Behaviour Cloning for Autonomous Driving

T. Kirthiga Devi
Department of Information Technology, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu. E-mail: kirthigt@srmist.edu.in

Akshat Srivatsava
Department of Information Technology, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu. E-mail: akshatsrivatava_anil@srmuniv.edu.in

Kritesh Kumar Mudgal
Department of Information Technology, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu. E-mail: km8000@srmist.edu.in

Ranjnish Raj Jayanti
Department of Information Technology, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu. E-mail: rajnishraj_rajeshwar@srmist.edu.in

T. Karthick*
Department of Information Technology, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu. E-mail: karthict@srmist.edu.in

Received July 22, 2020; Accepted September 28, 2020
ISSN: 1735-188X
DOI: 10.14704/WEB/V17I2/WEB17061

Abstract

The objective of this project is to automate the process of driving a car. The result of this project will surely reduce the number of hazards happening everyday. Our world is in progress and self driving car is on its way to reach consumer's door-step but the big question still lies that will people accept such a car which is fully automated and driverless. The idea is to create an autonomous Vehicle that uses only some sensors (collision detectors, temperature detectors etc.) and camera module to travel between destinations with minimal/no human intervention. The car will be using a trained Convolutional Neural Network (CNN) which would control the parameters that are required for smoothly driving a car. They are directly connected to the main steering mechanism and the output of the deep learning model will control the steering angle of the vehicle. Many algorithms like Lane Detection, Object Detection are used in tandem to provide the necessary functionalities in the car.

Keywords

Raspberry Pi, Lane Detection, Obstacle Detection, OpenCV, Deep Learning, Convolutional Neural Networks (CNN).

Introduction

Behavioral cloning is one of the methods in human sub-cognitive skills that helps in capturing and reproducing the computer program. This technique helps the human subject to perform the dexterity, by which his or her actions are being recorded along with the situation that gave rise to the action. The behaviour pattern has been recorded as log and these records are used as input to train an program. The training data outputs facilitate the set of rules to reproduce the skilled behavior pattern[1]. The modus operandi to construct the appropriate automatic control model systems for complex tasks or rules for which control theory are inadequate. Self- driving cars are constantly perpetually in the headlines. These vehicles, designed to hold passengers from point vector A to B without a presence of human maneuver, promise to bring greater mobility in constructing, to reduce the road congestion and fuel consumptions, and to create safe roads. Driverless vehicles[2] are still an emerging novelty domain due to the dynamically changing environment. Advanced driver-assistance systems (ADAS) are available in the market these days. They include features such as adaptive throttle control, emergency braking system, lane detection, and more. The goal of ADAS is to make driving safe by reducing the human errors[3]. However, even the most advanced ADAS system requires the driver to pay full attention while driving, he/she should intervene whenever necessary. The main edge that we gain via these automated vehicles is safety. According to the recent survey by NHTSA, it has estimated that majority of the life threatening crashes are due to human fault. These risk factors can be severely reduced if the human intervention while driving is reduced though the cars are still open to physical damage or circuit failure. Technology does have its down side but we need to look at how it would benefit us. These automated cars would be a major help for physically disabled and old people.

State of the Art (Literature Survey)

In this paper we have used an end-to end trainable model instead of the standard pipeline model. Due to this we required a large amount of data. In this type of modelling architecture the output was predicted in one shot which made it faster than the standard pipeline method. The approach follows a regression based mapping and is able to predict the accurate steering angle required to maintain the vehicle in lane. Results in this paper show a huge advantage that a RCNN network has over a simple CNN network.[4]

The research paper proposes various modelling techniques along with hyperparameter tuning methods for facilitating the process of driving a car. According to the paper, defining all possible situations is not possible so in theory, we can leverage data from

large fleets of human driven cars and use them to make predictions in our autonomous vehicle[5]. The issues due to overfitting and dataset biasing are properly handled by this model. In this paper we used NVIDIA model architecture as the base for our model. It delivered accurate results in most conditions. The system automatically learns how to drive a car by adjusting throttle and steering angle according to the given situation. This NVIDIA architecture processes the input in YUV format and uses the NVIDIA drive PX system as a computational power source[6]. It had three camera modules for capturing images from three different angles (Center, Left and right). The data is collected from a single camera module which is placed in the car so that it is able to capture the upcoming road (the front part) and then fed to CNN. The CNN architecture used is a deep 17 layer model [7]. We have used the ADAM optimizer as it can adjust the learning rate automatically and the activation function used is elu. The paper is a complete walk through, and all the steps have been documented. At the end we were successfully able to clone the sub-cognitive human behaviour[8].

The technique used in this paper is known as Behaviour Cloning from observation(BCO), it's a process that consists of two steps. The first step is to learn the important parameters by observing the human performing the specified task. The next step is to create a model out of it that predicts the outcome of an unknown situation. This paper also covers the impression that these automated car might have on the modern world and in the automobile market[9]. The pros and cons of using an autonomous vehicle over a manual one. It talks about the likeliness of people switching to a self driving car. The analysis indicates that the process of switching might begin in the 2030s. Some of the benefits it talked about were less traffic, more safety on roads, less number of life threatening accidents on road. The cons it mentioned were possibility of system failure, poor performance in some weather conditions and at night time.

In this paper, the main focus is given to DeepTest, a tool whose job is to detect inaccurate behaviour in autonomous cars. It mainly focuses on enhancing the models accuracy by predicting the mistakes that the DL network can make and what outcomes might come out of it[10].

Hardware / Software Requirements

1) Raspberry Pi

The Raspberry pi is designed as a single size -board computer. The model we are using is known as “ModelB+”. It consists of a wifi module and a slot to connect the external

camera module. The power is provided through the custom build PCB that we have made. Here we are using raspberry pi as our primary device mainly used for image processing.

2) Pi Camera

It is an external 8MP camera used for providing high-definition images of the road to the raspberry pi. In this project we will use this sensor for providing input to our Deep Learning Model.

Fig. 1 Features offered in Raspberry pi Model B+

3) Raspbian OS

Of all the linux OS, Raspbian is considered as a variant of Raspberry Pi. Raspbian is designed to be user-friendly with its operating system.. It consists of all the default softwares required. It is free and based on the DEBIAN, and it can be easily downloaded from the official Raspberry pi website.

4) Arduino UNO

This microcontroller is also used in our self driving car. It acts as a slave device in our car and it acts according to the input provided by the raspberry pi. It regulates the motor and adjusts its speed as per the requirements.

Fig. 2 Arduino Uno

5) L298N H bridge Motor

Motor It allows the simultaneous speed, direction execution of control wih 2 DC motors. It operates on voltage between 05-35 V with a peak voltage current of 2A. The 2 PWM pins on it are used to control the speed of motors.

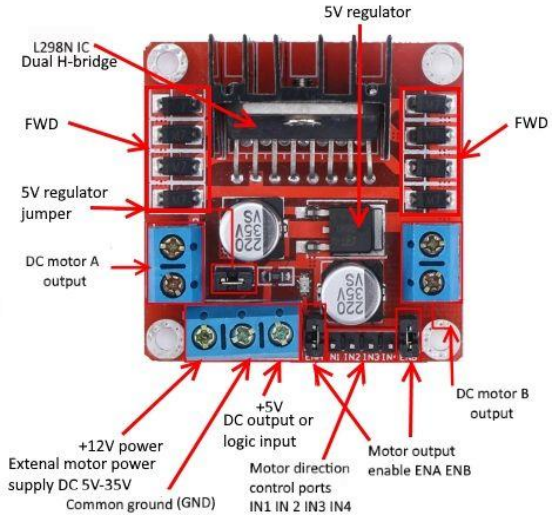

Fig. 3 A L298H dual H bridge Motor

6) Hardware Components Connection

The 4 wheels of the chassis are connected separately to 4 DC motors. Then another serial connection is made between the adjacent wheels so that they move in the same direction. These motors are connected to the circuit with L298N Dual H bridge motor driver motor to help in regulating the speed through the arduino as required. Now connect raspberry pi to the arduino UNO after setting up the raspberry pi cam on it. We also build a custom made PCB in order to provide circuits for providing all the peripheral components.

Implementation

1. Preprocessing the Dataset

The entire dataset for all the modules were preprocessed using the opencv library. The images in the datasets were converted to grayscale (to reduce the computational power required to process them). The images were also down scaled while retaining their important features. Then the datasets were separated into three groups (training, validation and test datasets). After all this the datasets were ready to be fed into the neural network.

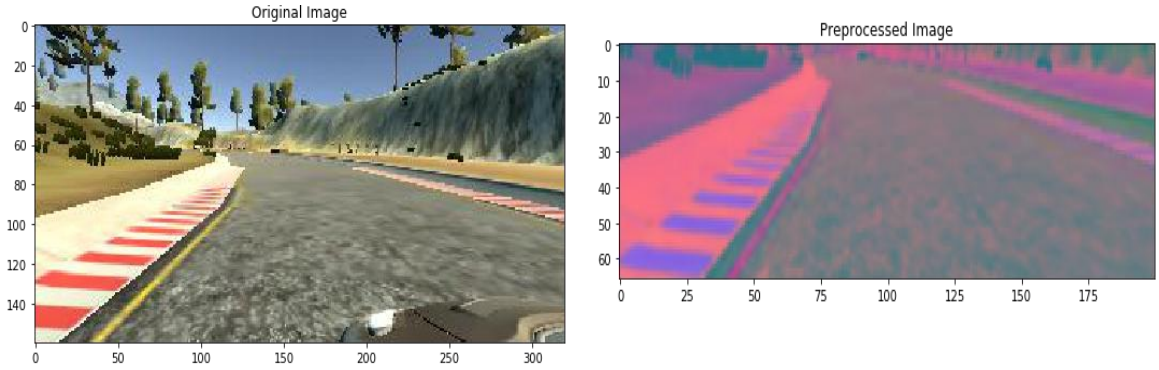

Fig. 4 Comparison between processed and original frame

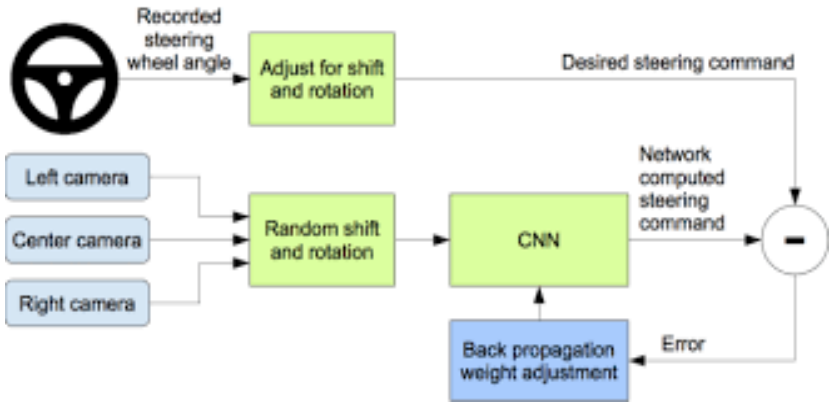

Fig. 5 Overview of the system

2. Lane Detection Model

Algorithm

1. Gather the dataset by driving the car in Udacity's self driving car simulation and record the video in frames along with the respective steering angles.
2. Load the dataset.
3. Preprocess the dataset before feeding it into the neural network.
4. Build the neural network using Keras.
5. Train the model and record the accuracy and loss.
6. Make changes if necessary.
7. Save the model for future use.

Model Summary

a) We used the sequential model in this case.

b) 5 convolutional 2D layers each of kernel size 5 X 5 are added.The activation layer used in every 'elu'.

c) 1 Dropout layer of shape (1,18,64).

d) A max pooling layer with 2 X 2 window
e) A convolutional layer with 64 kernels each of size 3 and padding to maintain size.
f) A max pooling layer with 2 X 2 window

To train the model data augmentation is required. Learning rate is initialized at 0.01 and Adam optimizer is used model was trained for a total of 30 epochs.

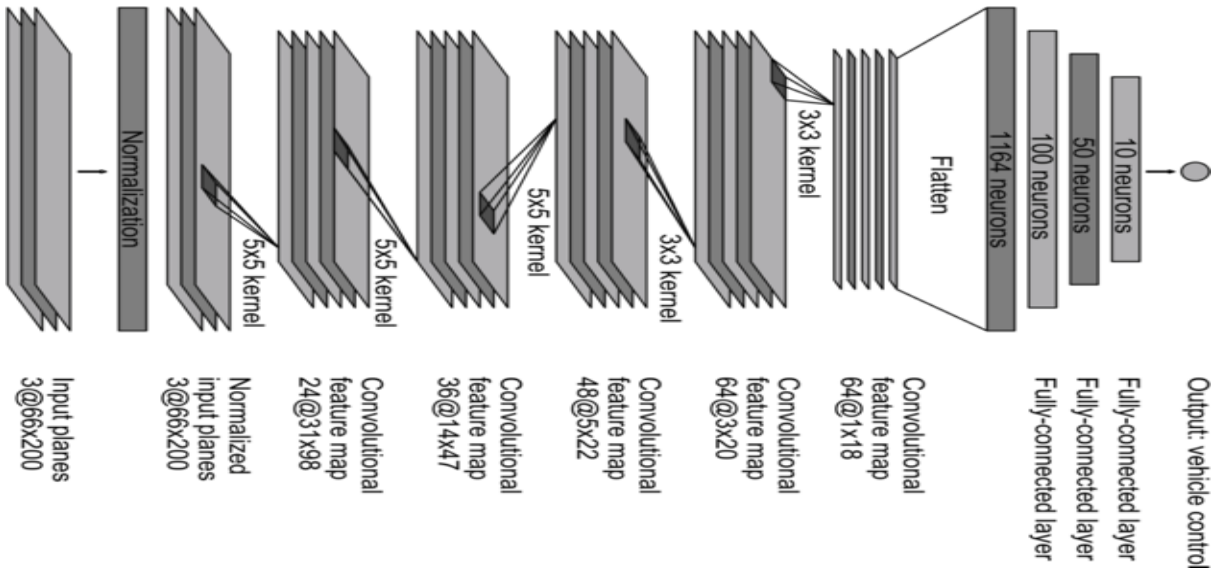

Fig. 6 Shows the Model Summary

After this the whole dataset is preprocessed before being provided into the neural network developed using keras. The images with steering angles zero are reduced because they were significantly more in quantity than the ones with positive(right) or negative(left) steering angle as it might have led to the model being biased in driving straight. After all the preprocessing the dataset is fed into the neural network. After the model was trained we used a simulation to test our model.

Results before changes:- When the model was validated using the validation dataset the accuracy was high but when the model was tested on a different track it was unable to navigate. This meant the model was unable to generalize new data.
Possible Reasons:- Due to small amount of the dataset gathered as the car was driven only three times around the track.
Changes made:- Image augmentation technique used to vary the size of the dataset drastically.
Also changed the learning rate from 1e -4 to 1e -3.

Results after the changes:-The car was successfully able to navigate the new track.

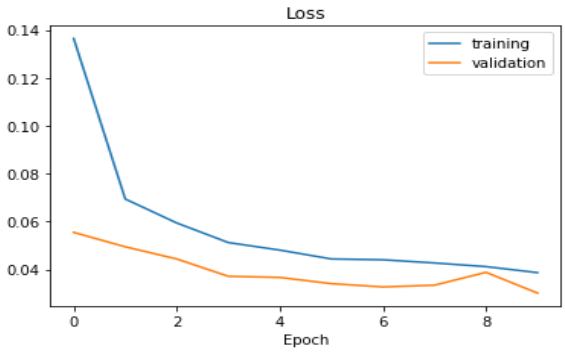

Fig. 7 Shows the training and validation loss of the lane detection model

3. Traffic Sign Detection Model

We have used the Lenet model architecture for this module and made some changes to it for achieving better accuracy. The dataset for this model was gathered from bitbucket website.

Algorithm

1. Check the dataset for data integrity before loading it.
2. After that split the dataset into three groups (training, validation, test datasets).
3. Preprocess the entire dataset before feeding it to the neural network.
4. Build the neural network using keras.
5. Train the model and record the accuracy and loss.
6. Make changes if necessary.
7. Save the model for future use.

Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 30)        780
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 30)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 15)        4065
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 15)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 540)               0
_________________________________________________________________
dense_3 (Dense)              (None, 500)               270500
_________________________________________________________________
dropout_2 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_4 (Dense)              (None, 43)                21543
=================================================================
Total params: 296,888
Trainable params: 296,888
Non-trainable params: 0
```

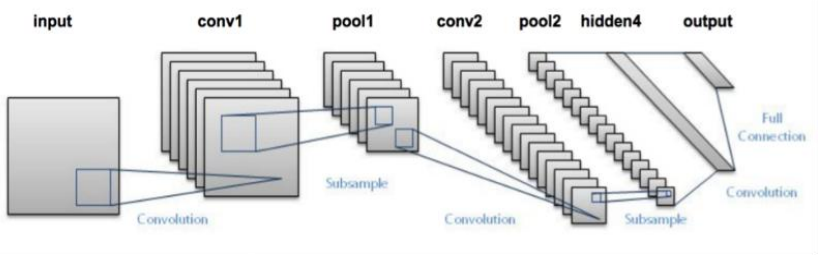

Fig. 8 Model Summary

After the preprocessing the dataset is fed into the neural network.

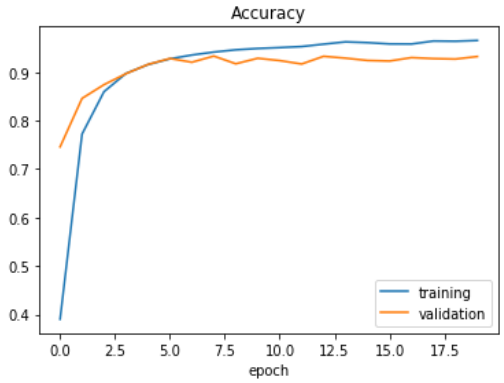

Fig. 9 Shows the training and validation accuracy of traffic sign classification model before changes

Results:- The validation accuracy achieved was 93% but the training accuracy 96% which resulted in over fitting.

Possible Reasons:- Due to small and inconsistent dataset.

Changes Made:- Added an extra convalutional layer after each existing convalutional layer. Also added a dropout layer after the first pooling layer.

Again used image augmentation to increase the size of dataset.

Results after changes:- The validation accuracy was raised to 98%.

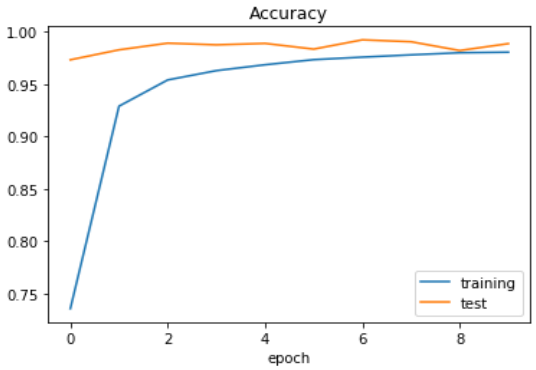

Fig. 10 Shows the training and validation accuracy of traffic sign classification model after changes

4. Object Detection Model

We have used the cascade training gui to generate the for our object detection module. The dataset was gathered from kaggle. Then the dataset was separated into two groups (positive-The ones with object in them along with the background scenery, negative - The ones with only the background scenery).Now after the model is generated detection of objects was successful with a 92% we had now use that information and to let the car know whether it had to stop or turn according to the distance between the car and the object. And for that we took the help of linear equations. We had to solve two of them simultaneously. The concept behind this is when the objects that are being detected in the image are being enclosed by red rectangle. As the object's distance changes, so does the size of the rectangle formed around the object.

The linear equations are:

$$D1=(X2-X1)m+c$$
$$D2=(X4-X3)m+c$$

Here in the above equations (X2-X1) is the length of the rectangle formed when the object detected is near to the car in the image and (X4-X3) is the length of the rectangle formed when the object detected if object detected is farther from the car. The distance D1 and D2 are initially measured using a measuring tape. Using the measured distance and the values of(X4-X3) and (X2-X1) the equations are solved simultaneously and the values of m and c are found. Later on the distance of the object detected can be calculated using the values of m,c and the length of the rectangle.

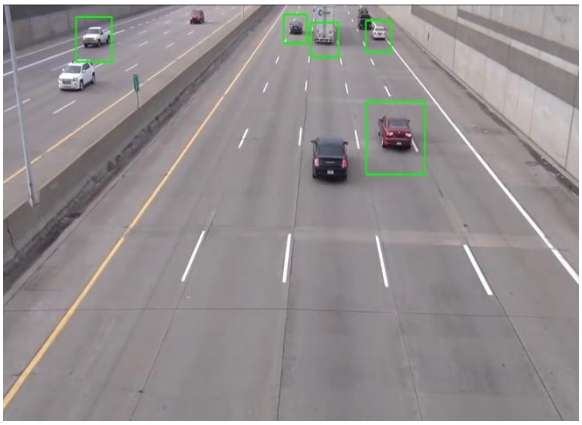

Fig. 11 Shows the output of object detection model

5. Video Capture

This module is done with the help of raspberry pi with camera module. The raspberry pi is powered using a power bank. Since the raspberry pi does not have a display it is connected to the computer. Before doing that ssh file needed to be created in the memory

space of the raspberry pi. After that we could connect using ethernet and use any ssh client to connect to the raspberry pi for graphical user interface. In our case we used putty. Then we configured the wifi in our raspberry pi so that next time we could connect our computer and our raspberry pi using wifi.

After the connection part was done we uploaded the enabled our camera module and uploaded the code to allow us to use the camera and capture frames.

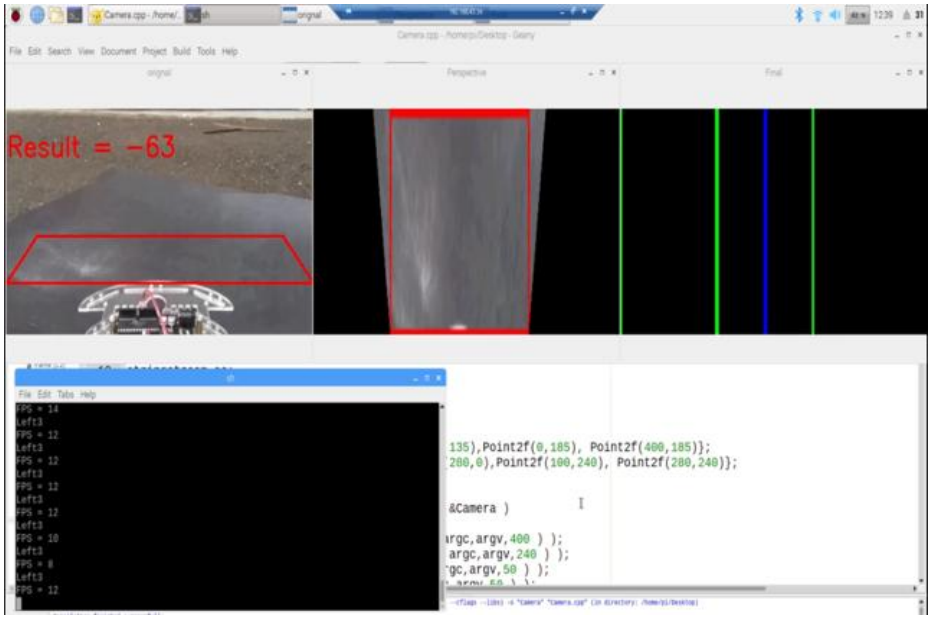

Fig. 12 Shows the output of Video Capture Module

Conclusion

In this paper, a method to make a self driving car is discussed. We discussed all the hardware assembly and even the deep learning part of the project. A single camera module just with the help of some image processing was able to drive a car flawlessly without any human intervention. The camera even detects any obstacles in front of the car and stops or slows accordingly. In order to make the car more versatile we require more sensors (LIDAR, RADAR etc).

References

Saksena, S.K., Navaneethkrishnan, B., Hegde, S., Raja, P., & Vishwanath, R.M. (2019). Towards Behavioural Cloning for Autonomous Driving. *In Third IEEE International Conference on Robotic Computing (IRC)*, 560-567.

Codevilla, F., Santana, E., López, A.M., & Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. *In Proceedings of the IEEE International Conference on Computer Vision*, 9329-9338.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., & Zhang, X. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Farag, W. (2019). Cloning safe driving behavior for self-driving cars using convolutional neural networks. *Recent Patents on Computer Science*, *12*(2), 120-127.

Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral cloning from observation. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, 4950-4957.

Tian, Y., Pei, K., Jana, S., & Ray, B. (2018). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. *In Proceedings of the 40th international conference on software engineering*, 303-314.

Litman, T. (2017). *Autonomous vehicle implementation predictions*. Victoria, Canada: Victoria Transport Policy Institute.

Karthick, T., & Manikandan, M. (2019). Fog assisted IoT based medical cyber system for cardiovascular diseases affected patients. *Concurrency and Computation: Practice and Experience*, *31*(12), e4861. https://doi.org/10.1002/cpe.4861

Kumar, A., & Sharma, A. (2017). Systematic Literature Review on Opinion Mining of Big Data for Government Intelligence. *Webology*, *14*(2), 6-47.

Anandan, M., Manikandan, M., & Karthick, T. (2020). Advanced Indoor and Outdoor Navigation System for Blind People Using Raspberry-Pi. *Journal of Internet Technology*, *21*(1), 183-195.