**Steering Angle Prediction for Autonomous Vehicles (Self Driving Car) Using Deep Learning**

**A Project Report**

Submitted in Partial Fulfillment of the Requirements for the

Award of the Degree of

**Bachelor of Technology**

In

**Electronics and Communication**

Submitted by

**Devashray Roll No.-1804095**

**Rajesh Kumar Roll No.-1804138**

**Pushkar Dhaker Roll No.-1804156**

Under the supervision of

**Dr. Rajarshi Bhattacharya**

Assistant Professor



**Department of Electronics and Communication Engineering**

NATIONAL INSTITUTE OF TECHNOLOGY PATNA

PATNA- 800 005

**Table of Contents**

1. Objective
2. Introduction
3. Literature Review
4. Udacity Simulator And Dataset
5. Approaches To the Problem
6. Challenges And Issues
7. Results
8. Conclusion
9. References
10. **OBJECTIVE**

The purpose of a Self-driving car project is to build a better autonomous driver. The car should be able to drive itself without falling off the track, with accelerating and braking at appropriate places. This chapter covers the problem statement of the project in brief and the higher-level solution approach used.

1. **INTRODUCTION**

# Problem Definition: -

Udacity released an open-source simulator for self-driving cars to depict a real-time environment. The challenge is to mimic the driving behavior of a human on the simulator with the help of a model trained by deep neural networks. The concept is called Behavioral Cloning, to mimic how a human drive.

The simulator contains two tracks and two modes, when we are going to launch the simulator namely, training mode and autonomous mode. The dataset is generated from the simulator by the user, driving the car in training mode. This dataset is also known as the “good” driving data. This is followed by testing on the track, seeing how the deep learning model performs after being trained by that user data. Another challenge is to generalize the performance on different tracks. That means, training the model using the dataset created on one of the tracks, and testing it on the other track of the simulator.

# Technologies Used: -

Technologies that are used in the implementation of this project and the motivation behind using these are described in this section.

**TensorFlow:** - This is open-source library developed by google, for this project we will use TensorFlow API called keras to build our model, that will predict the steering angle of autonomous vehicles.

**NumPy:** - NumPy can be used to perform a wide variety of mathematical operations on arrays.

**OpenCV:** - OpenCV is an open-source library for computer vision, machine learning, and image processing.

1. **LITERATURE REVIEW**

Steering angle prediction algorithms are divided into two categories as shown in **Fig. 1**.



**Computer vision-based approach**

**Neural network-based approach**

It follows two steps:

-Road boundary extraction (road region extraction)

- Steering angle computation.

End-to-End learning approach that requires:

* Lots of data to train the network to

make good predictions

* High-cost computing requirements

**Fig 1:** The two main approaches for steering angle prediction

1. **Computer Vision Based Approach**

**Augmentation And Image Pre-Processing: -**

The biggest challenge was generalizing the behavior of the car on Track\_2 which it was never trained for. In a real-life situation, we can never train a self-driving car model for every track possible, as the data will be too huge to process. Also, it is not possible to gather the dataset for all the weather conditions and roads. Thus, there is a need to come up with an idea of generalizing the behavior on different tracks.

This problem is solved using image preprocessing and augmentation techniques, which will be discussed in the following section:

# Crop

The images in the dataset have relevant features in the lower part where the road is visible. The external environment above a certain image portion will never be used to determine the output and thus can be cropped. Approximately, 30% of the top portion of the image is cut and passed in the training set. The snippet of code and transformation of an image after cropping and resizing it to original image can be seen in Figure 2.

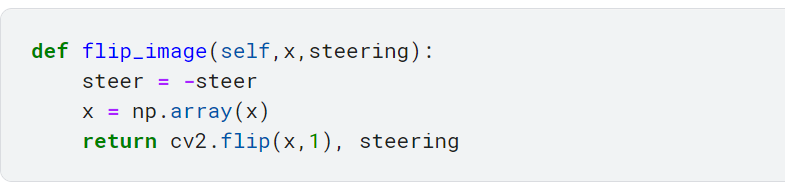


# 

**Fig 2. Crop image (continued)**

* **Flip (horizontal)**

The image is flipped horizontally (i.e., a mirror image of the original image is passed to the dataset). The motive behind this is that the model gets trained for similar kinds of turns on opposite sides too. This is important because Track\_1 includes only left turns. The snippet of code and transformation of an image after flipping it can be seen in Figure 3.







# Fig 3. Flip image

* **Shift (horizontal / vertical)**

The image is shifted by a small amount, here it is vertical shift in Figure 4 and horizontal shift in Figure 5.



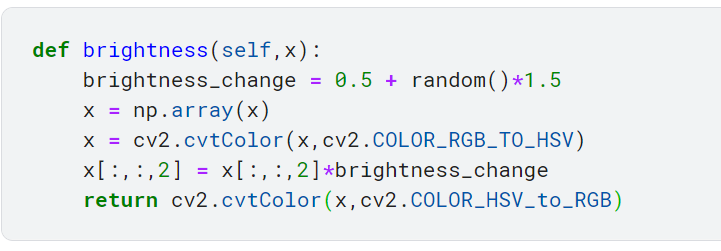
# Fig 4. Shift image vertical





**Fig 5. Shift image horizontal**

* **Brightness**

To generalize to the weather conditions with bright sunny day or cloudy, lowlight conditions, the brightness augmentation can prove to be very useful. The code snippet and increase of brightness can be seen in Figure 7. Similarly, I have randomly also lowered down the level of brightness for other conditions.

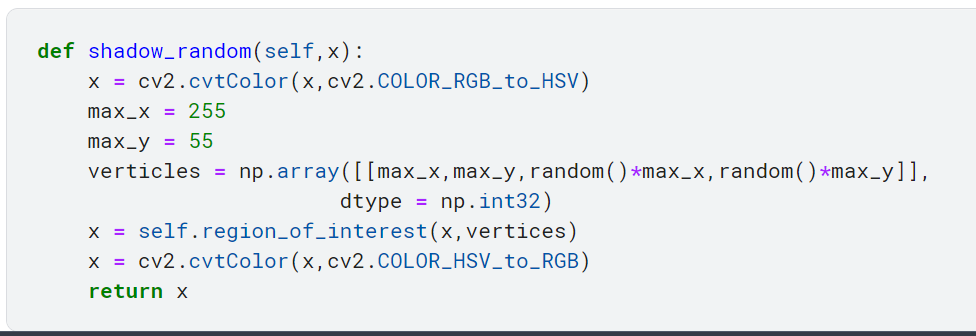
# Fig 6. Brightness increase

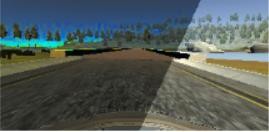
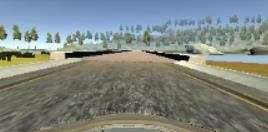




# Fig 7. Brightness increase (continued)

* **Shadows**

Even after taking into considerations the light conditions, there are still chances that there are shadows on the road. This will give an instance of half lit and half lowlight scenes in the image. To cast random shadows and solve this shadow fitting problem, this augmentation is applied on the dataset. A sample shadow augmentation with its code snippet is shown in the Figure 8.



# 

# Fig 8. Random shadows

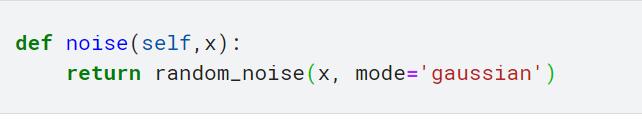
* **Random blur**

To take care of the distortion effect in the camera while capturing the images, this augmentation is used as an image captured is not clear every time. Sometimes, the

camera goes out of focus, but the car still needs to fit that condition and keep the car steady. This random blur augmentation can take such scenarios into consideration. The sample code snippet and the transformation can be seen in Figure 9.

# Fig 9. Random blur

* **Noise**

This adds random noise to the images by taking into consideration the unclean conditions by simulating dust or dirt particles and distortions while capturing the image. The sample of the code snippet and transformation can be seen in Figure 10.





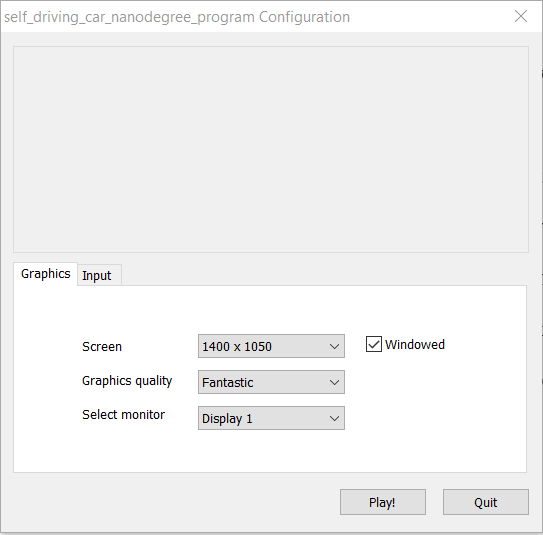
# Fig 10. Random Noise

# Convolutional Neural Networks (CNN)

* CNN is a type of feed-forward neural network computing system that can be used to learn from input data. Learning is accomplished by determining a set of weights or filter values that allow the network to model the behavior according to the training data.
* The desired output and the output generated by CNN initialized with random weights will be different. This difference (generated error) is backpropagated through the layers of CNN to adjust the weights of the neurons, which in turn reduces the error and allows us produce output closer to the desired one.
* CNN is good at capturing hierarchical and spatial data from images. It utilizes filters that look at regions of an input image with a defined window size and map it to some output. It then slides the window by some defined stride to other regions, covering the whole image. Each convolution filter layer thus captures the properties of this input image hierarchically in a series of subsequent layers, capturing the details like lines in image, then shapes, then whole objects in later layers.
* CNN can be a good fit to feed the images of a dataset and classify them into their respective classes.

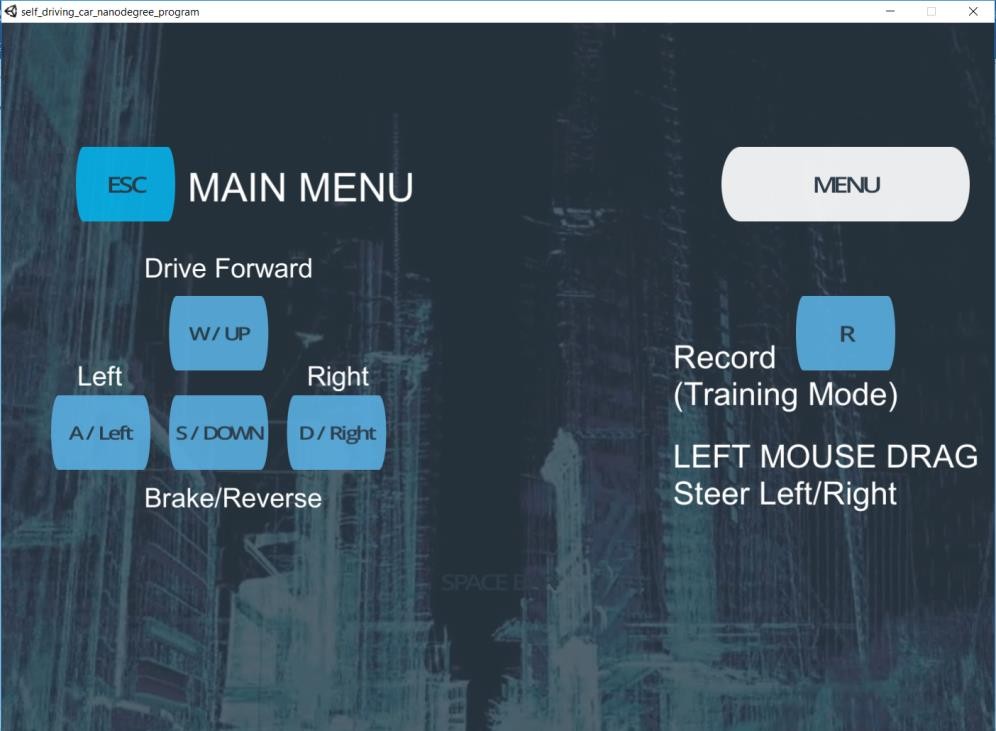
1. **UDACITY SIMULATOR AND DATASET**

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



# Fig 11. Configuration screen

The graphics and input configurations can be changed according to user preference and machine configuration as shown in Figure 11. The user pushes the “Play!” button to enter the simulator user interface. You can enter the Controls tab to explore the keyboard controls, quite similar to a racing game which can be seen in Figure 12.



# Fig. 12: Controls Configuration



**Fig. 13: First Screen**

The first actual screen of the simulator can be seen in Figure.13 and its components are discussed below. The simulator involves two tracks. One of them can be considered as simple and another one as complex that can be evident in the screenshots attached in Figure.14 and Figure.15. The word “simple” here just means that it has fewer curvy tracks and is easier to drive on, refer Figure.14



# Fig 14. Track\_1

The “complex” track has steep elevations, sharp turns, shadowed environment, and is tough to drive on, even by a user doing it manually. Please refer Figure.15



# Fig 15. Track\_2

There are two modes for driving the car in the simulator: (1) Training mode and (2) Autonomous mode.

The training mode gives you the option of recording your run and capturing the training dataset. The small red sign at the top right of the screen in the Figure 15 depicts the car is being driven in training mode.

The autonomous mode can be used to test the models to see if it can drive on the track without human intervention. Also, if you try to press the controls to get the car back on track, it will immediately notify you that it shifted to manual controls. The mode screenshot can be as seen in Figure 16.



# Fig 16. Autonomous mode

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

* The simulator has built the driving features in such a way that it simulates that there are three cameras on the car. The three cameras are in the center, right and left on the front of the car, which captures continuously when we record in the training mode.
* The stream of images is captured, and we can set the location on the disk for saving the data after pushing the record button. The image set are labelled in a sophisticated manner with a prefix of center, left, or right indicating from which camera the image has been captured.
* Along with the image dataset, it also generates a ***datalog.csv*** file. This file contains the image paths with corresponding steering angle, throttle, brakes, and speed of the car at that instance.

A few images from the dataset are shown in Figure 17.



Center0001 Right0001 Left0001



Center0099 Right0099 Left0099

# Fig 17. Dataset sample

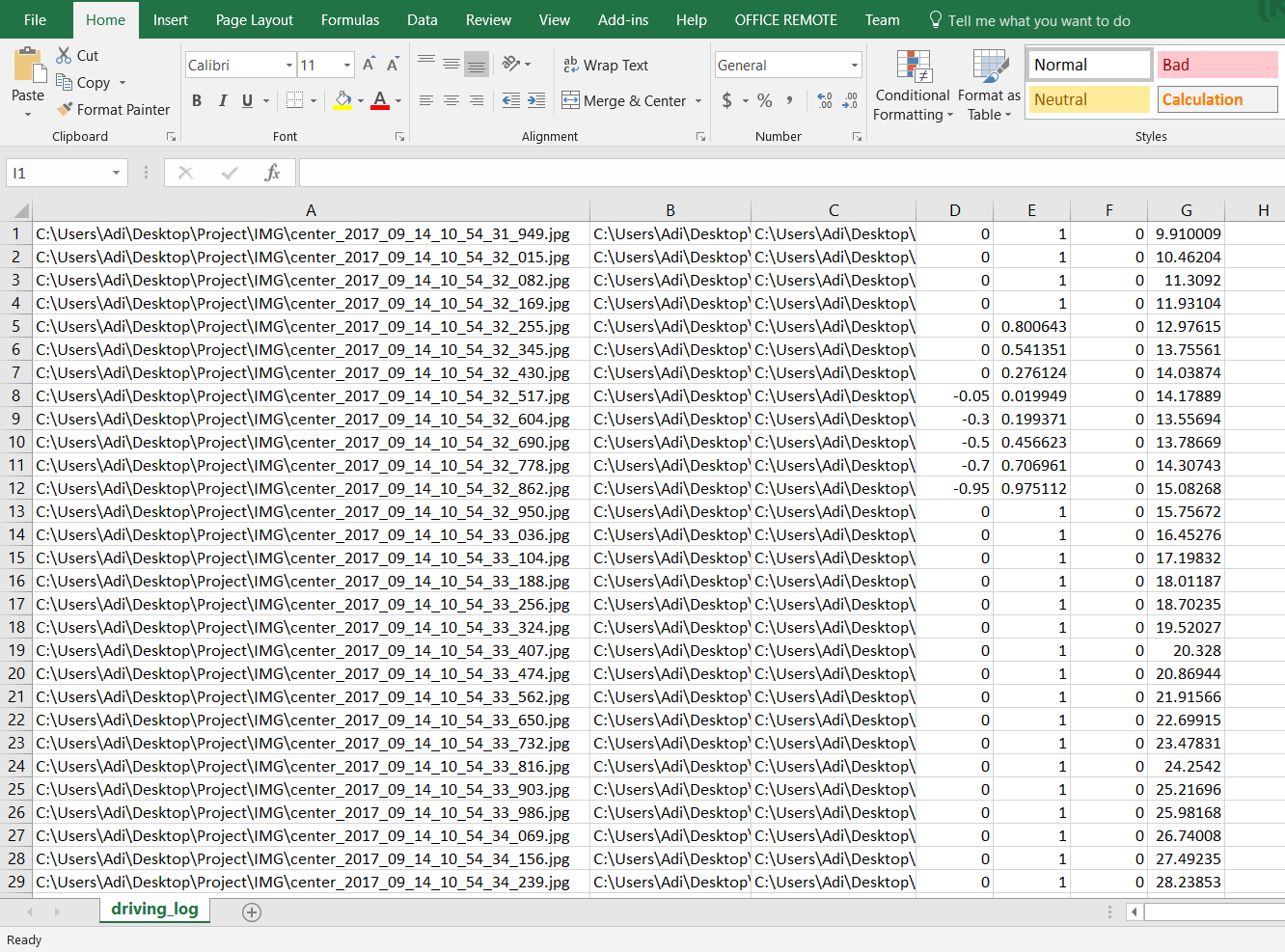
A sample of datalog.csv file is shown in Figure 18.

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

**Column 4:** contains the steering angle

Column value as 0 depicts straight, positive value is right turn and negative value is left turn.

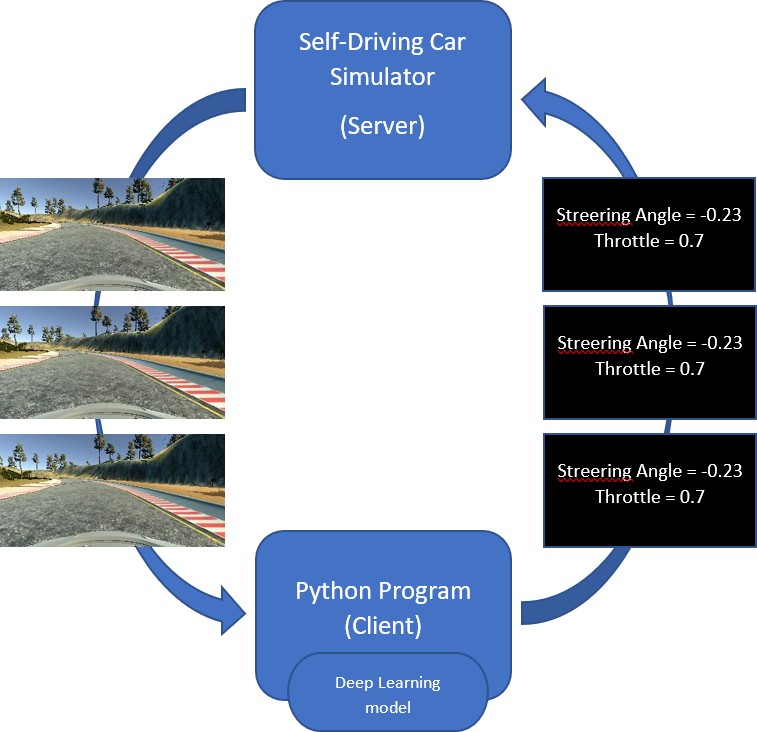
**Column 5:** contains the throttle or acceleration at that instance **Column 6:** contains the brakes or deceleration at that instance **Column 7:** contains the speed of the vehicle



**Fig 18. driving\_log.csv**

1. **APPROACHES TO THE PROBLEM**

The high-level architecture of the implementation can be seen in Figure 19.



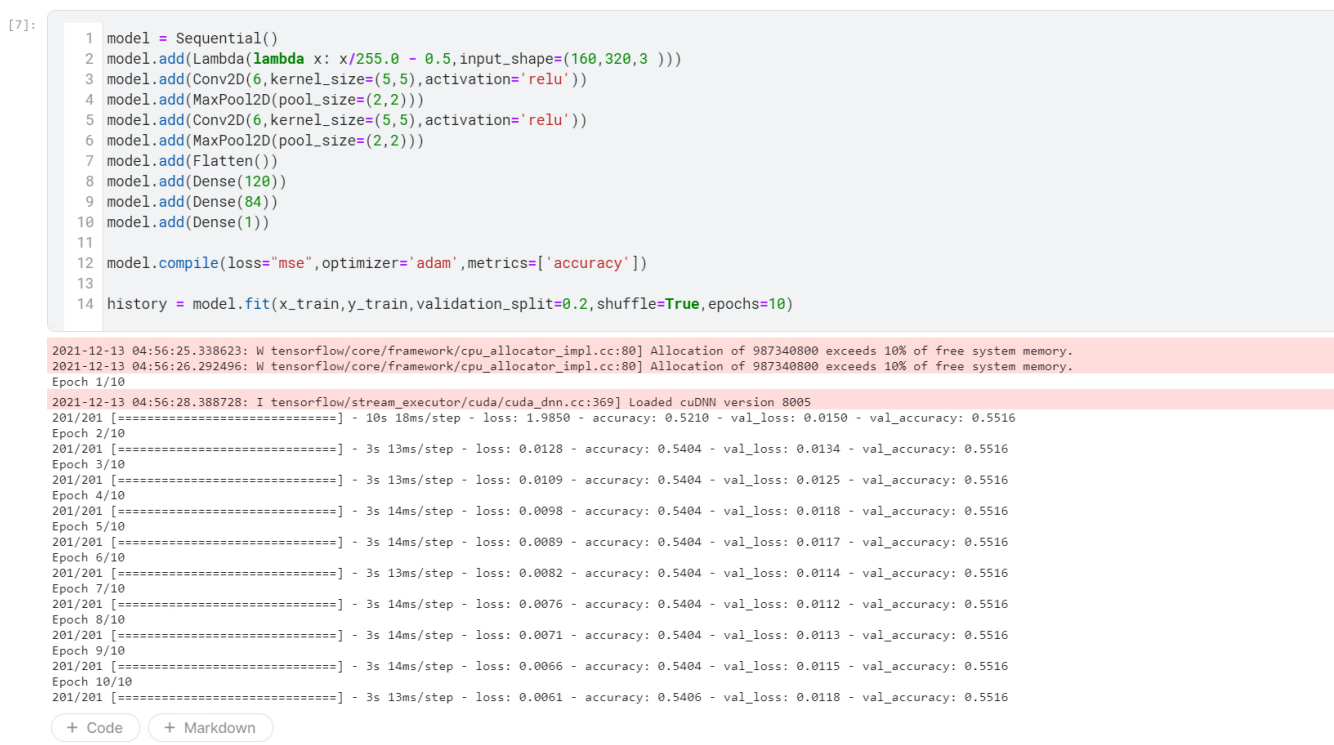
# Fig 19. Implementation Architecture

The problem is solved in the following steps:

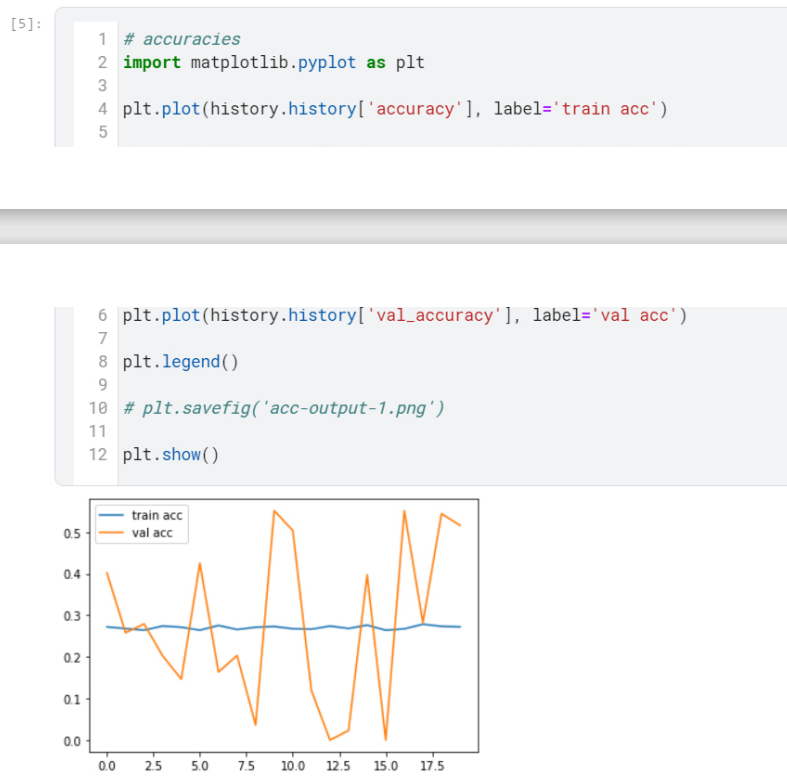
* + - The simulator can be used to collect data by driving the car in the training mode using a joystick or keyboard, providing the so called “good-driving” behavior input data in form of a driving log (.csv file) and a set of images. The simulator acts as a server and pipes these images and data log to the Python client.
    - The client (Python program) is the machine learning model built using Deep Neural Networks. These models are developed on Keras (a high-level API over TensorFlow). Keras provides sequential models to build a linear stack of network layers. Such models are used in the project to train over the datasets as the second Step.
    - Once the model is trained, it provides steering angles and throttle to drive in an autonomous mode to the server (simulator).
    - These modules, or inputs, are piped back to the server and are used to drive the car autonomously in the simulator and keep it from falling off the track.

****

 Model-1

 Model-2

1. **RESULTS**

**** Model-1(accuracy & loss)

****

**** Model-2 (accuracy & loss)****

1. **CHALLENGES AND ISSUES**

**a. SLOW IN UNKNOWN ENVIRONMENT**

It is found that CNN driving at higher speed remains a challenge. Researchers argue that CNN takes time to recover from bad mistakes, and it is found to be slow in responding to unknown scenarios. Therefore, we propose a hybrid of CNN and another DL algorithm to overcome its shortcoming in steering control at high speed.

**b. HIGH COMPUTATIONAL COST AND MEMORY CONSUMPTION**

In respect to the DL structure, CNN requires high computational cost and consumes a lot of memory space. This constitutes a challenge to steering control of the AVs as it has negative implications on both the hardware and software, which in turn makes the vehicle more expensive. Researchers should investigate different approaches for reducing high computational cost and memory consumption of the DL structure applied in controlling the steering of AVs.

**c. YET TO ATTAIN OPTIMUM STEERING PERFORMANCE**

The optimum performance of steering control via DL has not yet been achieved, and there is still room for improvement to attain the optimum performance. Better performance of steering control can be attained through improving the neural network architecture, specifically through batch normalization and skip connections. In addition, it can be extended to take care of obstacle avoidance through investigating mediated perception techniques using conditional networks or through adding a temporal aspect to the steering angle prediction

**d. NIGHT DRIVING**

The AVs are expected to drive during day and night time when deployed as commercial vehicles on public roads. If AVs only rely on daytime driving, their scope of activities remains. We therefore suggest that researchers include both driving scenarios in future studies.

# 8. CONCLUSION

* This project started with training the model and tweaking parameters to get the best performance on the tracks and then trying to generalize the same performance on different tracks.
* There was a need to use image augmentation and processing to achieve real time generalization when models performed best on 1 track and poor on 2 track.
* Applying Deep learning algorithm like CNN, the accuracy of steering angle prediction of autonomous vehicles is improved.
* Deep learning algorithms work very well in new and unknown scenarios when the dataset is very large i.e., have been trained on many hours of driving in different scenarios.

# REFERENCES

[1] Gidado, Usman Manzo, Haruna Chiroma, Nahla Aljojo, Saidu Abubakar, Segun I. Popoola, and Mohammed Ali Al-Gardai. "A Survey on Deep Learning for Steering Angle Prediction in Autonomous Vehicles." *IEEE Access* 8 (2020): 163797-163817

[2] Oliver Cameron, “Challenge #2: Using Deep Learning to Predict Steering Angles”, Published on 11 Aug 2016, https://medium.com/udacity/challenge-2-using-deeplearning-to-predict-steering-angles-f42004a36ff3, accessed Jul 2017

[3] Ujjwal Karan, “An intuitive Explanation to Convolutional Neural networks”, Published on 11 Aug 2016, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets, accessed Nov 2017

[4] Mariusz Bojarski, “End-to-End Deep Learning for Self-Driving Cars”,Published on 17 Aug, 2016, <https://devblogs.nvidia.com/deep-learning-self-driving-cars/>, accessed Nov 2017

[5] Dmytro Nasyrov, “Behavioral Cloning. NVidia Neural Network in Action.”, Published on 21 Aug 2017, https://towardsdatascience.com/behavioral-cloning-project-3-6b7163d2e8f9, accessed Jan 2017