Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

(An Autonomous Institute, Affiliated to Visvesvaraya Technological University, Belagavi, Accredited by NAAC, with 'A' Grade)

Near Jnana Bharathi Campus, Bengaluru – 560056

Aided By Govt. of Karnataka

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

A Mini-Project Report on

"Self-Driving Car"

Submitted in partial fulfilment of the requirement for the award of the Degree of

Bachelor of Engineering
In
Computer Science & Engineering

Submitted by

KUMAR VEDANT 1DA19CS078

For the academic year 2021-22

Under the Guidance of

Mr. Srinivasa A. H.

Associate Professor, Dept. of CSE,
Dr.AIT, Bengaluru – 56.

Visvesvaraya Technological University

Jnana Sangama, Belagavi, Karnataka 590018.

Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

(An Autonomous Institute, Affiliated to Visvesvaraya Technological University, Belagavi, Accredited by NAAC, with 'A' Grade)

Near Jnana Bharathi Campus, Bengaluru – 560056

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

Certified that the Mini-Project work titled **"Self-Driving Car"** carried out by **Kumar Vedant** bearing the USN **1DA19CS078,** is bonafide student of **Dr. Ambedkar Institute of Technology, Bengaluru,** in partial fulfilment for the award of **Degree in Bachelor of Engineering in Computer Science & Engineering** of Dr. Ambedkar Institute of Technology during the academic year 2021-22. It is certified that all corrections/suggestions indicated during Internal Assessment have been incorporated in the Mini-Project report deposited in the department. The Mini-Project report has been approved as it satisfies the academic requirements in respect of Mini-Project work prescribed for the said Degree.

----------	----------	----------
Signature of the Guide	**Signature of the HOD**	**Signature of the Principal**
Mr. Srinivasa A. H.	**Dr. Siddaraju**	**Dr. M. Meenakshi**
Associate Professor,	Professor and HEAD,	Principal,
Department of CSE,	Department of CSE,	Dr.AIT, Bengaluru – 56.
Dr.AIT, Bengaluru -56.	Dr.AIT, Bengaluru – 56.	

Viva-Voce Examination

Name of the Examiners	Signature with Date
1.	
2.	

ACKNOWLEDGEMENT

The sense of jubilation that accompanies the successful completion of this Mini-Project would be incomplete without mentioning and thanking all the people who played a vital role in the completion of this project by providing endless encouragement and support.

I would like to profoundly thank the **Management-PVPWT** of Dr. Ambedkar Institute of Technology, for providing such a healthy environment to learn and implement new technologies.

I would like to thank **Dr. M. Meenakshi,** Principal, Dr.AIT, who has always been a great source of inspiration while carrying out this Mini-Project.

I am extremely grateful to **Dr. Siddaraju**, Professor and HEAD, Department of CSE, Dr.AIT for providing me constant encouragement and permitting me to utilize the required laboratory facilities and a congenial working environment for the successful completion of this Mini-Project.

I am highly indebted to my guide **Mr. Srinivasa A. H.**, Associate Professor, Department of CSE, Dr.AIT for constant guidance and support, as well as for providing necessary information regarding the Mini-Project.

I would also like to thank all the teaching and non-teaching staff members of Department of Computer Science & Engineering Department for their support during the course of this Mini-Project implementation.

Lastly, I would like to thank my parents and friends whose constant encouragement and support was crucial in execution and completion of this Mini-Project.

Kumar Vedant

ABSTRACT

The biggest challenge of a self-driving car is autonomous lateral motion so the main aim of this project is to clone drives for better performance of the autonomous car for which I am using multilayer neural networks and deep learning techniques. I will focus to achieve autonomous cars driving in stimulator conditions. Within the simulator, preprocessing the image obtained from the camera placed in the car imitate the driver's vision and then the reaction, which is the steering angle of the car. The neural network trains the deep learning technique on the basis of photos taken from a camera in manual mode which provides a condition for running the car in autonomous mode, utilizing the trained multilayered neural network. The driver imitation algorithm fabricated and characterized in the project is all about the profound learning technique that is centered on the NVIDIA CNN model.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Going by means of vehicle is at present one of the most perilous kinds of transportation with over a million passing every year around the world. As nearly all car crashes (particularly fatal ones) are caused by driver error, driverless vehicles would viably dispose of about all dangers related to driving just as driver fatalities, road safety, and mobility for everyone, and injuries. The self-driving vehicle is the dormant beast that can make a huge difference. The self-driving vehicle is a vehicle outfitted with an autopilot framework and is equipped for driving without the help of human administrator. This innovation was fabricated initially using autonomy approach yet with the progress in the field of computer vision and AI we can utilize the deep learning approach. There are a couple of troubles that need to have been met before executing self-driving car. One of the most important functionalities of a self-driving vehicle is autonomous lateral control. Autonomous lateral motion traditionally depends on image processing. The process is divided into detecting byways, locating by way centers, track planning, track following, and a control logic. The precision of such frameworks depends essentially on how well-tuned the picture processing filters are. These strategies are amazingly sensitive to assortments in lighting and are slanted to false identification. An algorithm tuned to detect lane markings in a place where it is bright and sunny may not do well in a place where it is dark and gloomy. This framework utilizes a convolutional neural system to yield controlling points dependent on street images basically the model is set up to copy human driving behavior. Since an end-to-end model doesn't take a shot at truly described standards, it is less affected by changes in lighting conditions. At the end of this project, I will try to go in detail of how to control autonomous car, I will develop a model of a car motion and will test my control design in my stimulator.

Chapter 2

LITERATURE SURVEY

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 lane 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 is only

implementable in a controlled environment where I have a path created with white papers or stuff like that, it cannot work on roads because I 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. I need to put in every single road to train model that I need to use in the 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 my 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 I want to conclude the knowledge gained through this literature study I can make a model plan I will use the first method to convert the image to threshold so I can get a clear edge planning so edge detection can be easier by using canny edge detection. One unique thing I 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 I 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. 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. Not every method calculates the angle, some follows the color approach and some follows similar pixels approach. I we will be using angle approach.

Chapter 3

SYSTEM REQUIREMENTS

3.1 Software Requirements

1. Anaconda Distribution
2. Jupyter Notebooks
3. Text Editor(IDE): Visual Studio Code
4. Programming Language: Python 3
5. Python Libraries
    a. Numpy
    b. OpenCV
    c. Keras

3.2 Hardware Requirements

1. Laptop Model: DELL Inspiron 11 3148
2. Operating System: Windows 8.1 or high
3. Installed Ram: 4.00 GB DDR3L 1600 MHz
4. Processor: Intel(R) Core(TM) i3-4030U CPU

Chapter 4

SYSTEM DESIGN

4.1. 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.

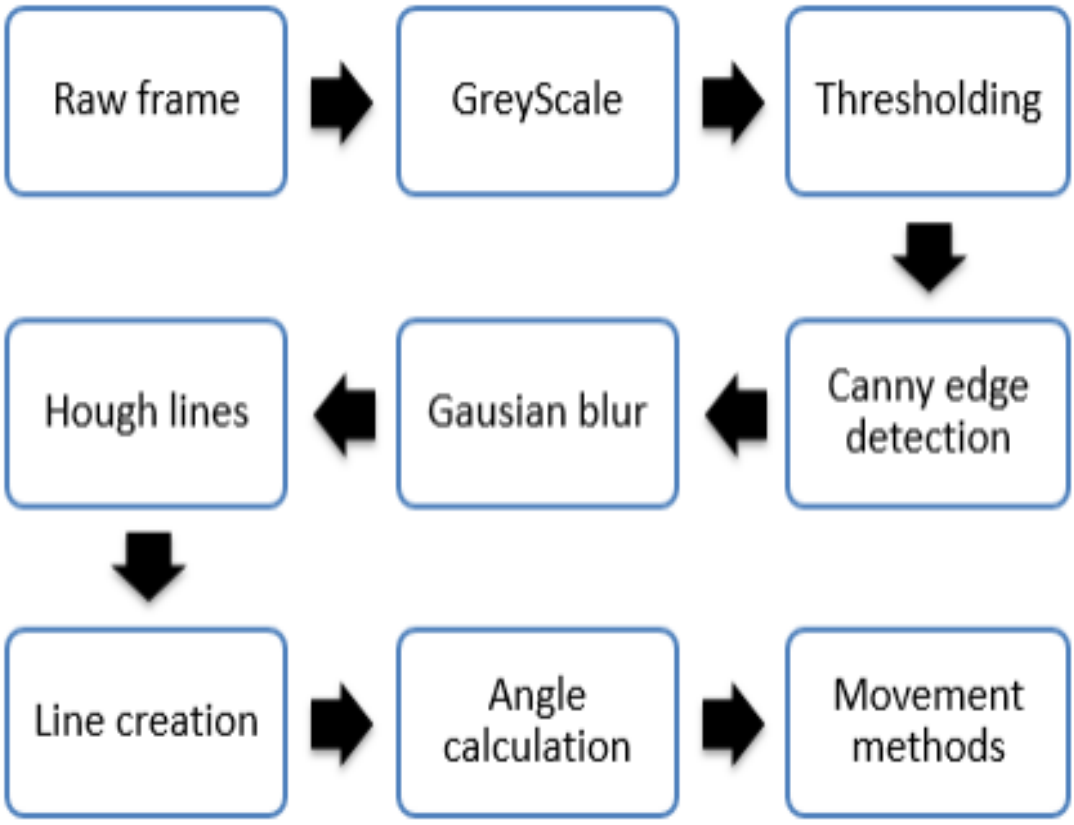

Figure 4.1: General Block Diagram for System Architecture

4.2. Working Principle of CNN

In CNN every image is represented in the form of pixel values and it compares images piece by piece. It is commonly used to examine visual pictures by handling information with a grid-like topology.

CNN has the following layers:

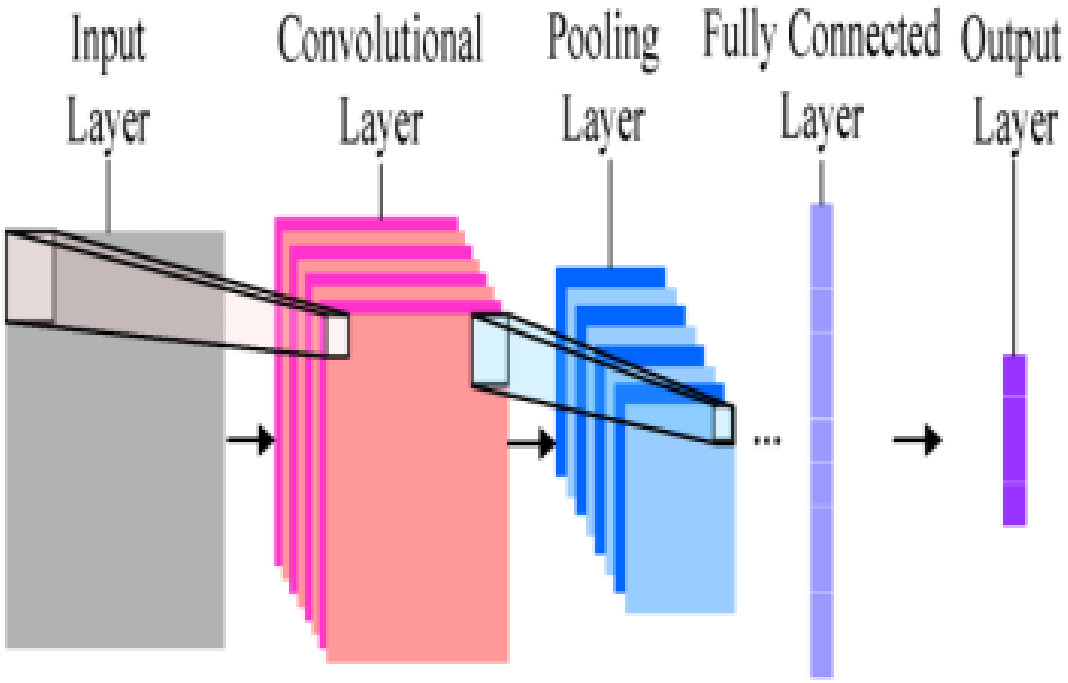

Figure 4.2: Layers of CNN

If any of the layers fail to perform their task then the process will never be executed. In the convolutional layer we first line up the feature along with picture and then multiply the pixel value with the corresponding value of filter and then adding them up and driving them with the absolute value of pixels. Relu layer represents rectified layer unit crafted by this layer is to expel all the negative qualities from the filter picture and afterward change it with zero. The node is activated if the image is above a particular quality and it is linearly dependent on the variable considering the input is above threshold value. Image received from Relu is shrunk in the pooling layer. The actual classification is done in a fully connected layer. We take the shrieked image and up that in a single list. And then we compare that image with our previously-stored list and judge the image. The last is the output layer which gives the output of the classified image.

Chapter 5

IMPLEMENTATION

5.1 Behavioral Cloning

Behavioral cloning is a technique by which sub-psychological aptitudes like - perceiving objects, understanding while simultaneously performing an action can be captured and imitated in a computer program. The skills performed by human entertainers are recorded alongside the circumstance that offered rise to the activity. The learning program yields a lot of rules that reproduce skilled behavior. This method is used to create automated control structures for complex tasks where the classical control view is inadequate.

In simple words, the work of behavioral cloning is to gather the information and train the model to impersonate the behavior with the data that has been collected.

In summary, these include Deep Neural Network(DNN) we are essentially going to download a self-driving car simulator which is open source by Udacity, we will then use this stimulator to create our very own training data for our model for driving a car through the training track inside the stimulator, as we drive the car through the stimulator we are going to take images at each instance of the drive, these images are going to represent our training dataset and the label of each specific image is the angle on which the car will turn at that given point. After that we will show all the pictures on our CNN and let her learn how to drive independently by studying our behavior as a manual driver, this key difference that our model will learn to adjust is the car's steering ability at any time, it will learn to effectively turn the steering angle to the right level based on the stimulation it finds itself in. After preparing the model we test it performance on a unique test track where the car will be driven independently. This behavioral cloning technique is very helpful and plays a big role in a real-life self-driving car as well. After learning this behavioral cloning technique, we will effectively be able to understand and even apply the science of self-driving cars.

5.2 Network Architecture

The algorithm comprises of modules that start from collecting data which contain images, steering angles, and speed. Then came the balancing part followed by processing which is very important for the proper preparation of data entering the neural network.

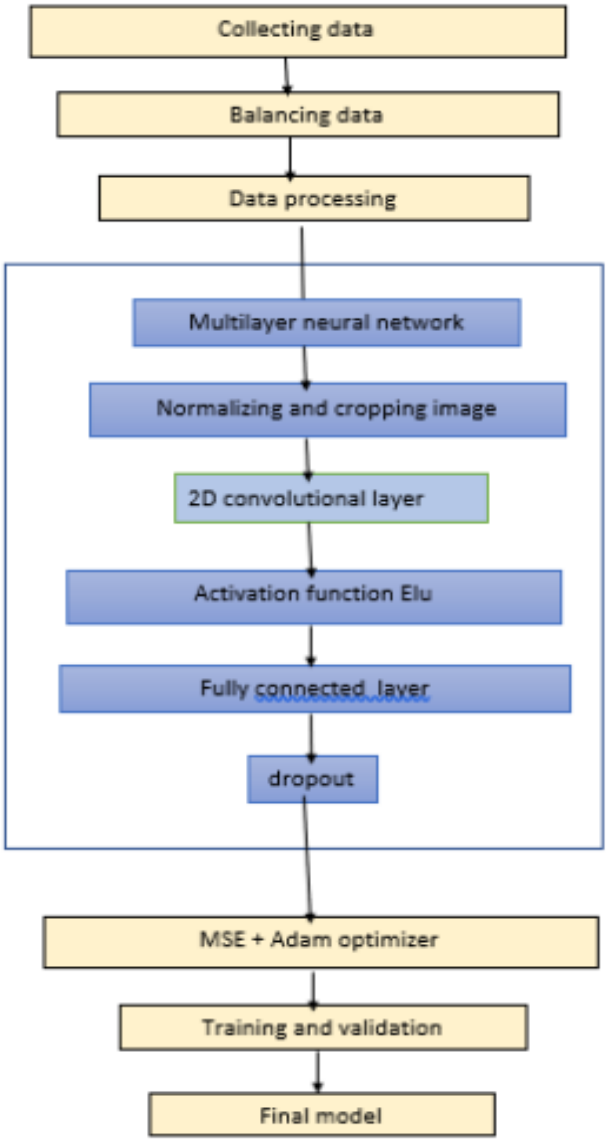

Figure 5.1: Block Diagram of Driver Imitate Algorithm

Three technique are used to preprocess the data:

1. Normalization
2. Argumentation
3. Cropping the image so that unwanted data like the sky and hood of the car will remove.

The multilayer neural network model which is the key part of the driver cloning algorithm. It consists of 5 convolutional layers, one leveling layer, flatten layer, and 3 fully connected layers. The first 5 layers are monitored with the ELU function. Then the MSE and Adam optimizer adjust the hyper parameters. The MSE (mean square error) limits the distinction among the predicted steering angle and the actual steering angle. The last is the training and the validation in which the data is divided into two sets, 80% training, and 20% validation.

5.2.1 Collecting data

We are going to drive our car through the stimulator using our keyboard keys. The more we drive the car precisely the more accuracy will come in our model i.e. cloning the behavior and hence the term called behavioral cloning. We downloaded the simulator provided by Udacity organization which is open-source. There is another open-source stimulator also which is available for this purpose for example air sim which is based on an unreal engine. We have to train the model by driving the car with the help of our keyboard button as we use to do in our video games. This track is organized in a manner to provoke the neural network to conquer sharp turns. The main thing to remember is to attempt to drive the car in the center of the road to ensure a balanced dataset. In simple words, the work of behavioral cloning is to gather the information and train the model to impersonate the behavior with the data that has been collected.

The car built with three cameras is situated on the left, right and one on the center. Images are collected, they collect the value of the steering wheel and speed throttle and brake at the current image as shown in Figure 5.2.

The steering angle is labeled from 1 to -1 where 1 is for the right turn, -1 is for the left turn, and 0 denotes that the car is moving in the straight path. Here the vertical axis specifies the values of our histogram i.e. frequency of each steering angle during stimulation.

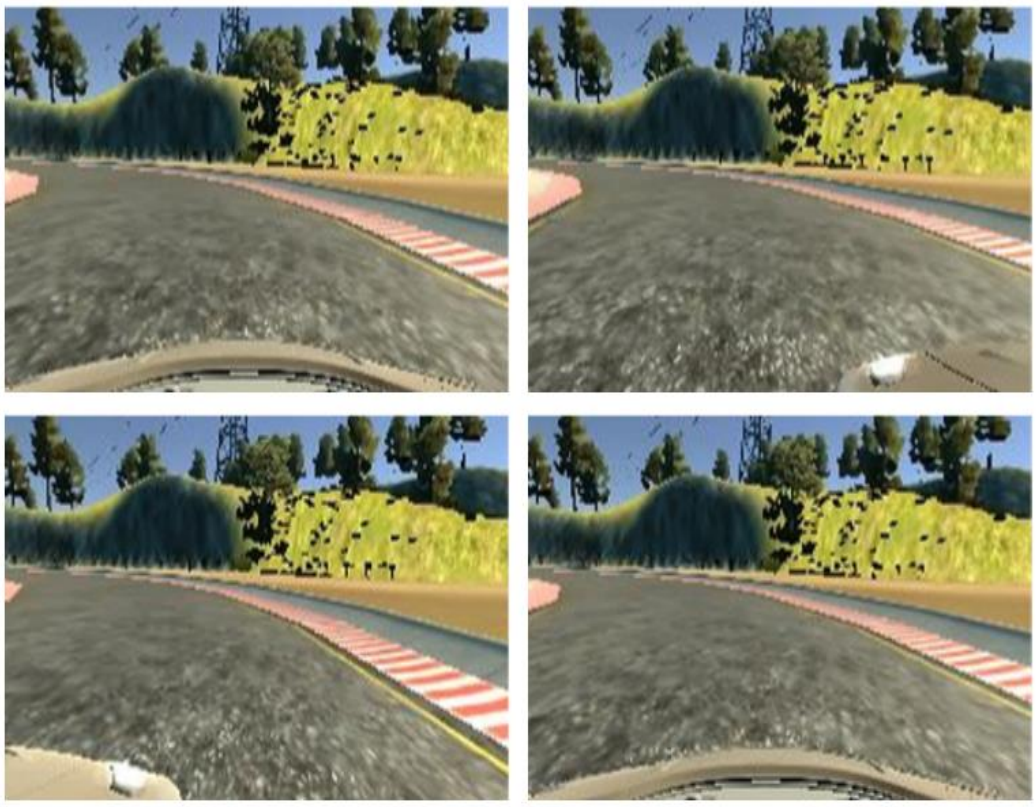

Figure 5.2: Example of Collected Image

```
1  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_39_953.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_39_953.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_39_953.jpg, 0, 0, 0,
   0.6815608
2  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_059.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_40_059.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_40_059.jpg, 0, 0, 0,
   0.6747161
3  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_173.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_40_173.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_40_173.jpg, 0, 0, 0,
   0.6665928
4  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_278.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_40_278.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_40_278.jpg, 0, 0, 0,
   0.6598981
5  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_387.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_40_387.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_40_387.jpg, 0, 0, 0,
   0.6519505
6  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_493.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_40_493.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_40_493.jpg, 0, 0, 0,
   0.6453903
7  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_594.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_40_594.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_40_594.jpg, 0, 0, 0,
   0.6388959
8  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_703.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_40_703.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_40_703.jpg, 0, 0, 0,
   0.6311884
9  C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_810.jpg, C:\Users\USER -
   DELL\Desktop\data\IMG\left_2022_07_10_21_54_40_810.jpg, C:\Users\USER - DELL\Desktop\data\IMG\right_2022_07_10_21_54_40_810.jpg, 0, 0, 0,
   0.6248362
10 C:\Users\USER - DELL\Desktop\data\IMG\center_2022_07_10_21_54_40_916.jpg, C:\Users\USER -
```

Figure 5.3: Example of CSV File

5.2.2 Balancing Data

In the chart, we can see that we have so many 0 steering angles i.e. the recursion of 0 is higher. This will create a problem by making our model biased toward driving down straight. This would cause data imbalance. To overcome this problem, we will improve this by eliminating all the samples above a given threshold so that the data will not be biased in driving straight all the time.

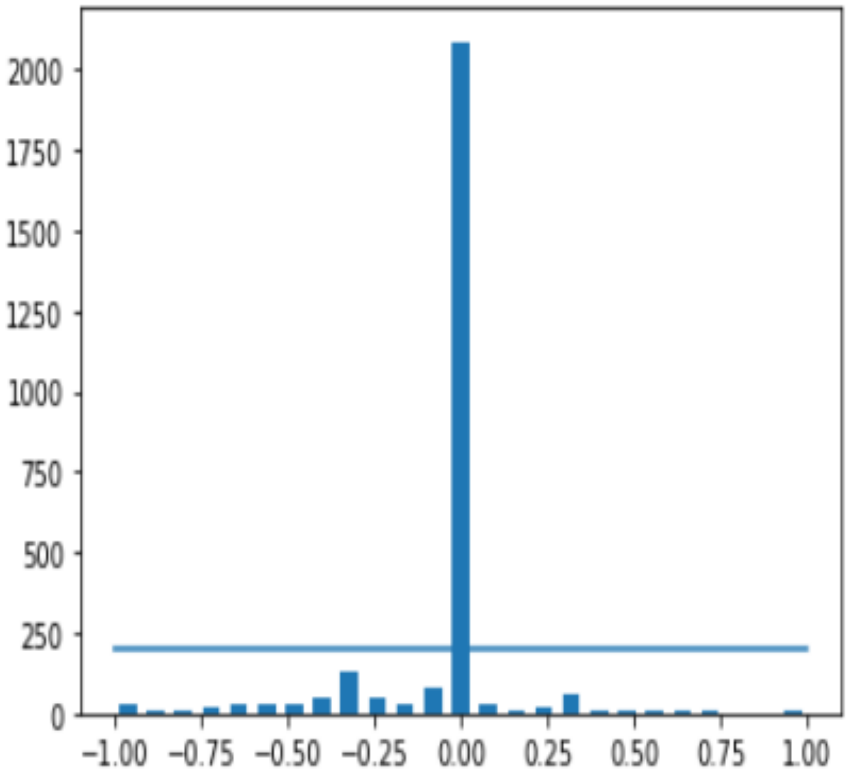

Figure 5.4: Bar Graph of Steering Angle

5.2.3 Reprocessing Data

This is one of the important steps, as we can see that in our image there is a lot of areas which are not of our use or we can say that it isn't imperative to concentrate on, so we wiped out from the image. Features like sky mountain which is there in the picture and the hood on the vehicle in the base. These portions are not at all relevant for our car to determine steering angles. So, we cropped this by using simple NumPy array slicing. As we can clearly see in our image that the image of axis 160x300 in which the height from 160 to 135 is the hood of the car and the range from 60 to 0 is the sky and landscape which is not of any use so we removed that portion from the photo and set its axis from 60 to 135. This will allow to concentrate more on significant features.

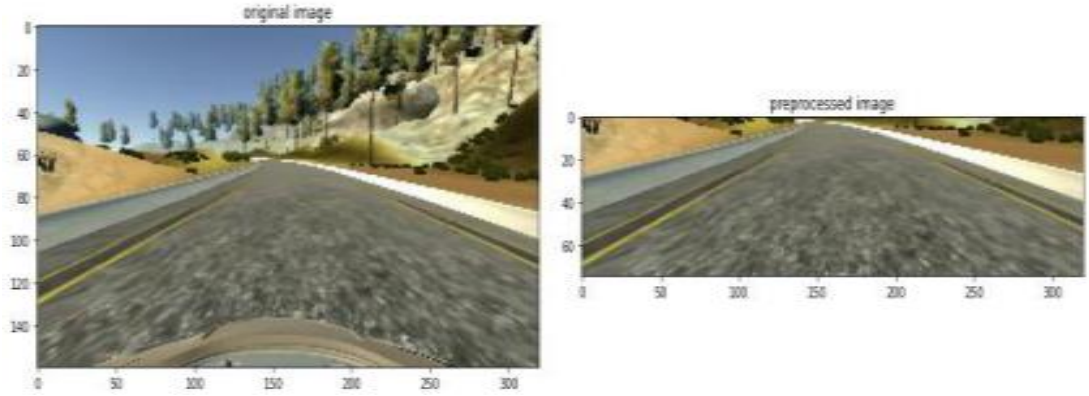

Figure 5.5: Cropped Image

The next step is to change the color space of the image, this is important for preprocessing because the model we use recommends that we use YUV color space where Y stands for luminosity or brightness of the image and UV represent chromium which adds color. We show a processed image by adding gaussian blur technique which helps in smoothing and reducing noise of the image.

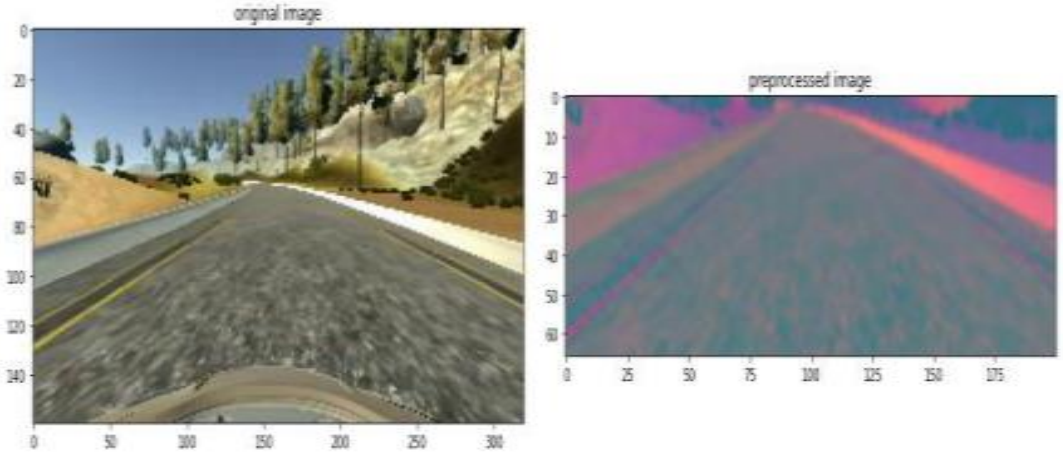

Figure 5.6: Preprocessed Image

5.2.4 Model

In behavioral cloning, our dataset is more complex than any other dataset because now we are dealing with images having dimensions 200x66. The popular model use foe behavioral cloning is the NVIDIA model and it also has to be implemented in real-time self-driving cars.

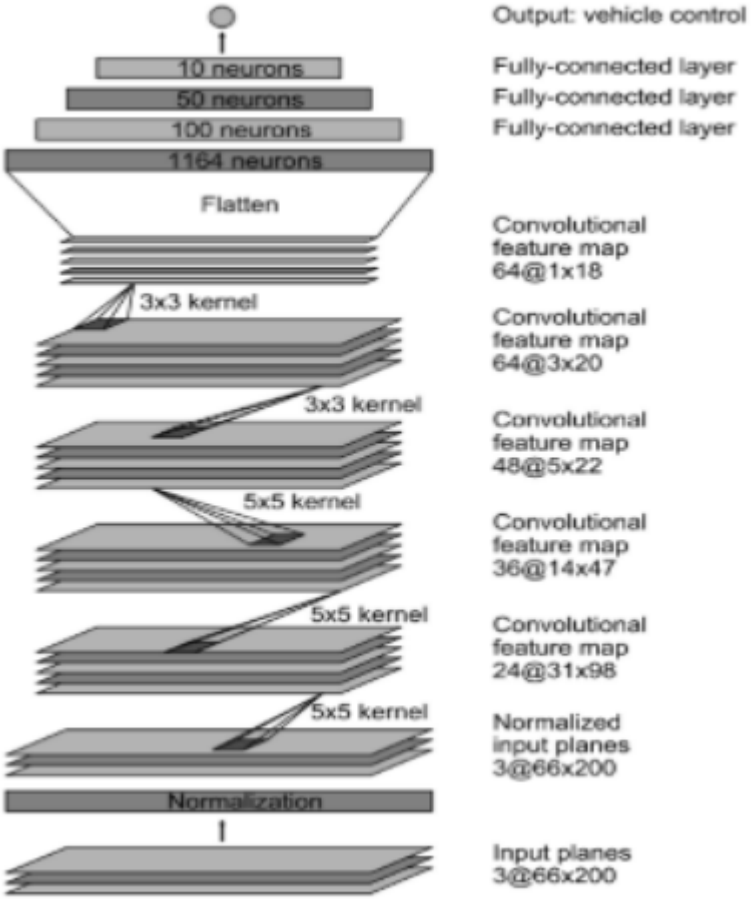

Figure 5.7: Model for Behavioral Cloning

This model will work in the following steps.

Step 1: Normalization layer (hard-coded) divided by 127.5 and subtract by 1.

Step 2: Convolutional layers with 24, 36, and 48 filters and 5x5 kernel and stride of 2.

Step 3: 2 Convolutional layers with 64 filters, 3x3 kernel, and stride 1.

Step 4: A flatten layer.

Step 5: 3 fully connected layer with an output size of 100, 50, and 10.

Step 6: final output layer that gives the steering layer.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 18, 64)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               115300
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
```

Figure 5.8: Model Sequence

Next, we did the training process and validate by using the epochs. Here we perform cycle of 30 epochs and each batch contains 100 elements which helps in decreasing the loss which is shown in Figure 5.9.

```
Train on 662 samples, validate on 166 samples
Epoch 1/10
662/662 [==============================] - 7s 10ms/step - loss: 0.6617 - val_loss: 0.2204
Epoch 2/10
662/662 [==============================] - 0s 394us/step - loss: 0.1945 - val_loss: 0.1587
Epoch 3/10
662/662 [==============================] - 0s 397us/step - loss: 0.1453 - val_loss: 0.1444
Epoch 4/10
662/662 [==============================] - 0s 394us/step - loss: 0.1362 - val_loss: 0.1505
Epoch 5/10
662/662 [==============================] - 0s 406us/step - loss: 0.1309 - val_loss: 0.1412
Epoch 6/10
662/662 [==============================] - 0s 390us/step - loss: 0.1243 - val_loss: 0.1392
Epoch 7/10
662/662 [==============================] - 0s 395us/step - loss: 0.1166 - val_loss: 0.1333
Epoch 8/10
662/662 [==============================] - 0s 397us/step - loss: 0.1154 - val_loss: 0.1269
Epoch 9/10
662/662 [==============================] - 0s 396us/step - loss: 0.1091 - val_loss: 0.1233
Epoch 10/10
662/662 [==============================] - 0s 400us/step - loss: 0.1172 - val_loss: 0.1248
```

Figure 5.9: Epochs with Loss Value

Chapter 6

RESULT

Behavioral cloning results in a simulated driving scenario behavioral cloning we stimulate driving scenarios which are implemented using CNN in which we use activation function which is Elu. Elu function uses mean square of error loss which is used for validating data. As we had expected, performance after the training of the data is better but the two values of trained and tested date are very close which lead us to a conclusion that the model portrays more of general qualities.

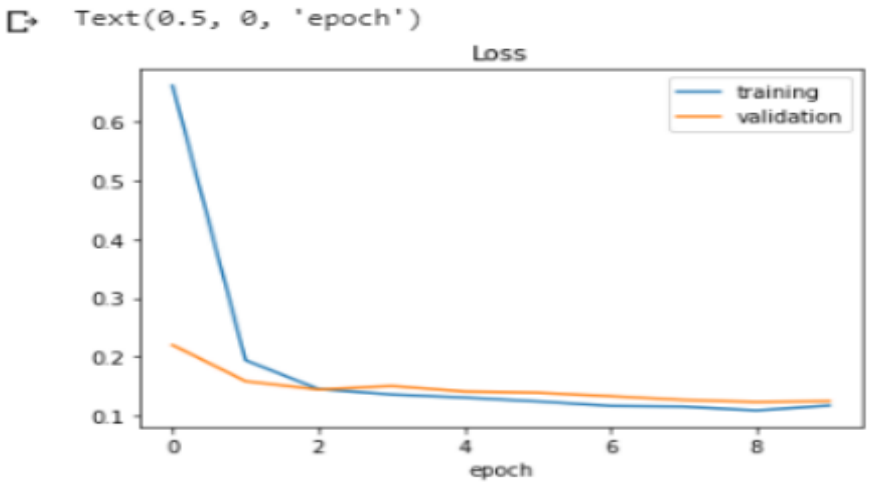

Figure 6.1: Training and Validation

After applying the above method here are the results, the car is running on its track and the result which can observe that in the right-hand side corner there is a speedometer. The trained model can successfully control the car in different, unknown tracks. The model can do laps consistently without failing. With a bigger training dataset comprising of various scenarios, the models' capacity to remain in autonomous mode will increase.

Figure 6.2: Driving in Autonomous Mode

APPLICATION

1. The main aim behind drivers for achieving autonomous driving is the reduction of traffic accidents by eliminating human error, increasing road capacity and traffic flow by reducing distance between cars and making use of traffic management information, relieving the car occupants from driving and navigation activities and allowing them to engage in other activities or rest.

2. A driverless car requires the combination of several techniques among which GNSS. These techniques will enable to guide autonomously a land vehicle from one point to another using public roads. In autonomous driving, GNSS can be used for navigation by determining the vehicle location and speed. With this information the vehicle route can be decided using digital maps. Lane and attitude determination could also benefit from GNSS if the accuracy is good enough. If the location information is shared among cars, GNSS could be a part of a short-range situation awareness system (awareness of other vehicles in the road and collision avoidance) although it is not expected that GNSS is the sole means of information for short-range situation awareness.

3. Examples of autonomous vehicle projects are:
    - **Google driverless car** - Project by Google that involves developing technology for driverless car.
    - **EUREKA Prometheus Project** - The programme for a European Traffic of Highest Efficiency and Unprecedented Safety was the largest R&D project ever in the field of driverless cars and was funded by the European Commission.
    - **BRAIVE** - Prototype autonomous vehicle developed by VisLab of the University of Parma.

CONCLUSION

This project gives one methodology under the stimulated condition for self-governing driving the methodologies use deep learning strategies and end to end figuring out how to accomplish copying of vehicles. The Nvidia neural network is the main frame of the driver cloning algorithm. It is consisting of five convolutional layers one leveling layer and four fully connected layers. The output we receive is the steering angle. The result its use of autonomous mode is successful autonomous driving along a predefined stimulated path through which the model was trained using smaller data sets. It ought to be stressed that all the data expected to train the system are independently created in manual mode, thus creating their own databases. We can improve our method by using a better stimulus generalization. Deficient generalizability happens as an outcome of little database in which controls its application to a natural situation. As though now the car in autonomous mode is running really good along a predefined stimulator route.

REFERENCES

[1] Nicolas Gallardo, "Autonomous Decision Making for a Driver-less Car", 2017.

[2] S. Liu et al., Creating Autonomous Vehicle Systems, Morgan Claypool Publishers, 2019.

[3] D. Bemstcin, and A. Kamhauser, (2012) "An Introduction to Map Matching far Personal Navigation Assistants", Princeton University, Princeton, New Jersey, August 2012.

[4] Joshi and M. R. James, "Generation of accurate lane-level maps from coarse prior maps and lidar," 2014.

[5] Qudsia Memon, Shahzeb Ali, Wajiha Shah, "Self-Driving and DriverRelaxing Vehicle", 2016.

[6] Naveen S Yeshodara, Nikhitha Kishore, "Cloud Based Self Driving Cars", 2014.

[7] https://medium.com/swlh/behavioural-cloning-end-to-end-learning-for-self-driving-cars-50b959708e59