is labeled with road type, weather condition, and the driver’s activity (staying in a lane, switching lanes, turning, and so forth). To train a CNN to do lane following we only select data where the driver was staying in a lane and discard the rest. We then sample that video at 10 FPS. A higher sampling rate would result in including images that are highly similar and thus not provide much useful information.

To remove a bias towards driving straight the training data includes a higher proportion of frames that represent road curves.

The first layer of the network performs image normalization. The normalizer is hard-coded and is not adjusted in the learning process. Performing normalization in the network allows the normalization scheme to be altered with the network architecture and to be accelerated via GPU processing.

The convolutional layers were designed to perform feature extraction and were chosen empirically through a series of experiments that varied layer configurations. We use strided convolutions in the first three convolutional layers with a 2×2 stride and a 5×5 kernel and a non-strided convolution with a 3×3 kernel size in the last two convolutional layers.

Data was acquired using either our drive-by-wire test vehicle, which is a 2016 Lincoln MKZ, or using a 2013 Ford Focus with cameras placed in similar positions to those in the Lincoln. The system has no dependencies on any particular vehicle make or model. Drivers were encouraged to maintain full attentiveness, but otherwise drive as they usually do. As of March 28, 2016, about 72 hours of driving data was collected.

The simulator takes pre-recorded videos from a forward-facing on-board camera on a human-driven data-collection vehicle and generates images that approximate what would appear if the CNN were, instead, steering the vehicle. These test videos are time-synchronized with recorded steering commands generated by the human driver.

14

Since human drivers might not be driving in the center of the lane all the time, we manually calibrate the lane center associated with each frame in the video used by the simulator. We call this position the “ground truth”.

We estimate what percentage of the time the network could drive the car (autonomy). The metric is determined by counting simulated human interventions (see Section 6). These interventions occur when the simulated vehicle departs from the center line by more than one meter. We assume that in real life an actual intervention would require a total of six seconds: this is the time required for a human to retake control of the vehicle, re-center it, and then restart the self-steering mode. We calculate the percentage autonomy by counting the number of interventions, multiplying by 6 seconds, dividing by the elapsed time of the simulated test, and then subtracting the result from 1:

15

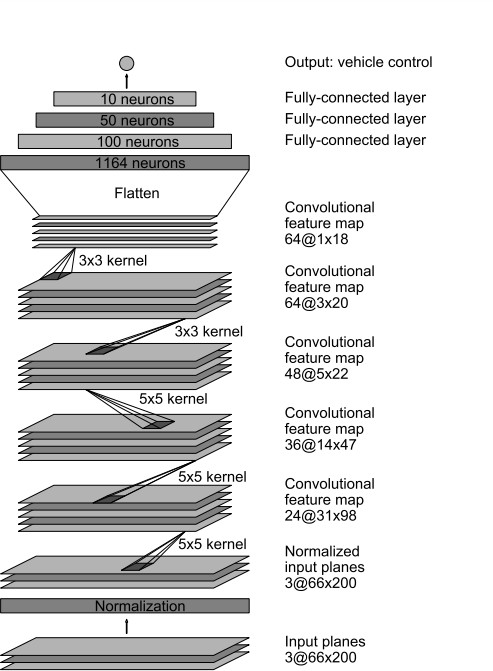


Figure 5 : CNN architecture. The network has about 27 million connections and 250 thousand parameters.

16

#### Agumentation

After selecting the final set of frames we augment the data by adding artificial shifts and rotations to teach the network how to recover from a poor position or orientation. The magnitude of these perturbations is chosen randomly from a normal distribution. The distribution has zero mean, and the standard deviation is twice the standard deviation that we measured with human drivers. Artificially augmenting the data does add undesirable artifacts as the magnitude increases (see Section 2).

#### Simulation

Before road-testing a trained CNN, we first evaluate the networks performance in simulation. A simplified block diagram of the simulation system is shown in Figure 6.

The simulator takes pre-recorded videos from a forward-facing on-board camera on a human-driven data-collection vehicle and generates images that approximate what would appear if the CNN were, instead, steering the vehicle. These test videos are time-synchronized with recorded steering commands generated by the human driver.

Since human drivers might not be driving in the center of the lane all the time, we manually calibrate the lane center associated with each frame in the video used by the simulator. We call this position the “ground truth”.

The simulator transforms the original images to account for departures from the ground truth. Note that this transformation also includes any discrepancy between the human driven path and the ground truth. The transformation is accomplished by the same methods described in Section 2. The simulator accesses the recorded test video along with the synchronized steering commands that occurred when the video was captured. The simulator sends the first frame of the chosen test video, adjusted for any departures from the ground truth, to the input of the trained CNN. The CNN then returns a steering command for that frame. The CNN steering commands as well as the recorded human-driver commands are fed into the dynamic model [8] of the vehicle to update the position and orientation of the simulated vehicle.

The simulator then modifies the next frame in the test video so that the image appears as if the vehicle were at the position that resulted by following steering commands from the CNN. This new image is then fed to the CNN and the process repeats. The simulator records the off-center distance (distance

from the car to the lane center), the yaw, and the distance traveled by the virtual car.

17

When the off-center distance exceeds one meter, a virtual human intervention is triggered, and the virtual vehicle position and orientation is reset to match the ground truth of the corresponding frame of the original test video.

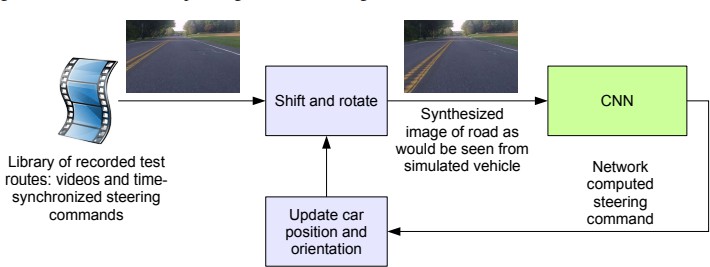


Figure 6 : Block-diagram of the drive simulator

#### Evaluation

Evaluating our networks is done in two steps, first in simulation, and then in on-road tests. In simulation we have the networks provide steering commands in our simulator to an ensemble of prerecorded test routes that correspond to about a total of three hours and 100 miles of driving in Monmouth County, NJ. The test data was taken in diverse lighting and weather conditions and includes highways, local roads, and residential streets.

#### Simulation Tests

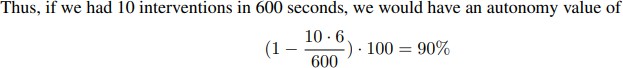
We estimate what percentage of the time the network could drive the car (autonomy). The metric is determined by counting simulated human interventions (see Section 6). These interventions occur when the simulated vehicle departs from the center line by more than one meter. We assume that in real life an actual intervention would require a total of six seconds: this is the time required for a human to retake control of the vehicle, re-center it, and then restart the self-steering mode. We calculate the percentage autonomy by counting the number of interventions, multiplying by 6 seconds, dividing by the elapsed time of the simulated test, and then subtracting the result from 1:

18





Figure 7 : Screen shot of the simulator in interactive mode. See Section 7.1 for explanation of the performance metrics. The green area on the left is unknown because of the viewpoint transformation. The highlighted wide rectangle below the horizon is the area which is sent to the CNN.



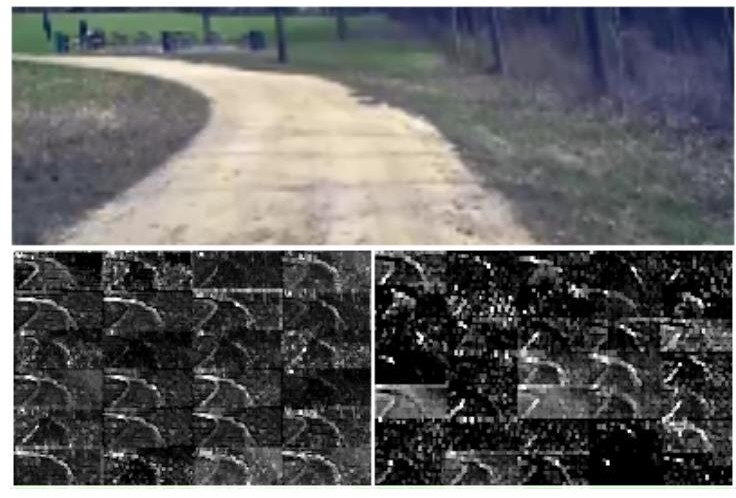
19

#### On-road Tests

After a trained network has demonstrated good performance in the simulator, the network is loaded on the DRIVETM PX in our test car and taken out for a road test. For these tests we measure performance as the fraction of time during which the car performs autonomous steering. This time excludes lane changes and turns from one road to another. For a typical drive in Monmouth County NJ from our office in Holmdel to Atlantic Highlands, we are autonomous approximately 98% of the time. We also drove 10 miles on the Garden State Parkway (a multi-lane divided highway with on and off ramps) with zero intercepts. A video of our test car driving in diverse conditions can be seen in .

#### Visualization of Internal CNN State

Figures 8 and 9 show the activations of the first two feature map layers for two different example inputs, an unpaved road and a forest. In case of the unpaved road, the feature map activations clearly show the outline of the road while in case of the forest the feature maps contain mostly noise, i. e., the CNN finds no useful information in this image. This demonstrates that the CNN learned to detect useful road features on its own, i. e., with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads, for example.



20

Figure 8: How the CNN “sees” an unpaved road. Top: subset of the camera image sent to the CNN. Bottom left: Activation of the first layer feature maps. Bottom right: Activation of the second layer feature maps. This demonstrates that the CNN learned to detect useful road features on its own, i. e., with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads.

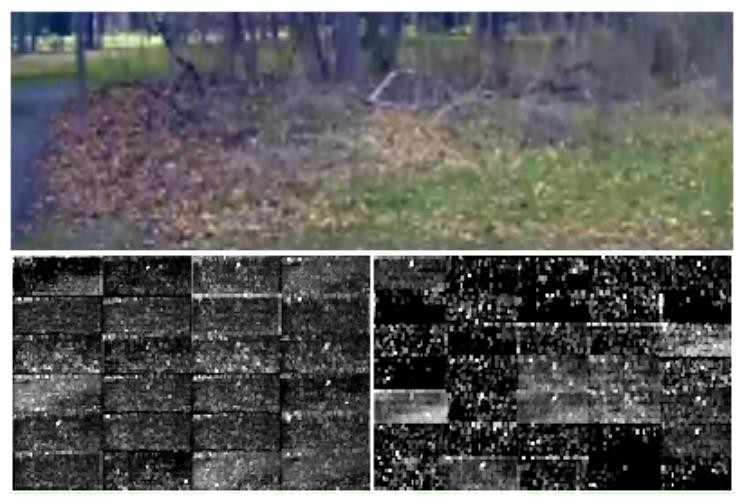


Figure 9: Example image with no road. The activations of the first two feature maps appear to contain mostly noise, i. e., the CNN doesn’t recognize any useful features in this image.

#### Advantages

* Safety; reduce accidents rate.
* Machines stick to the rules
* Speed - Save time
* Frictionless transport, no congestion
* New balances of public spaces; perhaps even road trains and intelligent routing to augment or replace public transport
* Improves health quality

21

* More Luxury

#### Disadvantages

* Reduce job
* Expensive
* Hackers can change route
* Failure of sensor, vehicle can create chances of accident
  + 1. **System Requirements**
       1. **Hardaware Requirements**
          - **CPU : 64-bit operating system, x64-based processor**

### HDD : TB

* + - * + **RAM : GB**
        + **Mobile : Android, Ios**
      1. **Software Requirements**
         * **Operating System : Windows Family**
         * **Coding Language : Python 3.6 & 3.7**
         * **Library : Tensorflow, Keras (Backend)**
         * **OpenCV : Famous Library**

22

* 1. **SOFTWARE ENVIRONMENT**

#### Introduction to Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. The Python 2 language, i.e., Python 2.7.x, was officially discontinued on 1 January 2020 (first planned for 2015) after which security patches and other improvements will not be released for it.[32][33] With Python 2's end-of-life, only Python 3.5.x and later are supported. Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open- source implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

### SYNTAX AND SEMANTICS

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation.

Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

### INDENTION

Main article: Python syntax and semantics § Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end

of the current block. Thus, the program's visual structure accurately represents the program's semantic 23

structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages, indentation doesn't have any semantic meaning.

### STATEMENTS AND CONTROL FLOW

Python's statements include (among others):

The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object.

Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound.

Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However, at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.

* The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
* The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
* The while statement, which executes a block of code as long as its condition is true.
* The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
* The raise statement, used to raise a specified exception or re-raise a caught exception.

24

* The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
* The def statement, which defines a function or method.
* The with statement, from Python 2.5 released in September 2006, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behaviour and replaces a common try/finally idiom.
* The break statement, exits from the loop.
* The continue statement, skips this iteration and continues with the next item.
* The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
* The assert statement, used during debugging to check for conditions that ought to apply.
* The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.

The import statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import <module name> [as <alias>] or from <module name> import \* or from <module name> import <definition 1> [as <alias 1>],

<definition 2> [as <alias 2>],

The print statement was changed to the print () function in Python 3.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

### EXPRESSIONS

Some Python expressions are similar to languages such as C and Java, while some are not: Addition, subtraction, and multiplication are the same, but the behaviour of division differs.

There are two types of divisions in Python. They are floor division (or integer division) // and floating point/division. Python also added the \*\* operator for exponentiation.

From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.

25

From Python 3.8, the syntax: =, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.

In Python, == compares by value, versus Java, which compares numeri’s by value and objects by reference. (Value comparisons in Java on objects can be performed with the equals () method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example a <= b <= c.

Python uses the words and, or, not for its Boolean operators rather than the symbolic &&, ||, ! used in Java and C.

Python has a type of expression termed a list comprehension. Python 2.4 extended list comprehensions into a more general expression termed a generator expression.

Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.

Conditional expressions in Python are written as x if c else y (different in order of operands from the c? x : y operator common to many other languages).

Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.

Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through, and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

Python has a "string format" operator %. These functions analogous to printf format strings in C, e.g. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2".

26

In Python 3 and 2.6+, this was supplemented by the format () method of the str class, e.g. "spam={0} eggs={1}". format("blah", 2). Python 3.6 added "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'.

#### Python has various kinds of string literals

Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl- influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".

Triple-quoted strings, which begin and end with a series of three single or double quote marks.

They may span multiple lines and function like here documents in shells, Perl and Ruby.

Raw string varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.

Python has array index and array slicing expressions on lists, denoted as a[key], a[start: stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

List comprehensions vs. for-loops Conditional expressions vs. if blocks

The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c = 1) { ... } is syntactically valid (but probably unintended) C code but if c = 1: ... causes a syntax error in Python.

27

### METHODS

Methods on objects are functions attached to the object's class; the syntax instance. method(argument) is, for normal methods and functions, syntactic sugar for Class. method(instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby).

### APPLICATIONS OF PYTHON

As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

**Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

**Easy-to-read** − Python code is more clearly defined and visible to the eyes.

**Easy-to-maintain** − Python's source code is fairly easy-to-maintain.

**A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

**Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

**Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

**Databases** − Python provides interfaces to all major commercial databases.

**GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable** − Python provides a better structure and support for large programs than shell scripting.

#### Python OOPs Concepts

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In [Python,](https://www.javatpoint.com/python-tutorial) we can easily create and use classes and objects.

28

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

Major principles of object-oriented programming system are given below.

* Class
* Object
* Method
* Inheritance
* Polymorphism
* Data Abstraction
* Encapsulation

#### Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

**Syntax**

**class** ClassName:

<statement-1>

.

.

<statement-N>

#### Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute doc , which returns the docstring defined in the function source code.

When we define a class, it needs to create an object to allocate the memory. Consider the following example.

29

#### Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

#### Inheritance

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object.

By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.

it provides the re-usability of the code.

#### Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many, and morph means shape. By polymorphism, we understand that one task can be performed in different ways. For example - you have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

#### Encapsulation

Encapsulation is also an essential aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

#### Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonyms because data abstraction is achieved through encapsulation.

Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

#### Python Class and Objects

30

We have already discussed in previous tutorial, a class is a virtual entity and can be seen as a blueprint of an object. The class came into existence when it instantiated. Let's understand it by an example.

Suppose a class is a prototype of a building. A building contains all the details about the floor, rooms, doors, windows, etc. we can make as many buildings as we want, based on these details. Hence, the building can be seen as a class, and we can create as many objects of this class.

On the other hand, the object is the instance of a class. The process of creating an object can be called instantiation.

In this section of the tutorial, we will discuss creating classes and objects in Python. We will also discuss how a class attribute is accessed by using the object.

#### Creating classes in Python

In Python, a class can be created by using the keyword class, followed by the class name. The syntax to create a class is given below.

Syntax

**class** ClassName:

#statement\_suite

In Python, we must notice that each class is associated with a documentation string which can be accessed by using **<class-name>. doc .** A class contains a statement suite including fields, constructor, function, etc. definition.

Consider the following example to create a class **Employee** which contains two fields as Employee id, and name.

The class also contains a function **display(),** which is used to display the information of the **Employee.**

Here, the **self** is used as a reference variable, which refers to the current class object. It is always the first argument in the function definition. However, using **self** is optional in the function call.

#### The self-parameter

The self-parameter refers to the current instance of the class and accesses the class variables. We can use anything instead of self, but it must be the first parameter of any function which belongs to the class.

#### Creating an instance of the class

A class needs to be instantiated if we want to use the class attributes in another class or method.

31

A class can be instantiated by calling the class using the class name.

The syntax to create the instance of the class is given below.

<object-name> = <class-name>(<arguments>)

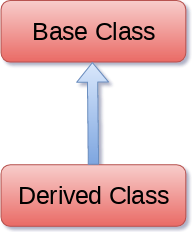
The following example creates the instance of the class Employee defined in the above example.

#### Python Inheritance

Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch.

In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class. In this section of the tutorial, we will discuss inheritance in detail.

In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name. Consider the following syntax to inherit a base class into the derived class.



#### Syntax

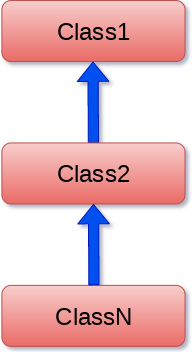
**class** derived-**class**(base **class**):

<**class**-suite>

#### Python Multi-Level inheritance

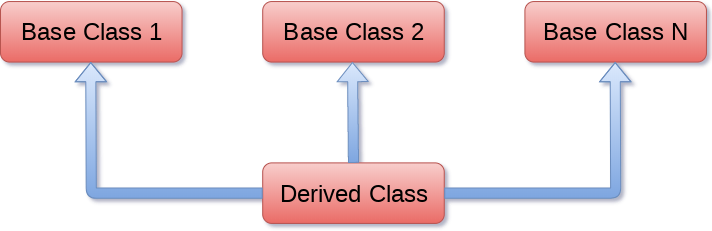
Multi-Level inheritance is possible in python like other object-oriented languages. Multi-level inheritance is archived when a derived class inherits another derived class. There is no limit on the number of levels up to which, the multi-level inheritance is archived in python.

32



#### Python Multiple inheritance

Python provides us the flexibility to inherit multiple base classes in the child class.



#### Method Overriding

We can provide some specific implementation of the parent class method in our child class. When the parent class method is defined in the child class with some specific implementation, then the concept is called method overriding. We may need to perform method overriding in the scenario where the different definition of a parent class method is needed in the child class.

Data abstraction in python

Abstraction is an important aspect of object-oriented programming. In python, we can also perform data hiding by adding the double underscore ( ) as a prefix to the attribute which is to be

33

hidden. After this, the attribute will not be visible outside of the class through the object.

#### Abstraction in Python

Abstraction is used to hide the internal functionality of the function from the users. The users only interact with the basic implementation of the function, but inner working is hidden. User is familiar with that **"what function does"** but they don't know **"how it does."**

In simple words, we all use the smartphone and very much familiar with its functions such as camera, voice-recorder, call-dialing, etc., but we don't know how these operations are happening in the background. Let's take another example - When we use the TV remote to increase the volume. We don't know how pressing a key increases the volume of the TV. We only know to press the "+" button to increase the volume.

That is exactly the abstraction that works in the [object-oriented concept](https://www.javatpoint.com/python-oops-concepts).

#### Why Abstraction is Important?

In Python, an abstraction is used to hide the irrelevant data/class in order to reduce the complexity. It also enhances the application efficiency. Next, we will learn how we can achieve abstraction using the [Python program.](https://www.javatpoint.com/python-programs)

**Syntax**

from abc **import** ABC

**class** ClassName(ABC):

We import the ABC class from the **abc** module.

#### Abstract Base Classes

An abstract base class is the common application program of the interface for a set of subclasses. It can be used by the third-party, which will provide the implementations such as with plugins. It is also beneficial when we work with the large code-base hard to remember all the classes.

#### Working of the Abstract Classes

Unlike the other high-level language, Python doesn't provide the abstract class itself. We need to import the abc module, which provides the base for defining Abstract Base classes (ABC). The

ABC works by decorating methods of the base class as abstract. It registers concrete classes as the 34

implementation of the abstract base. We use the *@abstractmethod* decorator to define an abstract method or if we don't provide the definition to the method, it automatically becomes the abstract method. Let's understand the following example.

### INSTALLATION OF PYTHON

Installing and using Python on Windows 10 is very simple. The installation procedure involves just three steps:

* Download the binaries
* Run the Executable installer
* Add Python to PATH environmental variables

To install Python, you need to download the official Python executable installer. Next, you need to run this installer and complete the installation steps. Finally, you can configure the PATH variable to use python from the command line.

**Step 1**: Download the Python Installer binaries

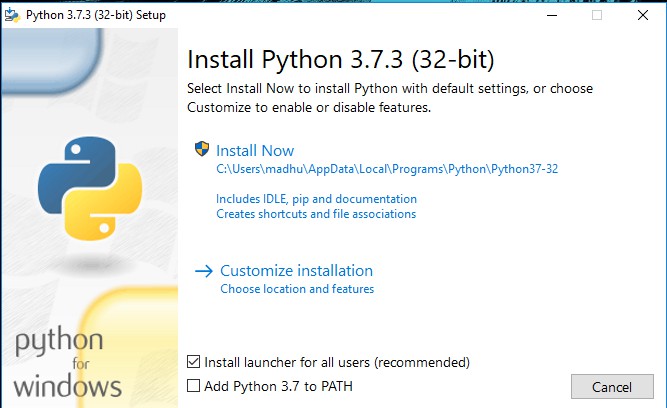
* Open the official Python website in your web browser. Navigate to the Downloads tab for Windows.
* Choose the latest Python 3 release. In our example, we choose the latest Python 3.7.3 version. Click on the link to download Windows x86 executable installer if you are using a 32-bit installer.
* In case your Windows installation is a 64-bit system, then download Windows x86-64 executable installer.

**Step 2:** Run the Executable Installer

1. Once the installer is downloaded, run the Python installer.

35

1. Check the Install launcher for all users check box. Further, you may check the Add Python 3.7 to path check box to include the interpreter in the execution path.



#### Installation Python 3.7.3

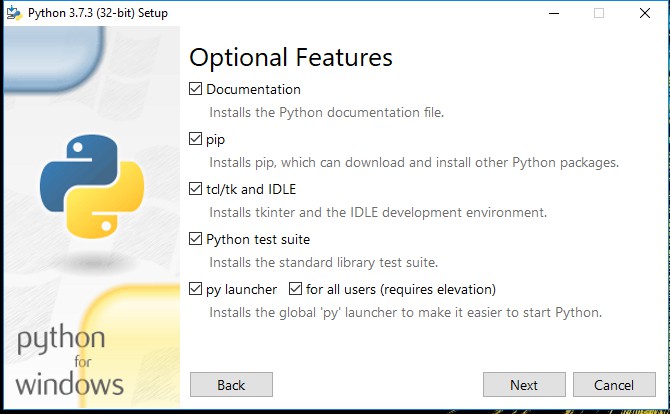
1. **Select Customize installation**.

Choose the optional features by checking the following check boxes:

1. Documentation
2. pip
3. tcl/tk and IDLE (to install tkinter and IDLE)
4. Python test suite (to install the standard library test suite of Python)
5. Install the global launcher for `.py` files. This makes it easier to start Python

36

1. Install for all users.



#### Fig: Optional Features

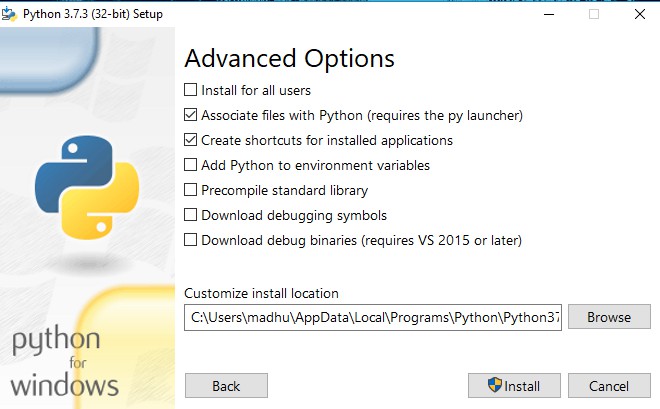
Click Next.

1. This takes you to Advanced Options available while installing Python. Here, select the Install for all users and Add Python to environment variables check boxes.

Optionally, you can select the Associate files with Python, Create shortcuts for installed applications and other advanced options. Make note of the python installation directory displayed in this step. You would need it for the next step.

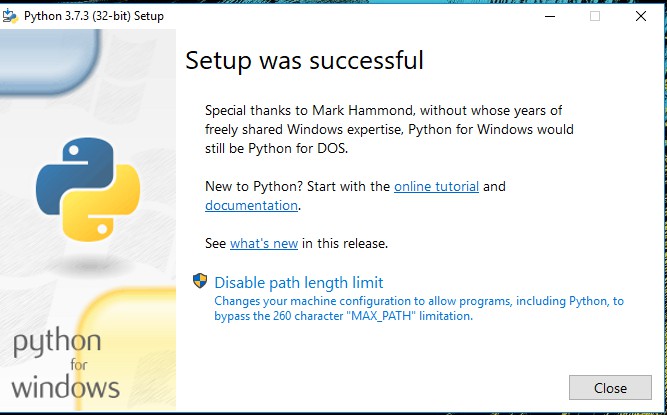
After selecting the Advanced options, click Install to start installation.

37



#### Fig: Advanced Options

1. Once the installation is over, you will see a Python Setup Successful window.



#### Fig : Settings Setup

38

**Step 3:** Add Python to environmental variables

The last (optional) step in the installation process is to add Python Path to the System Environment variables. This step is done to access Python through the command line. In case you have added Python to environment variables while setting the Advanced options during the installation procedure, you can avoid this step. Else, this step is done manually as follows.

In the Start menu, search for “advanced system settings”. Select “View advanced system settings”. In the “System Properties” window, click on the “Advanced” tab and then click on the “Environment Variables” button.

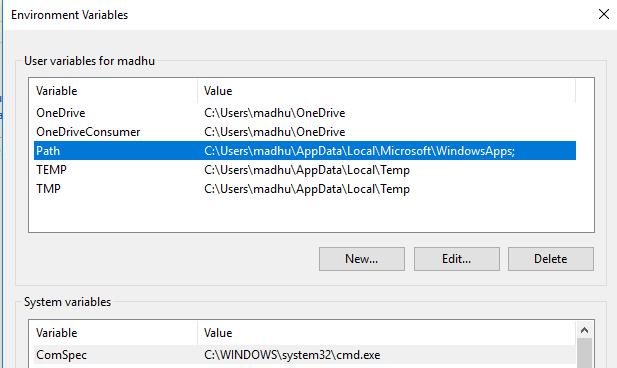
Locate the Python installation directory on your system. If you followed the steps exactly as above, python will be installed in below locations:

* C:\Program Files (x86)\Python37-32: for 32-bit installation
* C:\Program Files\Python37-32: for 64-bit installation

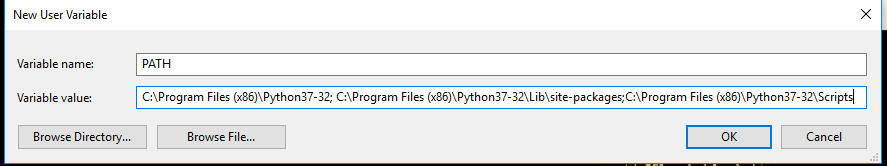
The folder name may be different from “Python37-32” if you installed a different version.

Look for a folder whose name starts with Python.

Append the following entries to PATH variable as shown below:



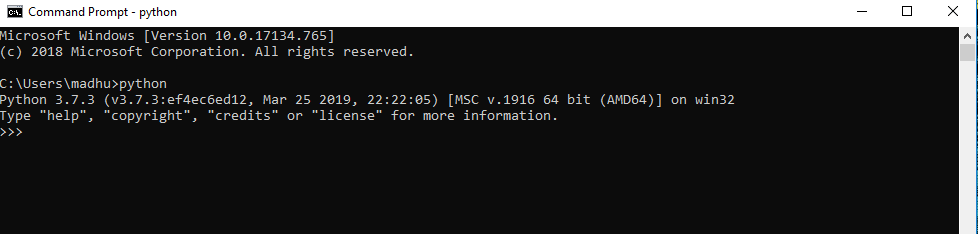
39



#### Environment Settings

**Step 4:** Verify the Python Installation

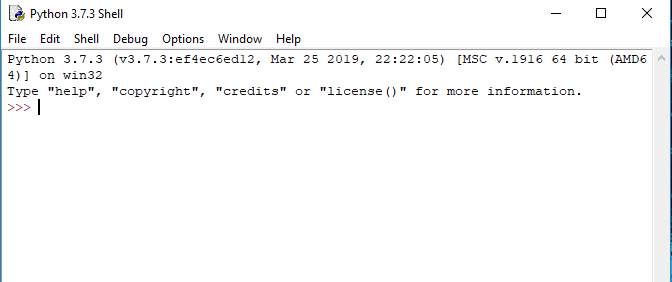
You have now successfully installed Python 3.7.3 on Windows 10. You can verify if the Python installation is successful either through the command line or through the IDLE app that gets installed along with the installation. Search for the command prompt and type “python”. You can see that Python 3.7.3 is successfully installed.



#### Fig: Command Prompt

An alternate way to reach python is to search for “Python” in the start menu and clicking on IDLE (Python 3.7 64-bit). You can start coding in Python using the Integrated Development Environment(IDLE).

40



**Python Shell Prompt**

### USES

Since 2003, Python has consistently ranked in the top ten most popular programming

languages in the TIOBE Programming Community Index where, as of February 2020, it is the third most popular language (behind Java, and C). It was selected Programming Language of the Year in 2007, 2010, and 2018.

* An empirical study found that scripting languages, such as Python, are more productive than conventional languages, such as C and Java, for programming problems involving string manipulation and search in a dictionary, and determined that memory consumption was often "better than Java and not much worse than C or C++".
* Large organizations that use Python include Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify and some smaller entities like ILM and ITA. The social news networking site Reddit is written entirely in Python.
* Python can serve as a scripting language for web applications, e.g., via mod\_wsgi for the Apache web server. With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle and Zope support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications.
* SQLAlchemy can be used as data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.

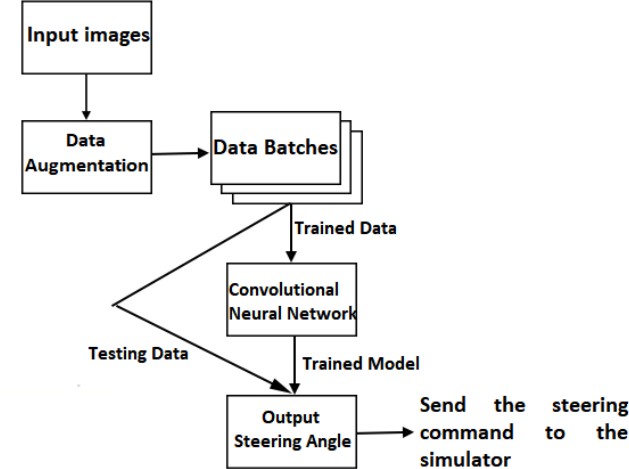
41

* Libraries such as NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing, with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a mathematical software with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus.
* Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like scorewriter and capella.
* GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.
* Python is commonly used in artificial intelligence projects with the help of libraries like TensorFlow, Keras, Pytorch and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.
* Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS 4, FreeBSD (as a package), NetBSD, OpenBSD (as a package) and macOS and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.
* Python is used extensively in the information security industry, including in exploit development.
* Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.
* Due to Python's user-friendly conventions and easy-to-understand language, it is commonly used as an intro language into computing sciences with students. This allows students to easily learn computing theories and concepts and then apply them to other programming languages.
* LibreOffice includes Python, and intends to replace Java with Python. Its Python Scripting Provider is a core feature[169] since Version 4.0 from 7 February 2013.

42

# System Design

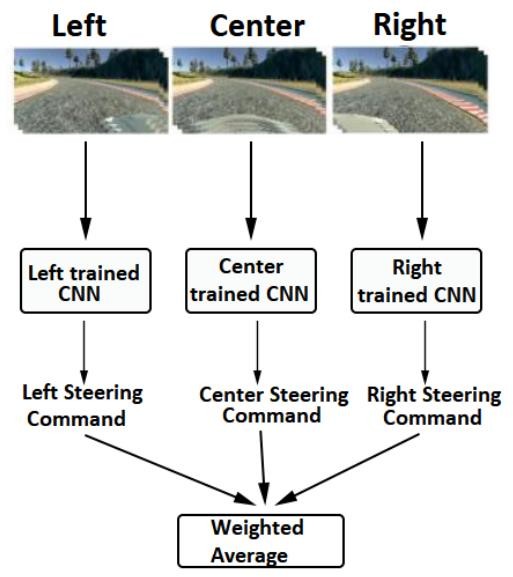
* + 1. **System Architecture**



**Fig.10** System Architecture

43

* + 1. **Model Prediction**



**Fig.11** Model Prediction

#### UML Diagrams

Unified Modeling Language is popular in the market because it is easy to understand. This is part of software engineering. Developer gets better idea about the system.

44

#### Use Case Diagram

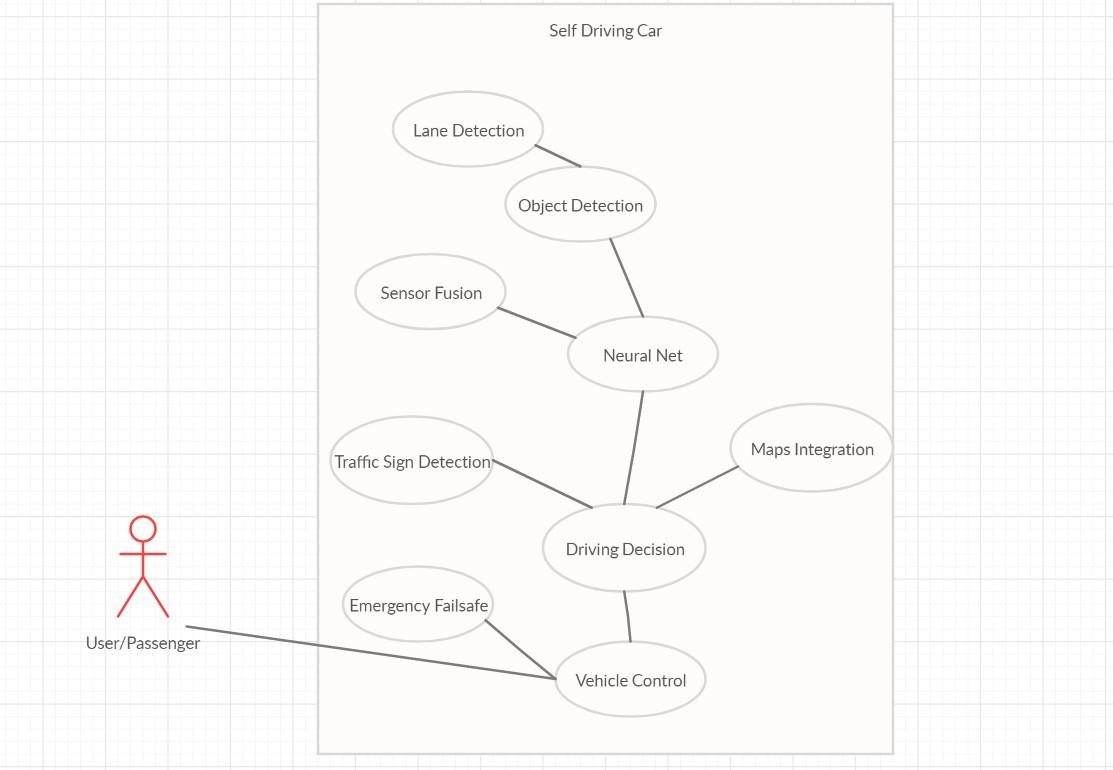


Fig. 12 Activity Diagram

45

# Data Flow Diagram

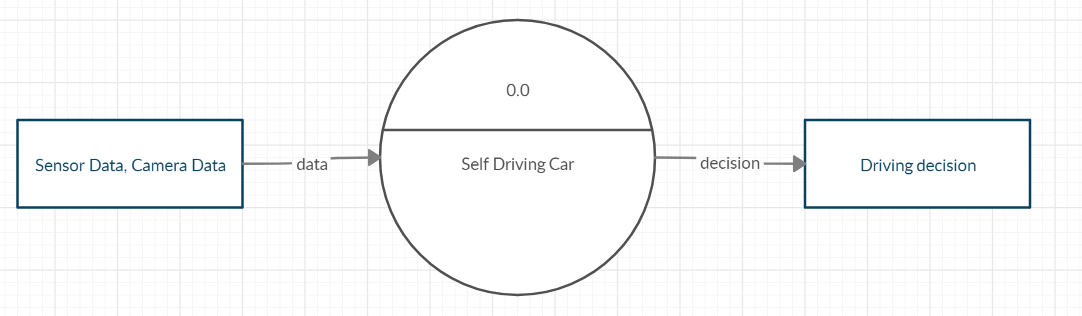


Fig. 13 : Level-0 DFD

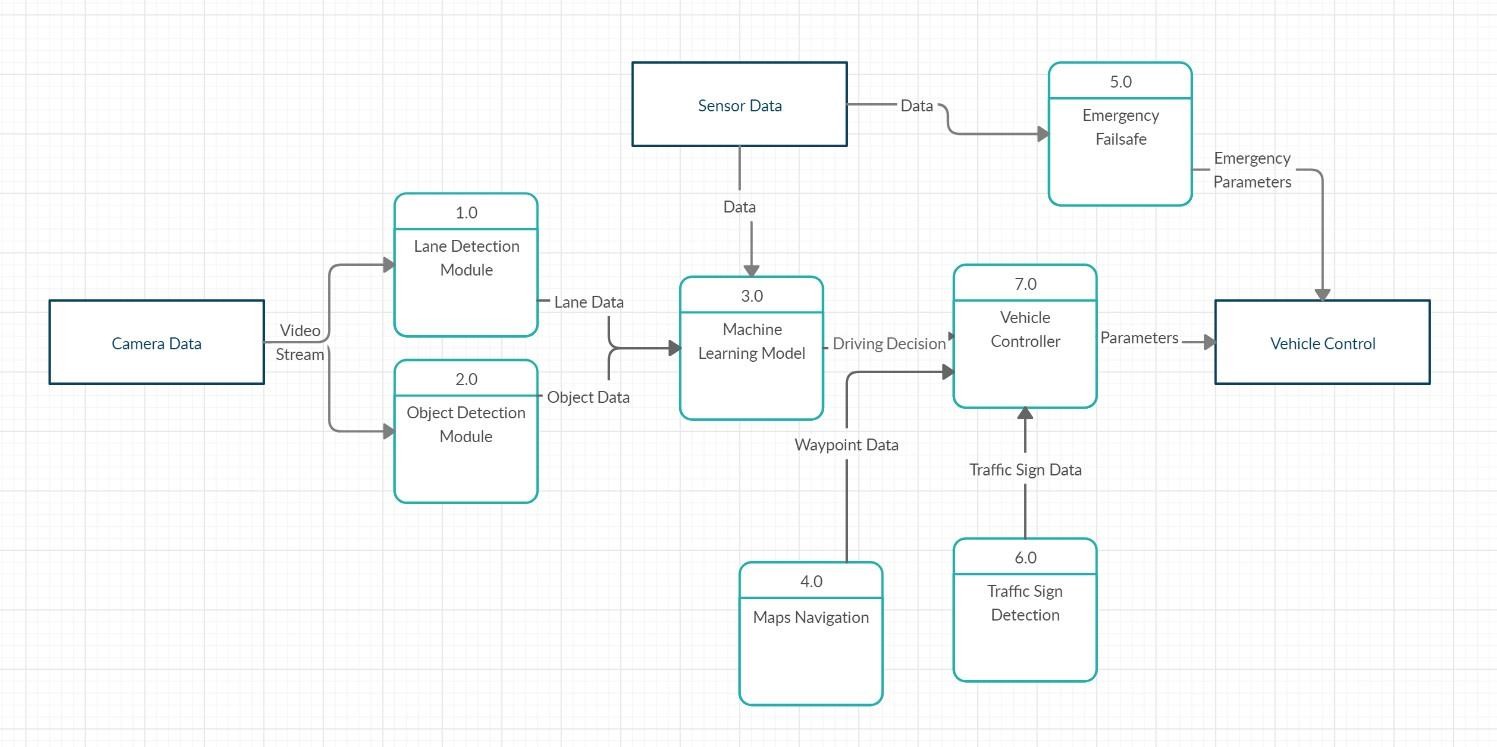


Fig. 14.Level-1 DFD

46

#### Sequence Diagram

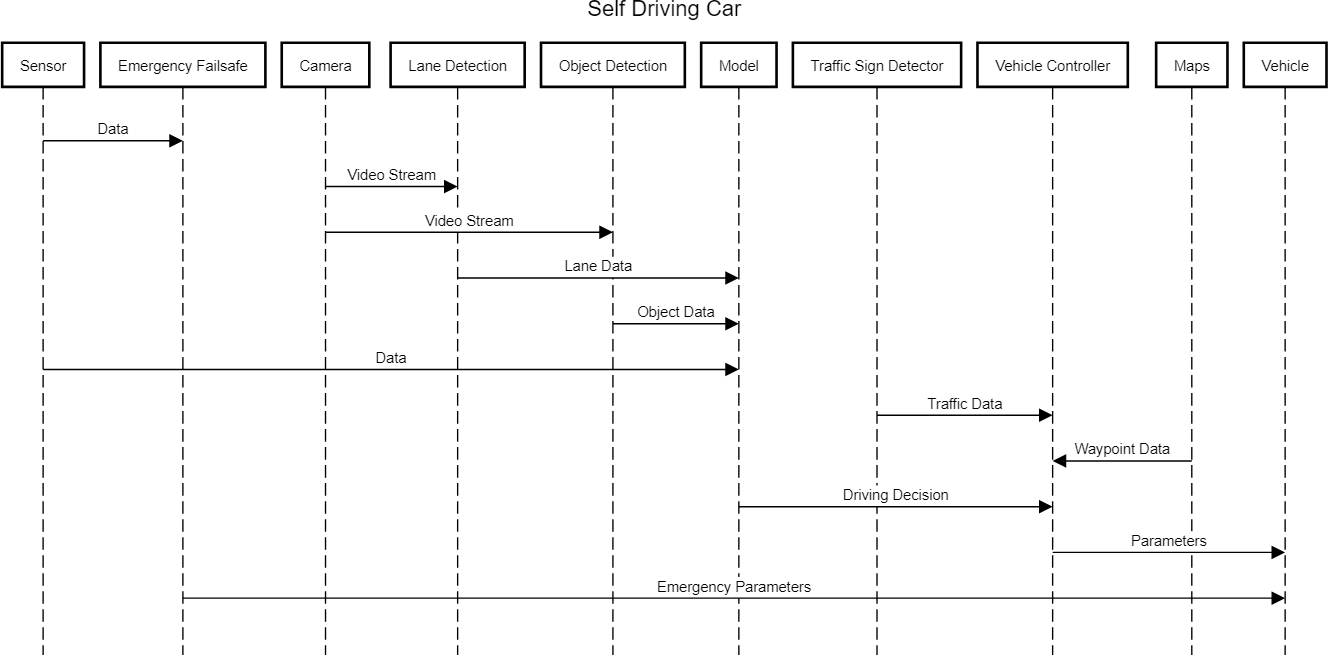


Fig. 15 Sequence Diagram

47

#### State Diagram

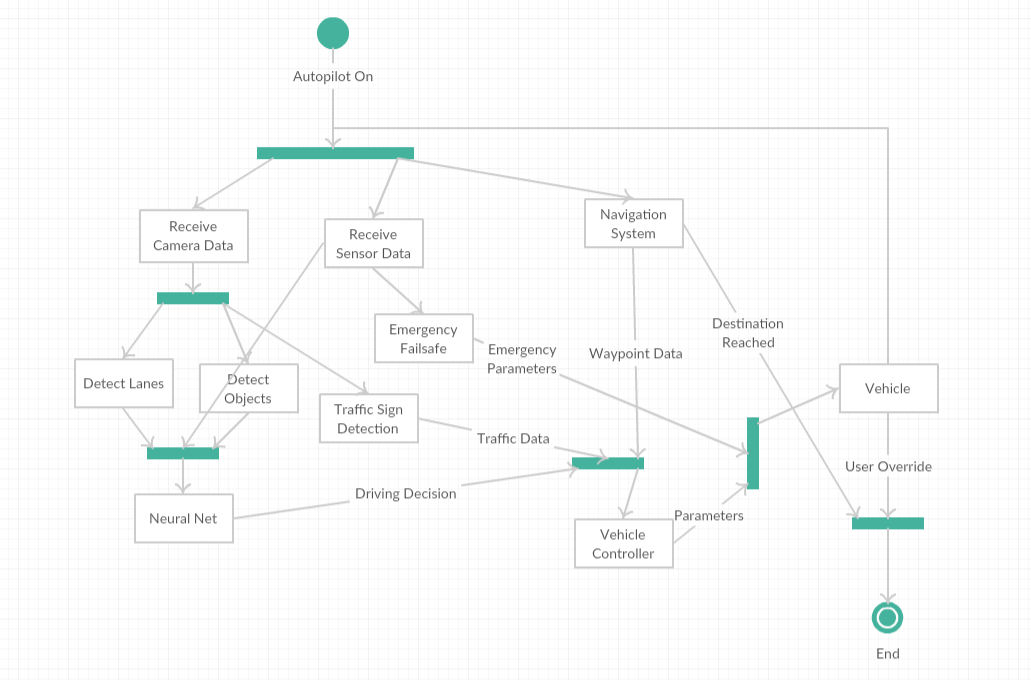


Fig. 16 State Diagram

48

* 1. **IMPLEMENTATION**

First part of the system is the sensors; which are helps to get information about surrounding environment. Perception system give exact location of the objects and path. Planning will be depends on the first two factors. Driving control is based on the various algorithms. I mounted sensors on the real Car which is mentioned in the next chapter. I did supervised learning to check how well neural network predict throttle value and steering angle. After that Object detection has been completed by me using YOLO. Driving rules are the important aspect in driving, so Traffic sign detection has implemented using transfer learning.

Overview of the system is depicted using below block diagram

Sensors

* Camera
* GPS
* Others

Perception

* Object Detection
* Traffic Sign Detection
* Object Tracking
* Localization

Planning

* Route Planning
* Trajectory Prediction
* Behaviour Planning

Control

* PID
* MPC
* Others

Fig. 17 Functional Layers of Self Driving Car

49

#### Driving Models

This section is divided into three parts. First part is about learning types. Second part is the dataset and lastly training. Humans learn in many ways like observing, experience, based on rewards and so on. Like humans, computers also learn accordingly. Supervised learning is student-teacher concepts; Supervisor observes the environment and based on action it will set weights of the neural network. Supervised learning refers to learning by training a model on labelled data. It is a very common approach for predicting an outcome. For example, let’s say we want to predict who is likely to open an email we send. We can use the data from past sends along with the “label” telling us if the recipient opened the email or not.

From there, we can build a training data set with data points about the recipient (location, demographics, past email engagement behaviour) along with the label. Our model trains by trying many different ways to predict the label based on the other data points until it finds the best one. Now that model can be used to predict who will open the next email campaign we send.

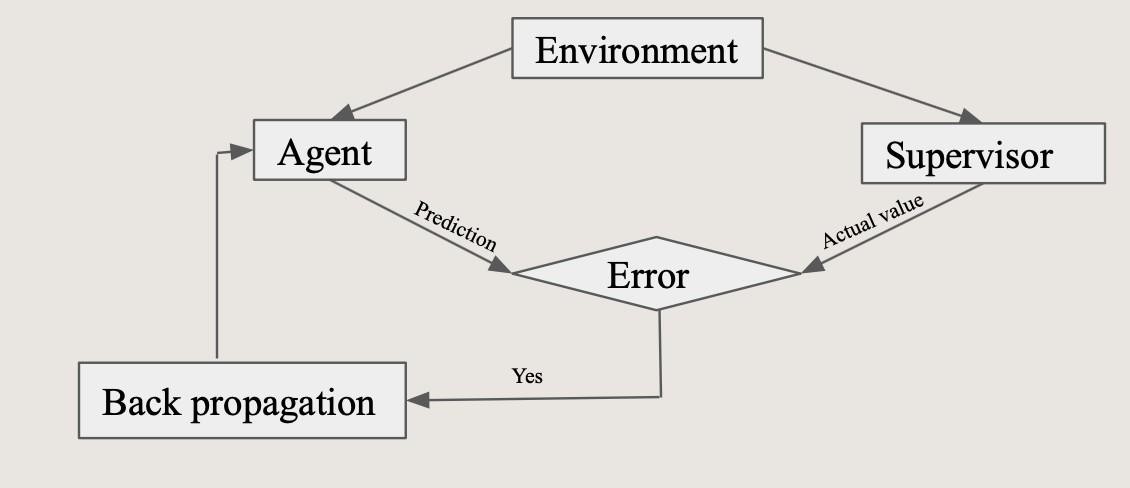
Fig. 4.2 Block diagram of Supervised Learning

Fig. 18 Block diagram of Supervised Learning

50

#### Dataset Preparation

Udemy made a GUI for self-driving car simulation in Unity. We generated data using that Github repo. Data generation took one day because it worked with the images. The software provides two different tracks and one car. Someone can change paths in Unity. The vehicle had three cameras mounted on the top. The camera captures the images of the left side, center, and right side. Throttle value, steering angle, and velocity decided according to the car driving on the track. All the information has been stored in the CSV file. CSV file has image name, throttle value, steering angle, speed, and reverse. Throttle value changes between zero and one. Steering varies from -1 to 1. Here minus one means left side, zero means inline, one means turn right. Speed is in float, whereas reverse holds only true or false. We worked at a college for model development because they have a multicore CPU and GPUs. The model took two days in training. These models are used to predict throttle value and starring value. This model was simulated in Carla. Firstly, actors are created in Carla and then fixed sensors such as Collision detector, Cameras. Camera captures the image and feeds it to the Neural Network for prediction and based on the prediction the car moves in Carla’s environment.

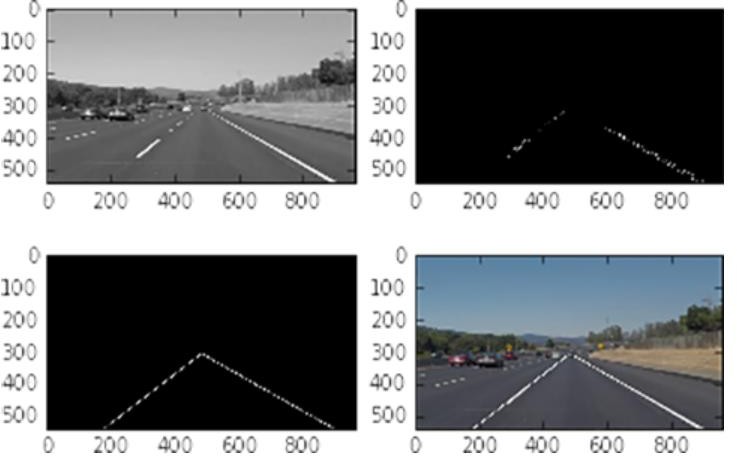


Fig 19 Training Dataset

51

In our driving sequences we have observed that there were very few states where vehicle is trying to change its direction which in turn results into imbalance of dataset. As you can see below in the figure distribution of steering values is not uniform. So we have tried to rebalance our data by increasing rare samples of data. It was done by flipping and duplicating certain set of example which resulted into nice balanced set of examples.

#### Train and Test model

Model has three input images and one output. Based on the road situation, the model will decide speed, steering angle and break. The architecture for the model is the mansion below.

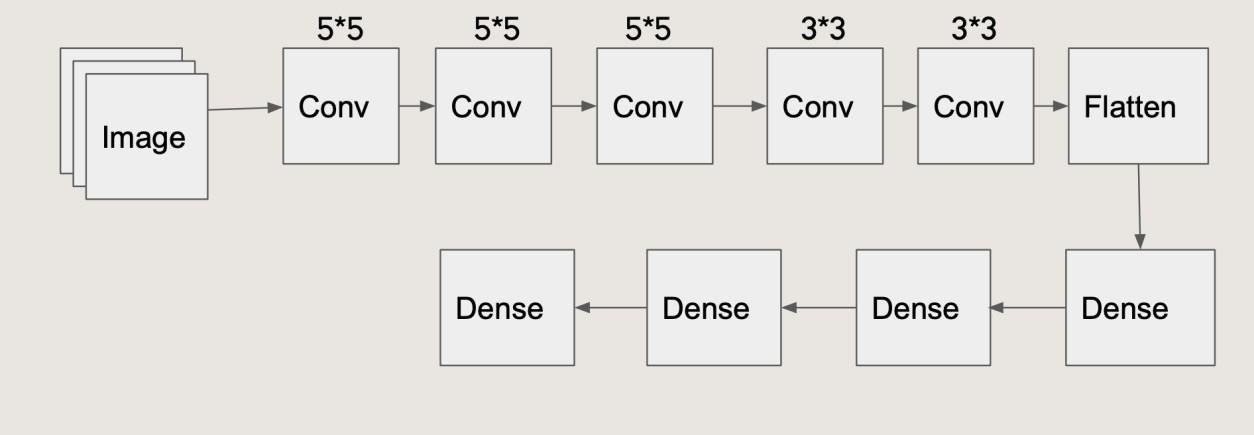


Fig. 20 Architecture

52

#### Test:

In our driving sequences we have observed that there were very few states where vehicle is trying to change its direction which in turn results into imbalance of dataset. As you can see below in the figure distribution of steering values is not uniform. So we have tried to rebalance our data by increasing rare samples of data. It was done by flipping and duplicating certain set of example which resulted into nice balanced set of examples.

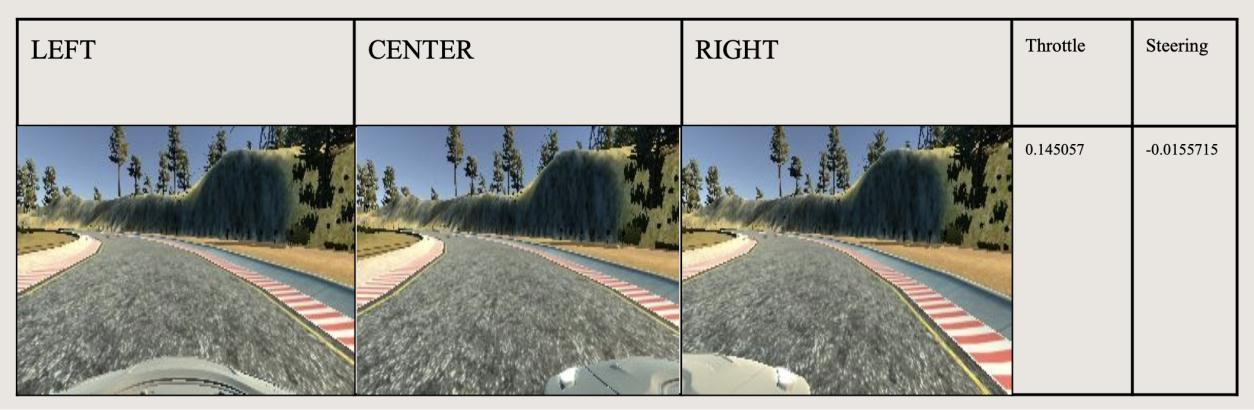


Fig 21 Test Results

It is quite simple solution but we need a lot of data and good network architecture. Issues with this Approach:

* Expert Driving Sequences
* Noisy Outcomes
* No Control Over Controller (It’s just a Black Box)

53



Truthfully, I didn’t understand how the code actually worked, so today I tried to change that.

Below is the main block of code I used yesterday. In particular, I’ve copied the primary function, which is called “draw\_lane\_lines”. Basically, a function is a block of code that takes some input (in this case a photo), manipulates the input in some way, and then outputs the manipulation (in this case the lane lines).

This primary function uses some other helper functions defined elsewhere in the code, but these helper functions are mostly just slightly cleaner ways of consuming the pre-made functions from the libraries I downloaded yesterday (like OpenCV, for example).

Actually, I only focused on the first five, which output the mathematical representation of the lane lines. The last two manipulations just create the visuals so us humans can visually appreciate the math (in other words, these steps aren’t necessary when a self-driving car is actually consuming the outputted data). Thus, based on my research today, I will now attempt to explain the following sequence of image processing events: Input image → 1. Greyscale image, 2. Gaussian Blur, 3. Canny edge detection, 4. Mask edges image, 5. Hough lines → Lane line output

It’s important to remember that an image is nothing more than a bunch of pixels arranged in a rectangle. This particular rectangle is 960 pixels by 540 pixels. The value of each pixel is some combination of red, green, and blue, and is represented by a triplet of numbers, where each number corresponds to the value of one of the colors. The value of each of the colors can range from 0 to 255,.

54

# SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

#### Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application

.it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

#### 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.

55

#### 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. Acceptance testing for Data Synchronization:

* The Acknowledgements will be received by the auditor after the data is received by the cloud server
* The auditors audit operation is done only when there is a request from user
* The Status of data information on the cloud is viewed only by the cloud server

56

#### Risk Analysis

pedestrians and any other obstacles. It will also slow down on junctions, and overtake other vehicles, and execute emergency stops when other options are not available. On the hardware side, some issues are occurring due to insufficient compute power. The car is following lanes but some issues are occurring due to limitations of the hardware. The car will execute an emergency stop when the sensor detects any obstacles using the sensor. The backbone of this project is the LAN server because raspberry pi has insufficient computation power. Car drives automatically very well after training using a high-resolution dataset. Car stops whenever it finds a STOP sign or RED signal and if the ultrasonic sensor detects any object it will break and avoid collision.



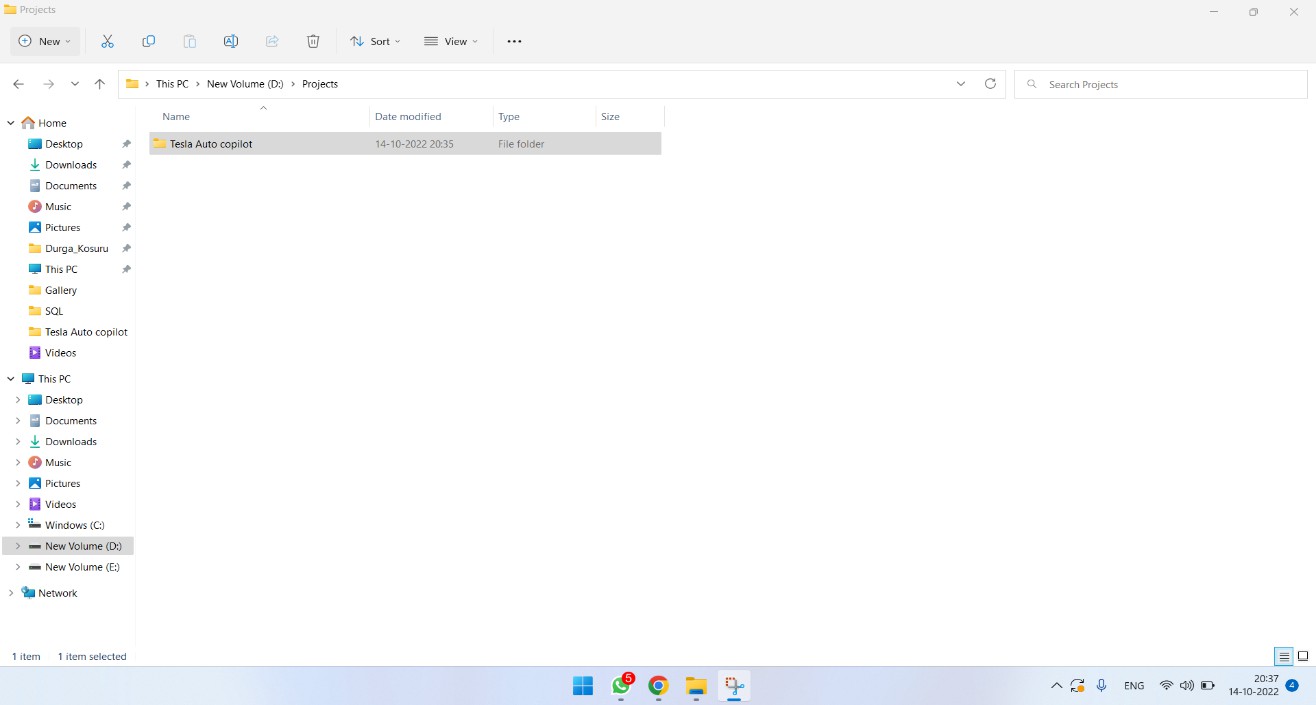
57

#### Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Action** | **Steps** | **Expected Output** | **Actual Output** | **Result** |
| **TUID1** | Object Detection | Detect objects onthe road | Objects Detected | As expected | Pass |
| **TUID2** | Manual Car Control | Control hardware car manually with keyboard over network | Car responsive, controls functioning | As expected | Pass |
| **TUID3** | Transmit camera stream | Transmit camera live stream to server(PC) | Stream transmits | Little lag occurs | Pass |
| **TUID4** | Foreign Vehicle Detection | Detect other vehicles on the road | Other vehicles detected with distance | As expected | Pass |
| **TUID5** | Emergency Stop | Stop Vehicle if obstacle in front, lane change not possible | Vehicle Comes to stop | As expected | Pass |
| **TUID6** | Speed Controller | Adjust vehicle speed close totarget speed | Vehicle speed adjusting | Some inconsistencies | Pass |
| **TUID7** | Lane Following (hardware) | Hardware car drives in lane | Car stays in lane, moves forward | Car follows the lane accurately but with intermediate pauses due to lack of synchrony. | Pass |

58

# RESULTS AND OUTPUTS



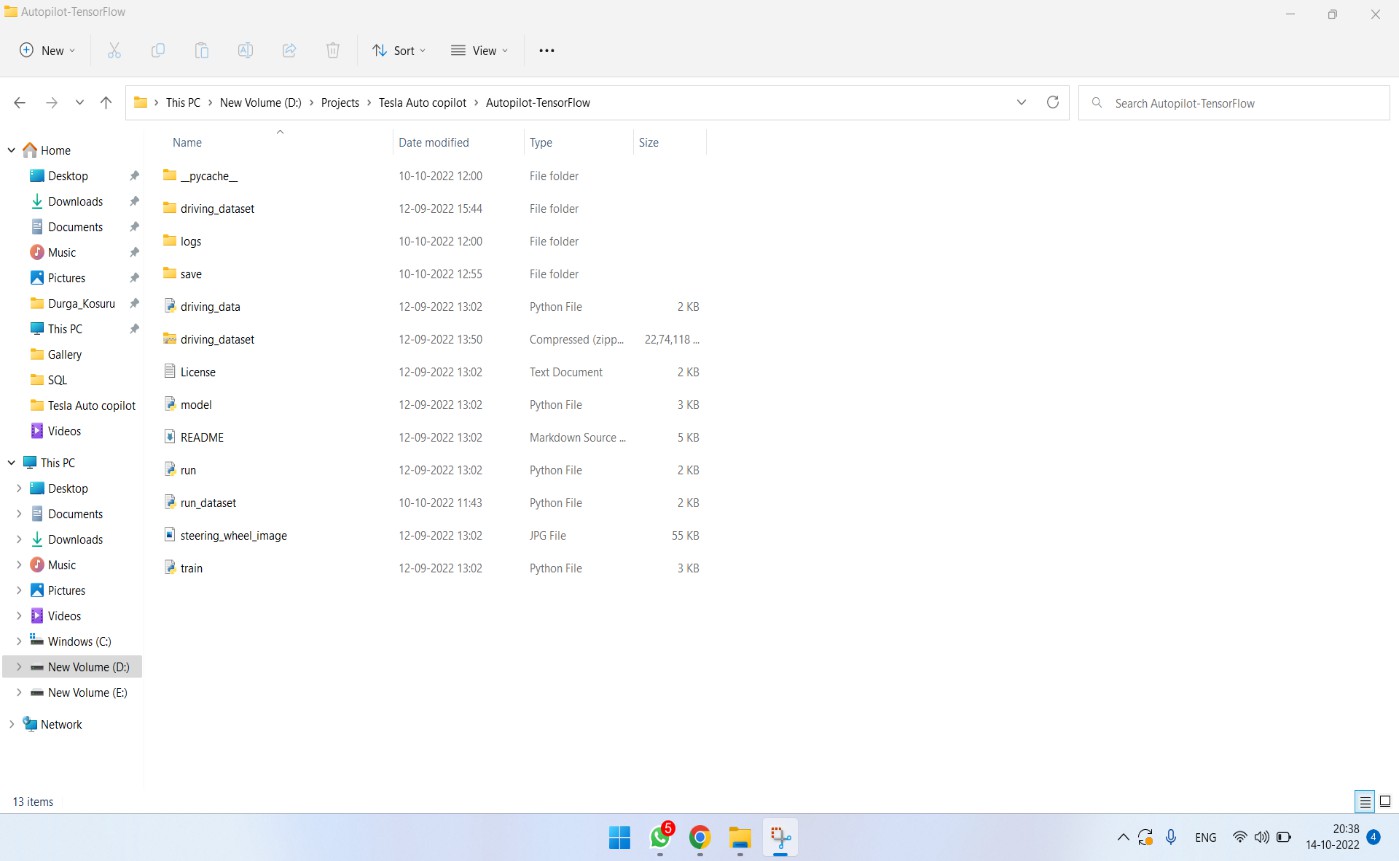


Fig :23 File path

59

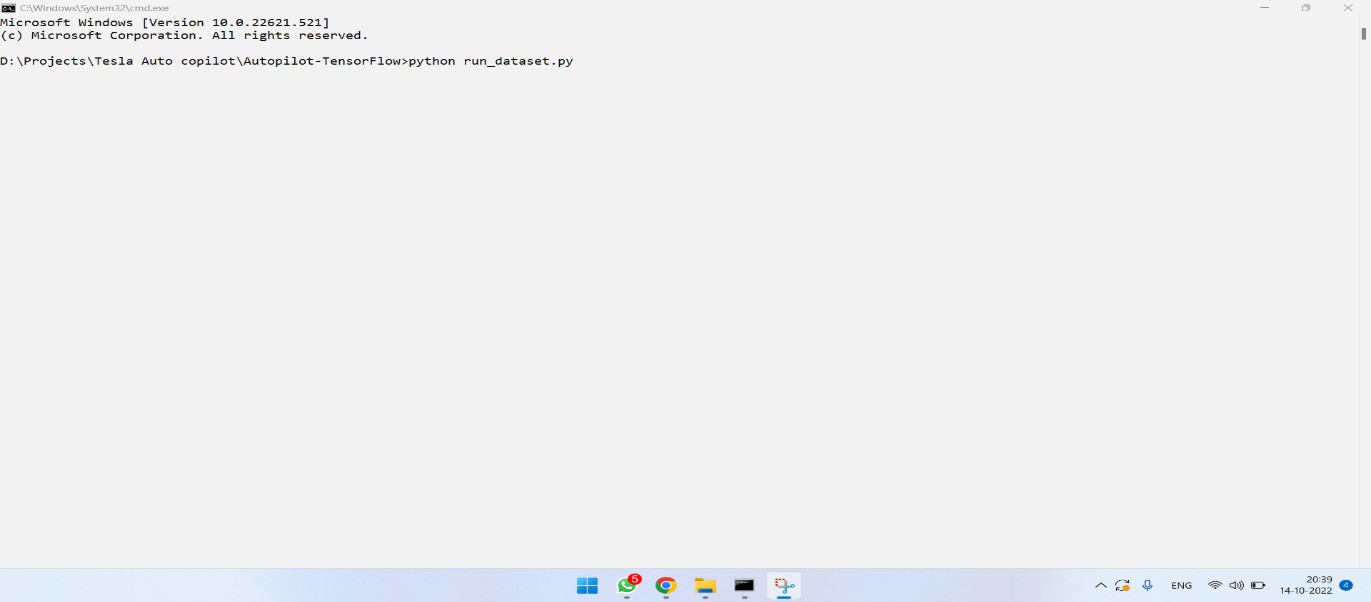


Fig: 24 Execution path

# OUTPUT:

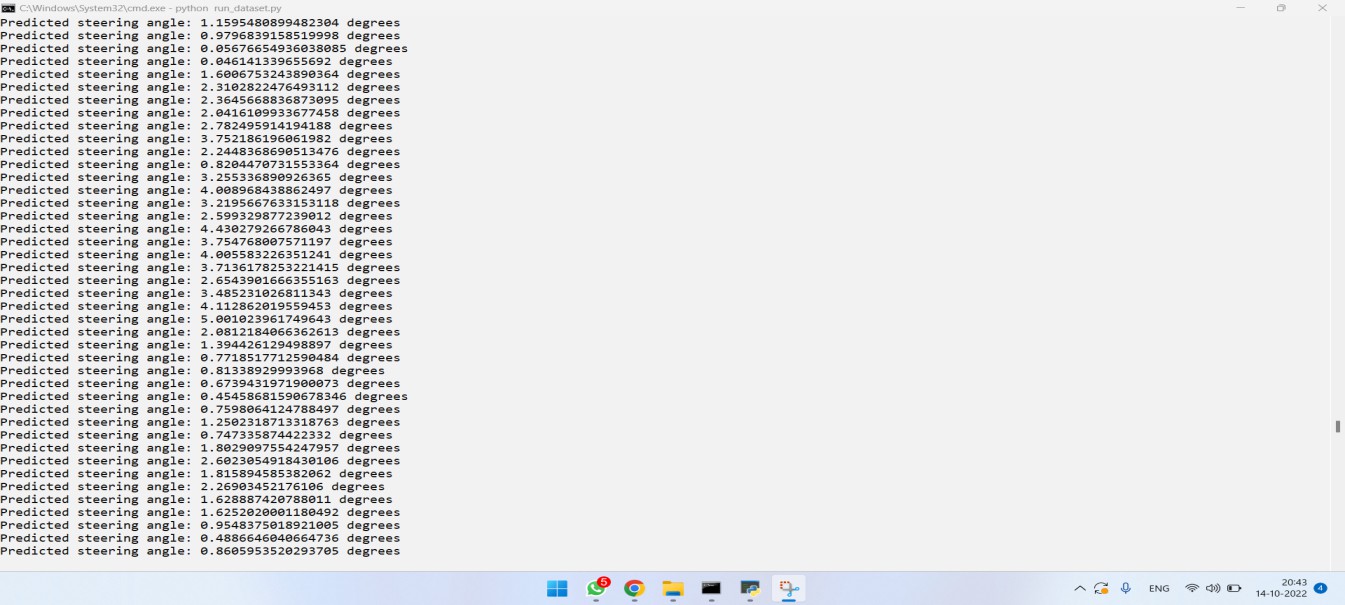
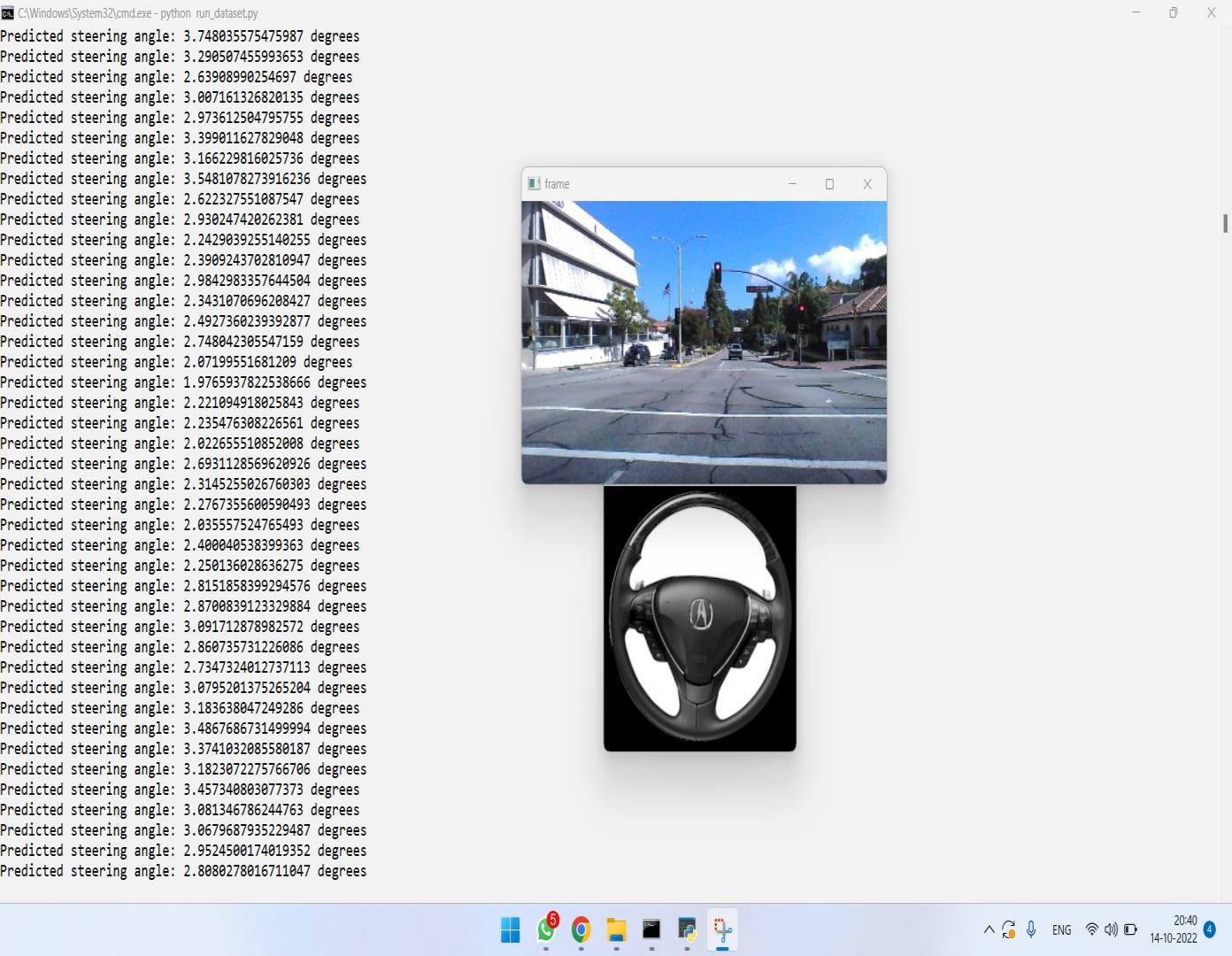
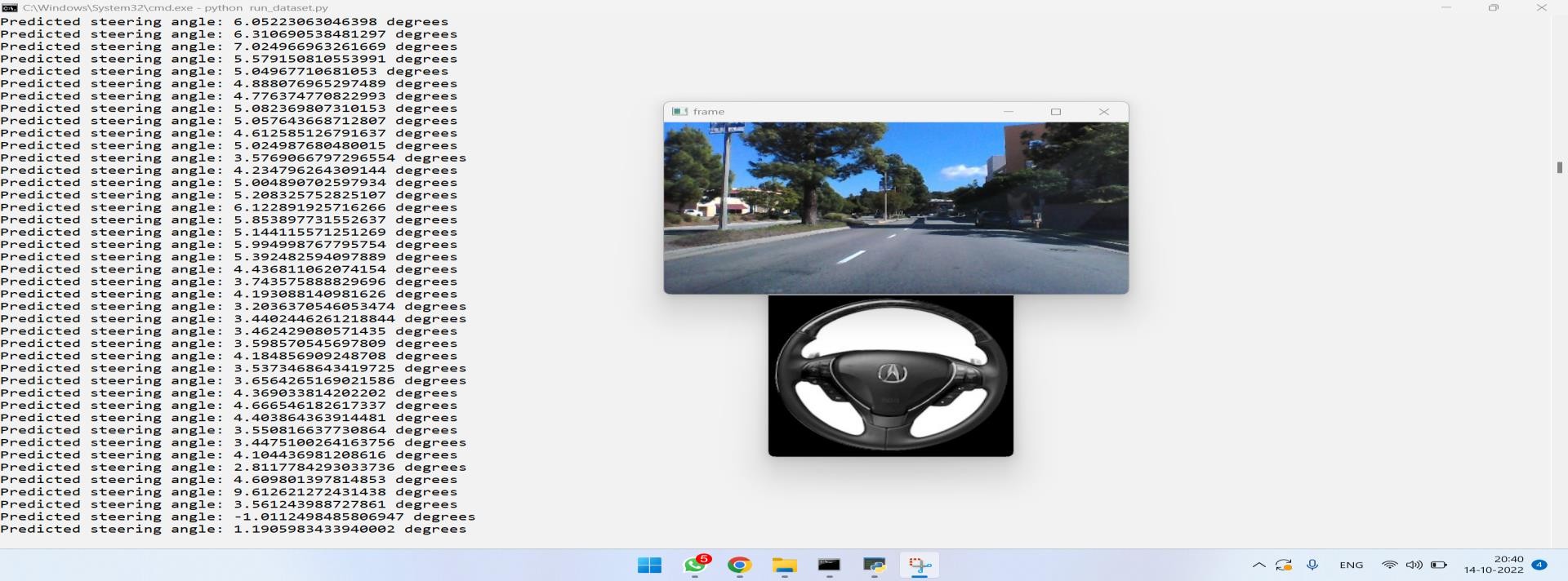


Fig 25 Running the programm

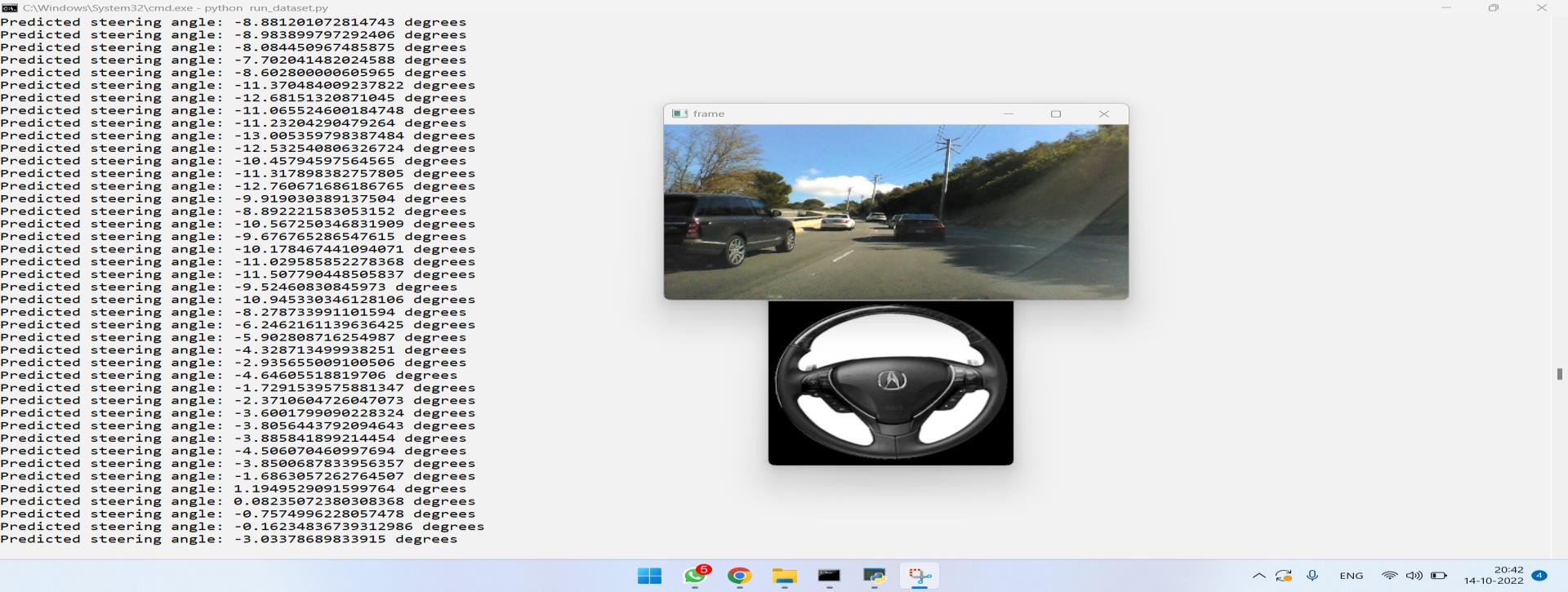
60





61

## Fig 26 : Final output



62

# CONCLUSION

We have empirically demonstrated that CNNs are able to learn the entire task of lane and road following without manual decomposition into road or lane marking detection, semantic abstraction, path planning, and control. A small amount of training data from less than a hundred hours of driving was sufficient to train the car to operate in diverse conditions, on highways, local and residential roads in sunny, cloudy, and rainy conditions. The CNN is able to learn meaningful road features from a very sparse training signal (steering alone).

The system learns for example to detect the outline of a road without the need of explicit labels during training More work is needed to improve the robustness of the network, to find methods to verify the robustness, and to improve visualization of the network-internal processing steps.

In this project i.e “An Intelligent Autopilot System that Learns Drive”, we were able to use the Convolutional Neural Network (CNN) to effectively predict the steering angles and the inner information of CNN can be understood along with how they can be tuned. We have also found that CNN techniques can be used in semantic abstraction, marking detection, path cleaning, and control. In sunny, cloudy, and rainy conditions, a limited amount of training data from less than a hundred hours of driving was adequate to train a vehicle to work in varied conditions on suburban roads, local and highways.

#### Future Scope:

The final output is a simple linear combination of the previous ten neurons, which I thought could be improved. I changed this by applying an inverse tangent function to the linear combination, which I thought made more intuitive sense. The inverse tangent gives the network a way to sort of tool to “recover” the angle of the curvature from the visual data, instead of having to relearn a way to convert slopes or tangents into radian measures. In practice, it really didn’t make a difference, but I kept it just for fun.

This system is work perfect in the simulated environment like Carla but face some difficulties in the

63

new environment. All the sensor work according to requirement and stop where it needed. This system will forcefully stop at the some unfamiliar situations like image resolutions is not cleared or person on the way. The Car fatalities will not occurred because of working algorithm. Path prediction is done by Kalman Filter and based on that further path is decided. Light and brightness of the surrounding is affect driving because of Camera. So to get good performance high quality Camera is required. Computation power is also limited therefore delay can easily noticed. This can be solved using powerful mobile computation devices such as Jetson, but it is costly.

#### Follow-up Activities:

* Improve vehicle hardware
* Improve Vehicle Controller
* Deploy more sensors such as LIDAR, RADAR which enables us in tracking down objects precisely.
* Audio sounds
* Car sound
* Break sound
* collision sound
* Horn sound
* Predict move of Objects.
* Vehicle to vehicle communication using edge computation (M2M)

64

# BIBILOGRAPHY

1. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551, Winter 1989. URL: [http://yann.lecun.org/exdb/publis/pdf/lecun-89e.pdf.](http://yann.lecun.org/exdb/publis/pdf/lecun-89e.pdf)
2. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. URL: <http://papers.nips.cc/paper/> 4824-imagenet-classification-with- deep-convolutional-neural-networks. pdf.
3. L. D. Jackel, D. Sharman, Stenard C. E., Strom B. I., , and D Zuckert. Optical character recognition for self-service banking. AT&T Technical Journal, 74(1):16–24, 1995.
4. Large scale visual recognition challenge (ILSVRC). URL: <http://www.image-net.org/> challenges/LSVRC/. [5] Net-Scale Technologies, Inc. Autonomous off-road vehicle control using end-to-end learning, July 2004. Final technical report. URL: [http://net-scale.com/doc/net-](http://net-scale.com/doc/net-scale-dave-report.pdf) [scale-dave-report.pdf.](http://net-scale.com/doc/net-scale-dave-report.pdf)

[6] Dean A. Pomerleau. ALVINN, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, 1989. URL: [http://repository.cmu.edu/cgi/viewcontent.](http://repository.cmu.edu/cgi/viewcontent) cgi?article=2874&context=compsci. [7] Wikipedia.org. DARPA LAGR program. <http://en.wikipedia.org/wiki/DARPA_LAGR_> Program.

[8] Danwei Wang and Feng Qi. Trajectory planning for a four-wheel-steering vehicle. In Proceedings of the 2001 IEEE International Conference on Robotics & Automation, May 21– 26 2001. URL: http: //[www.ntu.edu.sg/home/edwwang/confpapers/wdwicar01.pdf.](http://www.ntu.edu.sg/home/edwwang/confpapers/wdwicar01.pdf) [9] DAVE

2 driving a lincoln. URL: https://drive.google.com/open?id= 0B9raQzOpizn1TkRIa241ZnBEcjQ.

1. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun “Faster R-CNN: Towards Real- Time Object Detection with Region Proposal Networks” June 2015 IEEE Transactions on Pattern Analysis and Machine Intelligenc

65

1. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi “You Only Look Once: Unified, Real-Time Object Detection” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
2. Raspberry pi. February 2012. https://en.wikipedia.org/wiki/Raspberry\_Pi [13]Piborg. HC SR04 Ultrasonic sensor. https:/[/www](http://www.piborg.org/sensors-1136/hc-sr04).[piborg.org/sensors-1136/hc-sr04](http://www.piborg.org/sensors-1136/hc-sr04)

[14]Keras. Keras Simple. Flexible. Powerful. https://keras.io

[15]DC Motor. 2010. https://en.wikipedia.org/wiki/DC\_motor [16]Etcher. July 2017. https://en.wikipedia.org/wiki/Etcher\_(software) [18]Tensorflow. March 2020. <https://www.tensorflow.org/>

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. URL: <http://papers.nips.cc/paper/> 4824-imagenet-classification- with-deep-convolutional-neural-networks. pdf.

[3] L. D. Jackel, D. Sharman, Stenard C. E., Strom B. I., , and D Zuckert. Optical character recognition for self-service banking. AT&T Technical Journal, 74(1):16–24, 1995.

[4] Large scale visual recognition challenge (ILSVRC). URL: <http://www.image-net.org/> challenges/LSVRC/. [5] Net-Scale Technologies, Inc. Autonomous off-road vehicle control using end-to-end learning, July 2004. Final technical report. URL: [http://net-scale.com/doc/net-](http://net-scale.com/doc/net-scale-dave-report.pdf) [scale-dave-report.pdf.](http://net-scale.com/doc/net-scale-dave-report.pdf)

[6] Dean A. Pomerleau. ALVINN, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, 1989. URL: [http://repository.cmu.edu/cgi/viewcontent.](http://repository.cmu.edu/cgi/viewcontent) cgi?article=2874&context=compsci. [7] Wikipedia.org. DARPA LAGR program. <http://en.wikipedia.org/wiki/DARPA_LAGR_> Program.

1. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran

66

Associates, Inc., 2012. URL: <http://papers.nips.cc/paper/> 4824-imagenet-classification-with- deep-convolutional-neural-networks. pdf.

1. L. D. Jackel, D. Sharman, Stenard C. E., Strom B. I., , and D Zuckert. Optical character recognition for self-service banking. AT&T Technical Journal, 74(1):16–24, 1995.
2. Large scale visual recognition challenge (ILSVRC). URL: <http://www.image-net.org/> challenges/LSVRC/. [5] Net-Scale Technologies, Inc. Autonomous off-road vehicle control using end-to-end learning, July 2004. Final technical report. URL: [http://net-scale.com/doc/net-](http://net-scale.com/doc/net-scale-dave-report.pdf) [scale-dave-report.pdf.](http://net-scale.com/doc/net-scale-dave-report.pdf)

[6] Dean A. Pomerleau. ALVINN, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, 1989. URL: [http://repository.cmu.edu/cgi/viewcontent.](http://repository.cmu.edu/cgi/viewcontent) cgi?article=2874&context=compsci. [7] Wikipedia.org. DARPA LAGR program. <http://en.wikipedia.org/wiki/DARPA_LAGR_> Program.

1. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. URL: <http://papers.nips.cc/paper/> 4824-imagenet-classification-with- deep-convolutional-neural-networks. pdf.
2. L. D. Jackel, D. Sharman, Stenard C. E., Strom B. I., , and D Zuckert. Optical character recognition for self-service banking. AT&T Technical Journal, 74(1):16–24, 1995.
3. Large scale visual recognition challenge (ILSVRC). URL: <http://www.image-net.org/> challenges/LSVRC/. [5] Net-Scale Technologies, Inc. Autonomous off-road vehicle control using end-to-end learning, July 2004. Final technical report. URL: [http://net-scale.com/doc/net-](http://net-scale.com/doc/net-scale-dave-report.pdf) [scale-dave-report.pdf.](http://net-scale.com/doc/net-scale-dave-report.pdf)
4. Dean A. Pomerleau. ALVINN, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, 1989. URL: [http://repository.cmu.edu/cgi/viewcontent.](http://repository.cmu.edu/cgi/viewcontent) cgi?article=2874&context=compsci. [7] Wikipedia.org. DARPA LAGR program. <http://en.wikipedia.org/wiki/DARPA_LAGR_> Program.

, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. Remarkable item identification: A benchmark. IEEE

67

Trans. Picture Processing, 24(12):5706–5722, 2015.

* 1. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic picture division with profound convolutional nets, atrous convolution, and completely associated crfs. IEEE exchanges on design examination and machine insight, 40(4):834– 848, 2017.
  2. Shuhan Chen, Xiuli Tan, Ben Wang, and Xuelong Hu. Invert consideration for remarkable item discovery. In Proceedings of the European Conference on Computer Vision (ECCV), pages 234– 250, 2018.
  3. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A huge scope progressive picture data set. In 2009 IEEE gathering on PC vision and example acknowledgment, pages 248–255. IEEE, 2009.
  4. Zijun Deng, Xiaowei Hu, Lei Zhu, Xuemiao Xu, Jing Qin, Guoqiang Han, and Pheng-Ann Heng. R3net: Recurrent remaining refinement network for saliency discovery. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, pages 684–690. AAAI Press, 2018.
  5. Marc Ehrig and Jer' ome Euzenat. Loosened up exactness and re-ˆ call for cosmology coordinating. In Proc. K-Cap 2005 workshop on Integrating philosophy, pages 25–32. No business manager., 2005.
  6. Deng-Ping Fan, Ming-Ming Cheng, Yun Liu, Tao Li, and Ali Borji. Design measure: another approach to assess forefront maps. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4548–4557, 2017.
  7. Mengyang Feng, Huchuan Lu, and Errui Ding. Mindful criticism network for limit mindful striking article location. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1623–1632, 2019.
  8. Xavier Glorot and Yoshua Bengio. Understanding the trouble of prepareng profound feedforward neural organizations. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS, pages 249–256, 2010.
  9. Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. Picture investigation utilizing numerical morphology. IEEE exchanges on design investigation and machine insight,

68

(4):532–550, 1987.

* 1. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.
  2. Qibin Hou, Ming-Ming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip Torr. Deeply supervised salient object detection with short connections. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5300–5309, 2017.
  3. Xiaowei Hu, Lei Zhu, Jing Qin, Chi-Wing Fu, and PhengAnn Heng. Recurrently aggregating deep features for salient object detection. In AAAI, pages 6943–6950, 2018.
  4. Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2261–2269, 2017.
  5. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint, 2014.
  6. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
  7. Guanbin Li and Yizhou Yu. Visual saliency detection based on multiscale deep cnn features. IEEE Transactions on Image Processing, 25(11):5012–5024, 2016.
  8. Yin Li, Xiaodi Hou, Christof Koch, James M Rehg, and Alan L Yuille. The secrets of salient object segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 280–287, 2014.
  9. Jie Liang, Jun Zhou, Lei Tong, Xiao Bai, and Bin Wang. Material based salient object detection from hyperspectral images. Pattern Recognition, 76:476–490, 2018.
  10. Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Autodeeplab: Hierarchical neural architecture search for semantic image segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,

69

pages 82–92, 2019.

* 1. Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Jiashi Feng, and Jianmin Jiang. A simple pooling-based design for realtime salient object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3917–3926, 2019.
  2. Nian Liu, Junwei Han, and Ming-Hsuan Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3089–3098, 2018.
  3. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015.
  4. Shijian Lu and Joo-Hwee Lim. Saliency modeling from image histograms. In European Conference on Computer Vision, pages 321–332. Springer, 2012.
  5. Shijian Lu, Cheston Tan, and Joo-Hwee Lim. Robust and efficient saliency modeling from image co-occurrence histograms. IEEE transactions on pattern analysis and machine intelligence, 36(1):195–201, 2013.
  6. Zhiming Luo, Akshaya Mishra, Andrew Achkar, Justin Eichel, Shaozi Li, and Pierre- Marc Jodoin. Non-local deep features for salient object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6593–6601, 2017.
  7. Ke Ma, Zhixin Shu, Xue Bai, Jue Wang, and Dimitris Samaras. Docunet: Document image unwarping via a stacked unet. In CVPR, pages 4700–4709, 2018.
  8. Ran Margolin, Lihi Zelnik-Manor, and Ayellet Tal. How to evaluate foreground maps. 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, 2014.
  9. Vida Movahedi and James H Elder. Design and perceptual validation of performance measures for salient object segmentation. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, pages 49–56. IEEE, 2010.
  10. Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In European conference on computer vision, pages 483–499. Springer, 2016.

70

* 1. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In Autodiff workshop on NIPS, 2017.
  2. Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundaryaware salient object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7479–7489, 2019.
  3. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. Unet: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.
  4. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large- scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
  5. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.
  6. Zhiqiang Tang, Xi Peng, Shijie Geng, Lingfei Wu, Shaoting Zhang, and Dimitris Metaxas. Quantized densely connected u-nets for efficient landmark localization. In Proceedings of the European Conference on Computer Vision (ECCV), pages 339–354, 2018.
  7. Zhiqiang Tang, Xi Peng, Shijie Geng, Yizhe Zhu, and Dimitris N Metaxas. Cu-net: coupled u-nets. arXiv preprint arXiv:1808.06521, 2018.
  8. Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 136–145, 2017.

71