**DAYANANDA SAGAR COLLEGE OF ENGINEERING**

**(An Autonomous Institute affiliated to VTU, Belagavi - 590018)**

**Accredited by NBA, National Assessment & Accreditation Council (NAAC) with ‘A’ grade**

**A Mini Project Report on**

**“Simulation of self driving car using python”**

***Submitted in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

**IN**

**MECHANICAL ENGINEERING**

***Submitted by***

**Name Of The Candidates: SHASHANK SHEKHAR - USN: 1DS18ME149**

**: RAHUL KUMAR -USN: 1DS18ME058**

**: SATISH KUMAR UPADHYAY-USN: 1DS18ME147**

**: ASFAQUE AHMED -USN: 1DS17ME022**

***Guided by***

**Mr. MANJUNATH PATTAR**

Assistant professor

DEPARTMENT OF MECHANICAL ENGINEERING

DAYANANDA SAGAR COLLEGE OF ENGINEERING

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**JNANASANGAMA, BELAGAVI-590018**

**2020-2021**

**DAYANANDA SAGAR COLLEGE OF ENGINEERING**

**(An Autonomous Institute affiliated to VTU, Belagavi – 590018, Approved by AICTE & ISO 9001:2008 Certified)Accredited by NBA, National Assessment & Accreditation Council (NAAC) with ‘A’ gradeShavige Malleshwara Hills, Kumaraswamy LayoutBengaluru-560078**

**2020-2021**

**DEPARTMENT OF MECHANICAL ENGINEERING**

****

**CERTIFICATE**

Certified that the Mini Project report entitled **“SELF DRIVING CAR SIMULATION BY PYTHON PROGRAMMING”** carried out by **Shashank Shekhar (1DS18ME149), Rahul Kumar (1DS18ME058), Satish Kumar Upadhyay (1DS18ME147), Asfaque Ahmed (1DS17ME022)** bonafide students of **DAYANANDA SAGAR COLLEGE OF ENGINEERING**, an autonomous institution affiliated to **VTU, Belagavi** in partial fulfillment for the award of **Degree of Bachelor of Engineering** in **Mechanical Engineering** during the year **2020-2021**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Mini-Project report has been approved as it satisfies the academic requirements in respect of work prescribed for the Bachelor of Engineering Degree.

**Signature of the Guide Signature of the HOD**

**Mr. Manjunath Pattar Dr. R. Keshavamurthy**

Assistant professor Professor & HOD Dept. of ME Engg. Dept. of ME Engg. DSCE, Bengaluru DSCE, Bengaluru

**ACKNOWLEDGMENT**

Before we start with the report of our mini-project, we would like to express our

gratitude towards the people without whom this mini-project would not have been possible.

We express our deep sense of gratitude and indebtedness to **Mr. Manjunath Pattar**, Assistant Professor, Department of Mechanical Engineering, for his treasured guidance, unremitting assistance, and in the critical assessment of the mini-project.

We express our sincere thanks to **Dr. Keshavmurthy R,** HOD,

Department of Mechanical Engineering, for providing the facilities required for the completion of this project work.

It is with great pleasure, we extend our gratitude and thanks to **Dr. C P S Prakash,**

Principal, Dayananda Sagar College of Engineering, for his encouragement throughout the project.

.

We fell short of words to express our heartfelt thanks to all our family members and

friends and all those who have directly or indirectly helped me during our course.

**Shashank Shekhar (1DS18ME149)**

**Rahul Kumar (1DS18ME058)**

**Satish Kumar Upadhyay (1DS18ME147)**

**Asfaque Ahmed (1DS17ME022)**

**ABSTRACT / SYNOPSIS**

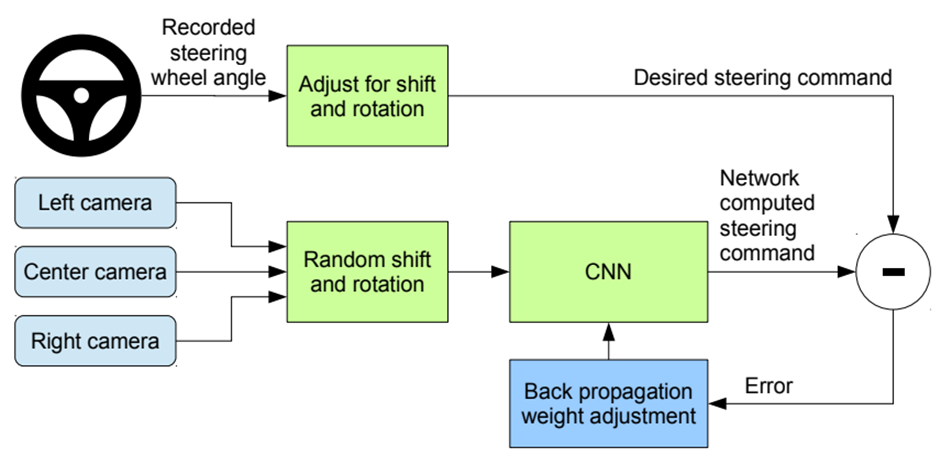
The topic of the project is **“SELF DRIVING CAR SIMULATION BY PYTHON PROGRAMMING”**. As the name suggests it is about the self-driving system which will function automatically according to the circumstances is provided to the car, also this system can act as a way of reduction of human error, ensuring safe driving. The project aims to achieve better decision-making capability to ensure safe driver-less drivers. Every project has some motivation behind its creation and in our case, to contribute something towards the improvement of the rising accidents in our country, this project will help in one way or another to reduce the accidents due to lack of focus during driving.

So coming to the technical aspect, this project’s base component is **JUPYTER NOTEBOOK**.

In today’s world automation is implemented in nearly everything, with this objective this project demonstrates the factors and technology to be used in the automation of vehicles. Self-driving cars are one of the hottest areas of research and business for the tech giants. This project involves training an end-to-end deep learning model that would let a car drive by itself around the track in a driving simulator. This project can be tested on the prototype with a suitable amount of images and data. For the prototype to be more accurate, a huge amount of data will be needed. Changes can be made in the betterment of the project.

**Tools required ( Software ):-**

1. **Python:-** used to write programs for the project.
2. **Udacity simulator:-** used for the data collection and testing of our model.

****

The system works on the CNN(Convolutional neural network), which receives the real-time data from the three point camera and sends it to the steering wheel, if the given command is wrong it will send it back to the CNN and the weights i.e constant values are adjusted to the desired steering command and sent back to the steering wheel.

TABLE OF CONTENTS

**TITLE PAGE………………………………………………………………………….**

**CERTIFICATE………………………………………………………………………..**

**ABSTRACT…………………………………………………………………………...**

**ACKNOWLEDGMENT.................................................................................................**

**LIST OF FIGURES…………………………………………………………………….**

**CHAPTER-1:INTRODUCTION....................................................................................1**

**1.1 LITERATURE SURVEY...............................................................................2**

**1.2 OBJECTIVE OF THE WORK………………………………………….....3**

**CHAPTER-2: DETAILING OF THE PROPOSED WORK........................................6**

**CHAPTER-3**

**3.1 TESTING........................................................................................................13**

**3.2 PROGRAM………………………………………………………………….**

**3.3 RESULTS .......................................................................................................15**

**CHAPTER 4: CONCLUSIONS AND SCOPE FOR FUTURE WORK ..................16**

**4.1 CONCLUSIONS...........................................................................................16**

**4.2 SCOPE FOR FUTURE WORK..................................................................16**

**REFERENCES ..............................................................................................................17**

**LIST OF FIGURES**

1.Fig 2.1 Proposed System Design

2.Fig 3.1- Histogram

3.Fig 3.2 - Augmented Data

4.Fig 3.3- CNN plot

5.Fig 3.4- Trained model

**INTRODUCTION**

In today’s world automation is implemented in nearly everything, with this objective this project demonstrates the factors and technology to be used in vehicle automation of vehicles. Self-driving cars are one of the hottest areas of research and business for the tech giants. This project involves training an end-to-end deep learning model that would let a car drive by itself around the track in a driving simulator. The project is based on concepts of basic deep learning and neural network, we can train our autonomous driving vehicle using the provided platform specifically for this purpose.

Many recent technological advances have helped to pave the way forward for fully autonomous vehicles. This special issue explores three aspects of the self-driving car revolution: a historical perspective with a focus on perception for autonomous vehicles, how government policy will impact self-driving cars technically and commercially, and how cloud-based infrastructure plays a role in the future. Significant improvements in the last decade have greatly advanced self-driving car technology. These new capabilities will have profound global impacts that could markedly change society, not to mention the significant improvements they bring to the overall efficiency, convenience, and safety of our roadways and transportation systems. Addressing self-driving technology-related concerns is important, particularly given these broad potential impacts. Worldwide, 10 trillion automobile miles are driven each year, with complex and novel conditions generating millions of situations in which autonomous vehicles could fail. Yet many challenges remain many challenges remain across all levels of system functionality.

**CHAPTER - 1**

**1.1: LITERATURE SURVEY**

1. **“CREATING AUTONOMOUS VEHICLE SYSTEMS”**

**– Morgan Claypool Publishers**

We are confident that any limitations of the current breed of autonomous vehicles will be successfully resolved to withstand the real-world test of time, through accurate training of models we shall be able to overcome the barriers.

1. **“AUTONOMOUS DRIVING IN URBAN ENVIRONMENTS: BOSS AND THE URBAN CHALLENGE “**

**– Journal of Field Robotics**

A three-layer planning system combines mission, Situation, and motion planning to drive in urban environments, this paper presented an overview of the process of implementation of complete automation in vehicles.

1. **“ROAD-BOUNDARY DETECTION AND TRACKING USING LIDAR SENSING”**

**– IEEE Transactions on Robotics & Automation**

This paper proposes a novel method based on extended filtering for fast detection and tracking of road curbs using successive range/bearing readings obtainedfrom a scanning LiDAR measurement system.

1. **“TRAFFIC FLOW PREDICTION WITH BIG DATA”**

**– IEEE 2015**

In this paper, a novel deep-learning-based traffic flow prediction method is proposed. A stacked autoencoder model is used to learn generic traffic flow features, and it is trained in a greedy layer-wise fashion. To the best of our knowledge, this is the first time that a deep architecture model is applied using autoencoders as building blocks to represent traffic flow features for prediction.

**1.2: OBJECTIVE OF THE WORK**

The objective of this project layer, layer-wise-wise includes, Our project would be used in the development of self-driving cars which could ensure safe driver-less rides, to get the knowledge of deep learning and analyze the data obtained to predict the movement of vehicles, training our model with the help of deep learning based on a predefined parameter such as steering angle, throttle, speed, etc. depending upon the situation, use of the concepts of CNN(convolutional neural network) to train our model, to implement self-learning and decision-making capabilities in the model through simulation and to implement future technologies to reduce human efforts. Though the models performed admirably on the track where they were trained, the major problem was to generalize this behavior to a second simulator circuit. The dataset for Track 1, which was straightforward to drive and had good road conditions, was utilized as the training set for the car to drive autonomously on Track 2, which has sharp curves, barriers, hills, and shadows. Image processing and other augmentation techniques were utilized to solve this challenge, allowing for the extraction of as much information and features from the data as possible. Finally, the car was able to generalize, effectively on Track2. In the future, the initiative aspires to achieve the same precision on real-time data.

**CHAPTER-2**

**DETAILING OF THE PROPOSED WORK**

How to train a self-driving car using Convolution neural networks CNN? We will be using the open-source Self-driving car simulator provided by Udacity that is used in their Self-driving car Nano degree program. Using this simulator we will first drive the car and collect data. Then we will train a CNN model to learn this behavior and then test it back on the simulator. The model we will use was proposed by NVIDIA. They used this model to train real car data and got promising results when they drove it autonomously.

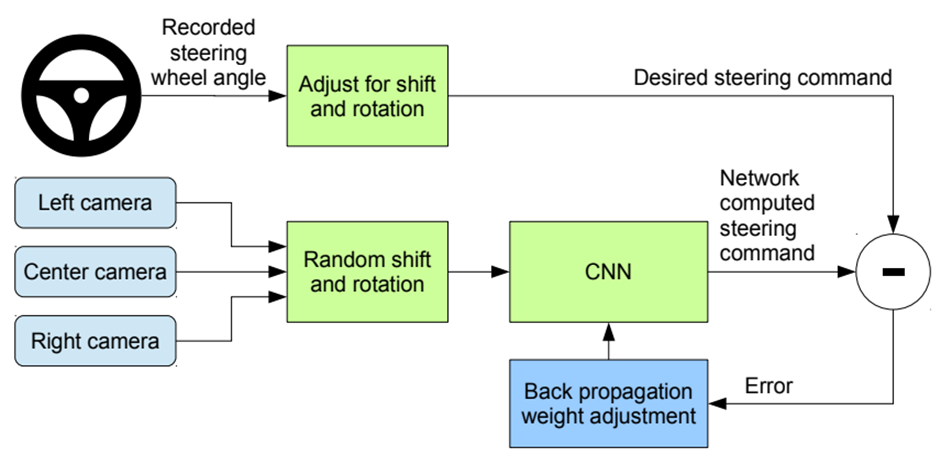
****

Fig 2.1 System Structure

So how do humans learn to drive? Do we calculate how much to turn based on the road lane? Does it have to be specific color road lanes? What if there are no road lanes? Well we will still be able to drive. So the question is how did we learn? For years and years, we watch and observe i.e collecting data of parents driving going on road trips going to school learning the basics of driving. And once we finally get behind them we have a basic understanding of roads, lanes, different signs, etc. So at this point, we already have a model in our head that knows the basics but once we start driving ourselves we train the model even further this time will be steering information along with acceleration and braking. So within a few days of training, we learn how to drive. So over time, we keep getting better as we keep getting more data. So the key here is that we collect data and based on this we create a model that generalizes how to drive. Above we can see the System Structure of NVIDIA’s method, where they collected 3 images from different cameras and recorded the steering angle for these images.

**CHAPTER 3-**

**3.1-TESTING**

**ALGORITHM**

**Initializing data**

1. All the necessary libraries are imported.

2. Read the image.

3. Columns are then assigned to the driving\_log.csv file.

4. The file extension of the center point camera is removed.

5. Print the total number of images.

**Visualize Distribution of Data**

1. The number of bins taken is 31.

2. The sample per bin is 1000.

3. Create a histogram of steering angles and nBins(=31).

4. Create two empty lists binDataList and removedIndexList.

5. Index of steering values is randomly stored in binDataList.

6. Create boundary conditions for the maximum values that can be stored in the per bins(=1000).

7. Bins containing values of more than 1000 are removed. 8. Store the index of the removed data list in removedIndexList.

9. Print the removed images and remaining images.

**Training and Validation images**

1. Create list images path and Steering.

2. Create for loop and read the CSV file to get the index.

3. Print the indexed data.

4. The steering value is then changed to a floating-point number.

5. Convert the list into an array.

6. Divide data randomly into two parts- a. Training data b. Validation data.

7. 80% training data and 20% Validation data.

8. Print the number of images in training and validation data.

**Data Augmentation**

1. Read the image stored in imgPath.

2. Moving the image left or right to create a new set of data. (pan)

3. Zooming in the image

4. Adjusting the brightness of the image, neither too dark nor too bright.

5. Flipping the images to get more right steering angle data

6. Running all the processes in the “if” loop, so that all the images are adjusted and

saved to get more sets of data

**Batch Generator**

1. Preprocessing the images

a. Crop the image from height.

b. Change the color from RGB to YUV.

c. Resizing the image.

2. Create an image batch and steering batch.

3. “For” loop to pick the index of the random image.

4. The index of the image and the index of the steering values are sent back to augment the image.

5. The augmented image is then pre-processed.

6. Pre-processed image is then added to the list we created imgBatch and steering value is added to the steering Batch.

**Training**

1. Create a deep learning model with various convolutional, pooling, and fully connected layers.

2. Train the model for 10 epochs.

3. Plot training loss and validation loss curves.

4. Save the model as a model.h5

**Testing**

1. Load the trained model saved (as a model.h5) earlier.

2. Preprocess image

a. Add gaussian blur filter

b. Resize to 200x66

c. Resolve by dividing each pixel by 255

3. Set max-speed.

4. Send the image for prediction to the model.

5. To view the simulation:

a. Create a socket connection to port 4567 (fixed for audacity self-driving car simulator)

b. Run the self-driving car simulator, select screen resolution, automation mode, and track for the car.

6. Print the values for steering, throttle, and speed at the console respectively.

**3.2 PROGRAM**

**//CODE FOR TESTING OF SELF DRIVING CAR SIMULATION**

import pandas as pd

import numpy as np

import os

import matplotlib.pyplot as plt

from sklearn.utils import shuffle

from sklearn.model\_selection import train\_test\_split

import matplotlib.image as mpimg

from imgaug import augmenters as iaa

import cv2

import random

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Convolution2D,Flatten,Dense

from tensorflow.keras.optimizers import Adam

import socketio

import eventlet

from flask import Flask

from tensorflow.keras.models import load\_model

import base64

from io import BytesIO

from PIL import Image

def getName(filePath):

return filePath.split('\\')[-1]

def importDataInfo(path):

columns = ['Center','Left', 'Right', 'Steering', 'Throttle', 'Brake', 'Speed']

data = pd.read\_csv(os.path.join(path, 'driving\_log.csv'), names = columns)

#### REMOVE FILE PATH AND GET ONLY FILE NAME

#print(getName(data['center'][0]))

data['Center']=data['Center'].apply(getName)

#print(data.head())

print('Total Images Imported',data.shape[0])

return data

path = 'myData'

data = importDataInfo(path)

def balanceData(data,display=True):

nBin = 31

samplesPerBin = 1000

hist, bins = np.histogram(data['Steering'], nBin)

if display:

center = (bins[:-1] + bins[1:]) \* 0.5

plt.bar(center, hist, width=0.06)

plt.plot((np.min(data['Steering']), np.max(data['Steering'])), (samplesPerBin, samplesPerBin))

plt.show()

removeindexList = []

for j in range(nBin):

binDataList = []

for i in range(len(data['Steering'])):

if data['Steering'][i] >= bins[j] and data['Steering'][i] <= bins[j + 1]:

binDataList.append(i)

binDataList = shuffle(binDataList)

binDataList = binDataList[samplesPerBin:]

removeindexList.extend(binDataList)

print('Removed Images:', len(removeindexList))

data.drop(data.index[removeindexList], inplace=True)

print('Remaining Images:', len(data))

if display:

hist, bins = np.histogram(data['Steering'], (nBin))

plt.bar(center, hist, width=0.06)

plt.plot((np.min(data['Steering']), np.max(data['Steering'])), (samplesPerBin, samplesPerBin))

plt.show()

return data

def loadData(path, data):

imagesPath = []

steering = []

for i in range(len(data)):

indexed\_data = data.iloc[i]

print(indexed\_data)

imagesPath.append(os.path.join(path, 'IMG' , indexed\_data[0]))

print(os.path.join(path, 'IMG' , indexed\_data[0]))

steering.append(float(indexed\_data[3]))

imagesPath = np.asarray(imagesPath)

steering = np.asarray(steering)

return imagesPath, steering

xTrain, xVal, yTrain, yVal = train\_test\_split(imagesPath, steerings, test\_size=0.2,random\_state=10)

print('Total Training Images: ',len(xTrain))

print('Total Validation Images: ',len(xVal))

data = balanceData(data,display=True)

def augmentImage(imgPath,steering):

##pan

img = mpimg.imread(imgPath)

if np.random.rand() < 0.5:

pan = iaa.Affine(translate\_percent={"x": (-0.1, 0.1), "y": (-0.1, 0.1)})

img = pan.augment\_image(img)

##zoom

if np.random.rand() < 0.5:

zoom = iaa.Affine(scale=(1, 1.2))

img = zoom.augment\_image(img)

##brightness

if np.random.rand() < 0.5:

brightness = iaa.Multiply((0.4, 1.2))

img = brightness.augment\_image(img)

##flip

if np.random.rand() < 0.5:

img = cv2.flip(img, 1)

steering = -steering

return img, steering

imgRe, st = augmentImage('Test.jpg',0)

plt.imshow(imgRe)

plt.show()

def preProcess(img):

img = img[60:135,:,:]

img = cv2.cvtColor(img, cv2.COLOR\_RGB2YUV)

img = cv2.GaussianBlur(img, (3, 3), 0)

img = cv2.resize(img, (200, 66))

img = img/255

return img

imgRe = preProcess(mpimg.imread('Test.jpg',0))

plt.imshow(imgRe)

plt.show()

def batchGen(imagesPath, steeringList, batchSize, trainFlag):

while True:

imgBatch = []

steeringBatch = []

for i in range(batchSize):

index = random.randint(0, len(imagesPath) - 1)

if trainFlag:

img, steering = augmentImage(imagesPath[index], steeringList[index])

else:

img = mpimg.imread(imagesPath[index])

steering = steeringList[index]

img = preProcess(img)

imgBatch.append(img)

steeringBatch.append(steering)

yield (np.asarray(imgBatch),np.asarray(steeringBatch))

def createModel():

model = Sequential()

model.add(Convolution2D(24, (5, 5), (2, 2), input\_shape=(66, 200, 3), activation='elu'))

model.add(Convolution2D(36, (5, 5), (2, 2), activation='elu'))

model.add(Convolution2D(48, (5, 5), (2, 2), activation='elu'))

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

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

model.add(Flatten())

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

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

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

model.add(Dense(1))

model.compile(Adam(lr=0.0001),loss='mse')

return model

model = createModel()

model.summary()

history = model.fit(batchGen(xTrain, yTrain, 100, 1),steps\_per\_epoch=300,epochs=10,

validation\_data=batchGen(xVal, yVal, 100, 0),validation\_steps=200)

model.save('model.h5')

print('Model Saved')

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.legend(['Training', 'Validation'])

plt.title('Loss')

plt.xlabel('Epoch')

plt.show()

**//CODE FOR FINAL EXECUTION**

print('Setting UP')

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3'

import socketio

import eventlet

import numpy as np

from flask import Flask

from tensorflow.keras.models import load\_model

import base64

from io import BytesIO

from PIL import Image

import cv2

#### FOR REAL TIME COMMUNICATION BETWEEN CLIENT AND SERVER

sio = socketio.Server()

#### FLASK IS A MICRO WEB FRAMEWORK WRITTEN IN PYTHON

app = Flask(\_\_name\_\_) # '\_\_main\_\_'

maxSpeed = 10

def preProcess(img):

img = img[60:135,:,:]

img = cv2.cvtColor(img, cv2.COLOR\_RGB2YUV)

img = cv2.GaussianBlur(img, (3, 3), 0)

img = cv2.resize(img, (200, 66))

img = img/255

return img

@sio.on('telemetry')

def telemetry(sid, data):

speed = float(data['speed'])

image = Image.open(BytesIO(base64.b64decode(data['image'])))

image = np.asarray(image)

image = preProcess(image)

image = np.array([image])

steering = float(model.predict(image))

throttle = 1.0 - speed / maxSpeed

print('{} {} {}',({steering}, {throttle}, {speed}))

sendControl(steering, throttle)

@sio.on('connect')

def connect(sid, environ):

print('Connected')

sendControl(0, 0)

def sendControl(steering, throttle):

sio.emit('steer', data={

'steering\_angle': steering.\_\_str\_\_(),

'throttle': throttle.\_\_str\_\_()

})

if \_\_name\_\_ == '\_\_main\_\_':

model = load\_model('model1.h5')

app = socketio.Middleware(sio, app)

### LISTEN TO PORT 4567

eventlet.wsgi.server(eventlet.listen(('', 4567)), app)

**3.3 RESULT:**

**SOFTWARE -RESULT**

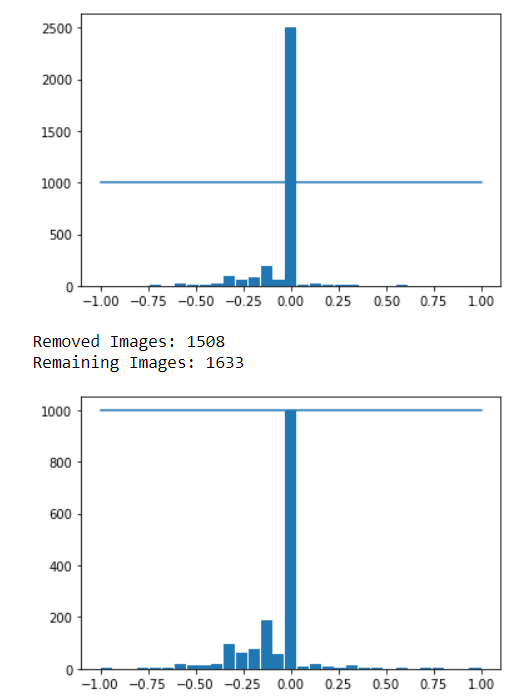
****

Fig 3.1- Histogram

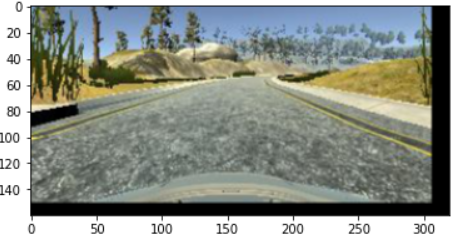
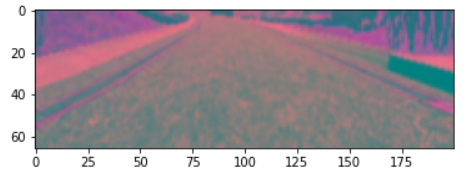


Fig 3.2 - Augmented Data

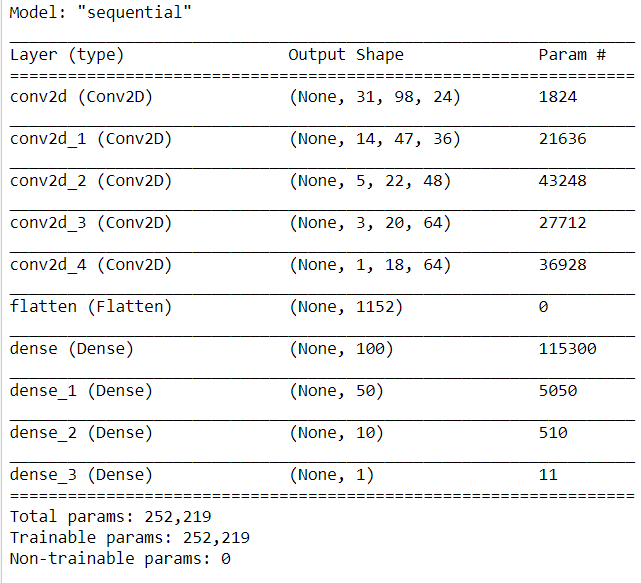


Fig 3.3- CNN plot

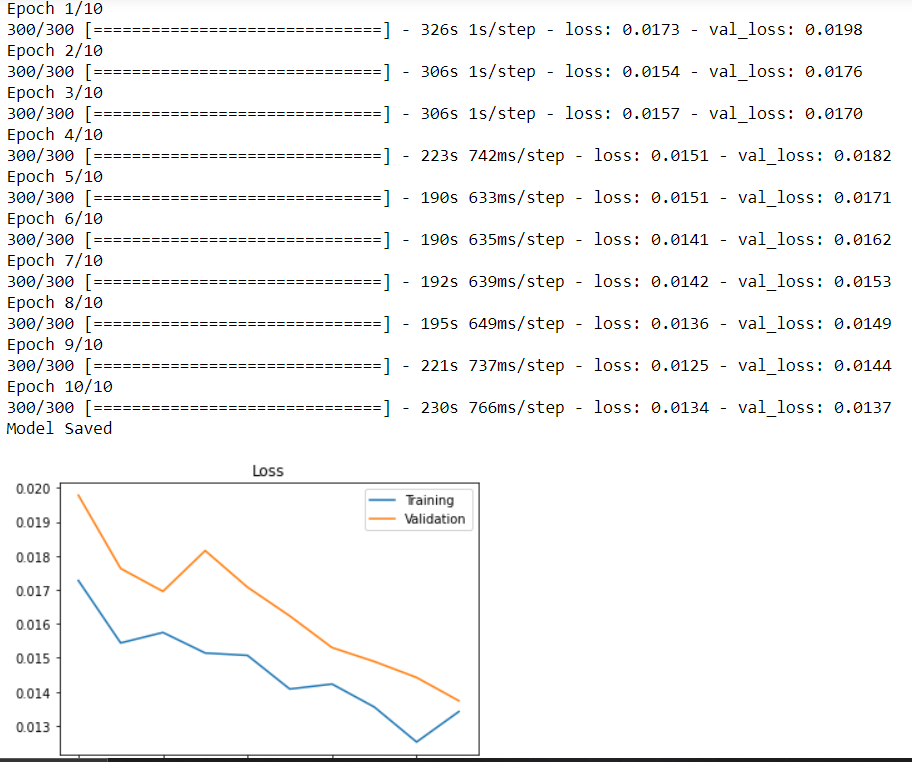


Fig 3.4- Trained model

**CHAPTER 4:**

**CONCLUSIONS AND SCOPE FOR FUTURE WORK**

**4.1 CONCLUSION**

The introduction of autonomous vehicles (AVs) into the world's congested highways signifies a paradigm shift in how human mobility is envisioned and implemented. We are currently living in a period of significant transition, in which active control of the vehicle is being transferred from the human driver to the on-board computer systems (85). The disparity reflects this change. Our project would be used in the development of self-driving cars to allow adequate driverless rides, to gain deep learning knowledge and analyse data to predict vehicle movement, and to train our model with deep learning based on a predefined parameter such as steering angle, throttle, speed, and so on. Use of CNN (convolutional neural network) concepts to train our model, apply self-learning and decision-making skills in the model through simulation, and implement future technologies to reduce human efforts, depending on the circumstances.

**4.2 SCOPE FOR FUTURE WORK**

The vehicle is capable of performing all driving operations and does not require the driver to be ready to take over navigation. However, under some conditions, such as off-road driving or other forms of aberrant or hazardous scenarios, the quality of the ADS navigation may deteriorate. The driver might be able to take control of the vehicle. The ADS technology is sophisticated enough that the vehicle can perform all driving functions regardless of the weather. The driver might be able to take control of the vehicle.

**REFERENCES**

# End-to-End Deep Learning for Self-Driving Cars By [Mariusz Bojarski](https://developer.nvidia.com/blog/author/mbojarski/), [Ben Firner](https://developer.nvidia.com/blog/author/bfirner/), [Beat Flepp](https://developer.nvidia.com/blog/author/bflepp/), [Larry Jackel](https://developer.nvidia.com/blog/author/ljackel/), [Urs Muller](https://developer.nvidia.com/blog/author/umuller/), [Karol Zieba](https://developer.nvidia.com/blog/author/kzieba/) and [Davide Del Testa](https://developer.nvidia.com/blog/author/ddeltesta/)

1. Outperforming Tensorflow’s Default Auto Differentiation Optimizers, with Interactive Code [Manual…. (2018). Towards Data Science. Retrieved 19 March 2018, from
2. Outperforming Tensorflow’s Default Auto Differentiation Optimizers, with Interactive Code [Manual…. (2018). Towards Data Science. Retrieved 20 March 2018