SELF DRIVING CAR SIMULATION

Submitted in partial fulfillment of the requirements of the degree of

T. E. Computer Engineering

By

Sougat Ganguly Roll No: 48 PID:172114

Jerin Varghese Roll No: 55 PID:172049

Arbaz Khan Roll No: 60 PID:172017

Guide(s):

Mr. Rupesh Mishra

Asst. Professor

Department of Computer Engineering

St. Francis Institute of Technology

(Engineering College)

University of Mumbai

2019-2020

CERTIFICATE

This is to certify that the project entitled **“Self Driving Car Simulation”** is a bonafide work of “ **Sougat Ganguly (Roll No: 48), Jerin Varghese (Roll No: 55), Arbaz Khan (Roll No: 60)”** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of T.E. in Computer Engineering.

(Mr. Rupesh Mishra)	(Dr. Kavita Sonawane)
Guide	Head of Department

Project Report Approval for T.E.

This project report entitled ***Self Driving Car Simulation*** by **Mr. Jerin Varghese, Mr. Sougat Ganguly, Mr. Arbaz Khan** is approved for the degree of ***T.E. in Computer Engineering.***

Examiners

1.---

2.---

Date: 07/06/2020

Place: Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed

----

(Signature)

Sougat Ganguly   Roll no: 48

Date: 07/06/2020

Abstract

Self-driving cars have become a trending subject with a significant improvement in the technologies in the last decade. The project purpose is to train a convolutional neural network to drive an autonomous car on the tracks of Udacity's Car Simulator environment. Udacity has released the simulator as an open source software and enthusiasts can use this simulator to train a virtual car how to drive using camera images and deep learning. Driving a car in an autonomous manner requires learning to control steering angle, throttle and brakes. Behavioral cloning technique is used to mimic human driving behavior in the training mode on the track. That means a dataset is generated in the simulator by a user driven car in training mode, and the deep neural network model then drives the car in autonomous mode.

Though the models performed well for the track it was trained with, the real challenge was to generalize this behavior on a second track available on the simulator. The dataset for Track_1 was used as the training set to drive the car autonomously on Track_2 whose path is new to the model. To tackle this problem, image processing and different data augmentation techniques were used, which allowed extracting as much information and features in the data as possible. Ultimately, the car was able to run on Track_2 generalizing well. The use of Convolutional NN made the task of handling images in the NN computationally feasible. The project aims at reaching the same accuracy on real time data in the future.

Contents

List of Figures

Chapter 1
Introduction

1.1 Project Description

To develop a self-driving car system using Convolutional Neural Networks. Generalising the model by training the model on one track and testing it on another unforeseen track. Udacity's Self-Driving Car simulator was built for Udacity's Self-Driving Car Nanodegree, to teach students how to train cars how to navigate road courses using deep learning.

1.2 Problem Formulation

Self-driving car is still a domain where extensive research work is carried out. Surpassing human accuracy was not the real challenge. The real challenge is surpassing or atleast coming close to the Bayes accuracy. But the undeniable fact is that self-driving cars are the future. Another huge task is overcoming the moral decision making in self-driving cars, however, this issue has not been addressed in our project.

A lot of research has been done on self-driving cars in the recent years, and big names such as Google are also one of the many who have started working on these cars. It can be added that the technology of self-driving cars does make it safer for humans, in case of accidents, and the autonomous cars also help improve the environment because these vehicles are cost efficient and help reduce greenhouse emissions by 90 percent. Even your own car can be converted into a self-driving car considering it is compatible, but the question is - do we really need self-driving cars? Even after a huge spending by tech companies around the world research still shows that the results are not positive enough, meaning that more people are unwilling to change and they want to stick to the traditional method of driving on their own.

1.3 Motivation

According to the World Health Organization, more than 1 million people lose their lives on the road due to car accidents, and C2ES (Center for Climate and Energy Solutions) states that about 60% of total energy consumed by transportation is from automobiles. These numbers show that cars cause serious casualties and are a major source of greenhouse gas emission. There have been attempts to solve these problems, but none of the solutions have been particularly effective. However, self-driving technology has potential to change all these problems. If self-driving cars can be driven safer and much more efficiently, it could save valuable lives and preserve the environment.

1.4 Proposed Solution

The simulator can be used to collect data by driving the car in the training mode using 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 serverandpipes 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 APIover 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 3step. Detailed description of CNN models experiment and used can be referred to in the chapter on network architectures. 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.

Behavioral cloning technique is used to mimic human driving behavior in the training mode on the track. That means a dataset is generated in the simulator by a user driven car in training mode, and the deep neural network model then drives the car in autonomous mode.

1.5 Scope of The Project

Autonomous cars aim at increasing human comfort, however, humans tend to not rely completely on the auto-driving feature of the cars. Hence, the more reliable the system is, the more humans can be at ease.

Simulation isn’t only limited to programming the self-driving cars, it could actually soon tell manufacturers how many autonomous cars they need to produce. It may even demonstrate how many autonomous cars an urban center needs to have in order to make sure the entire population can consistently get where they need to go without actually owning a car. Bolstering this future of simulation determining car needs, research from the Organization for Cooperation and Development found that 90% of urban cars will be unnecessary when autonomous cars hit the market in large quantities. This number was, of course, found by running a simulation of driving activities based on data from urban centers and finding the redundancies in the system.

Simulation will be involved in nearly every aspect of autonomous vehicles. From CFD analysis on car designs to real-time traffic simulations allow the AI systems to make choices and develop driving habits. Simulation is at the core of it all.

Chapter 2
Review of Literature

In order to understand the development of research in autonomous driving in the last years, it is important to conduct a literature review to understand the different fields of application through which autonomous driving has evolved as well as to identify research gaps. Therefore in the next sections the research process, methodology and findings of the literature review are presented.

Methodology & Data Collection Procedure

The research conducted is focused in a systematic keyword search in the topic section of literature databases, including EBSCO, Science Direct, Emerald and ISI web of knowledge. The search conducted included the specific terms, “Autonomous driving”, “Self driving car” and “Driverless car” either in the title, keywords or abstract and including only academic journals listed in the mentioned databases. Hence, the aim of this research was not to find all literature regarding Autonomous Driving that because of its size would lead to an enormous amount of results due to the extensive applications, testing and research in other fields (robotics, underwater vehicles, military, aeronautics, space vehicles, etc.), but to achieve an overview and overall classification to identify current gaps in the scientific literature body. Therefore taking into consideration only the literature publications relevant for roads, traffic, crossroads and studies related to commuting, transportation or production, and including all relevant publications found related to the automotive industry, as well as also considering papers in other topics that acknowledge that the application could be relevant for self-driving cars.

The search yielded 483 papers, 154 of which were found in EBSCO, 122 in Science Direct, 8 in Emerald and 199 in ISI Web of Knowledge. Furthermore, a thorough analysis of abstracts, titles and journals was conducted in order to eliminate duplicates and results that were found in more than one database, leading to the elimination of 63 repetitions, some of which were even found 3 times, reflecting the thoroughness of the search, delivering a total amount of 420 publications. Finally a last filtering process eliminated publications not fulfilling scientific standards nor a peer-review process leads to the exclusion of 21 further publications delivering a final number of 399 Publications, which can be found in Appendix I.

A set of analysis explained in the next sections were then conducted with the remaining 399 publications in order to understand the current state of the literature, its focus and development over time.

Findings

Analysis of publications over time

Through this method, good insight can be gained on the origin and development of a research field over time (Dahlander & Gann, 2010). As shown in Figure 2 we can observe the development of self-driving cars research throughout the last decades, demonstrating at the same time a continuous increase of the interest on the topic as well as important milestones that can also influence the topic development and research such as in this case the DARPA Grand Challenges in the beginning of 2004 and end of 2005 or the Urban Challenge in the end of 2007 (DARPA, 2015), that were relatively important events in the field. Furthermore, from 2013 to 2014 a rapid increase of 60.8% can be observed, which can be related to the actual interest on the topic.

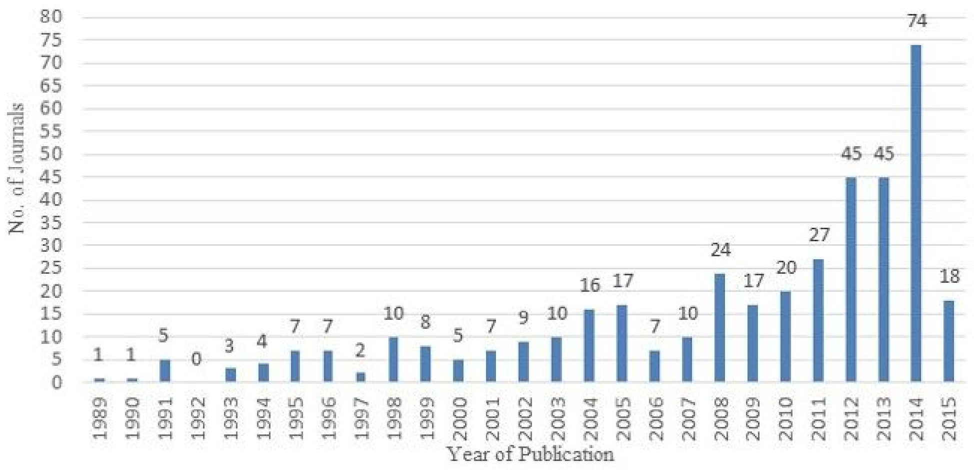


Figure 1: Analysis of Publications over time

Journal Count Analysis

Furthermore a journal count analysis was conducted in order to understand which journals were more dedicated to the topic depending on the amount of articles released, all journals with more than 5 publications are shown in Figure 3. "IEEE Transactions on Intelligent transportation systems." and "Robotics and Autonomous Systems" resulted to be the ones more focused on the topic with 17 publications each. This analysis shows the first hint on the natural focus of the literature in the technology development phase, as most of the listed journals are mainly technical and not specialized in the field of business and management

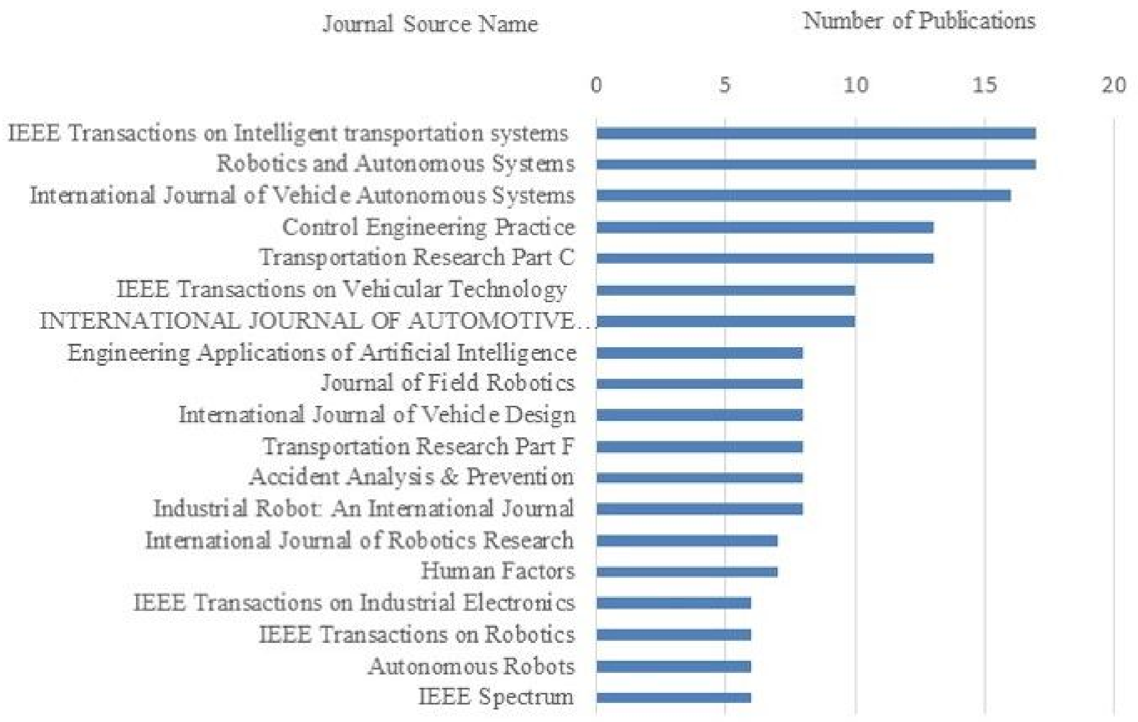
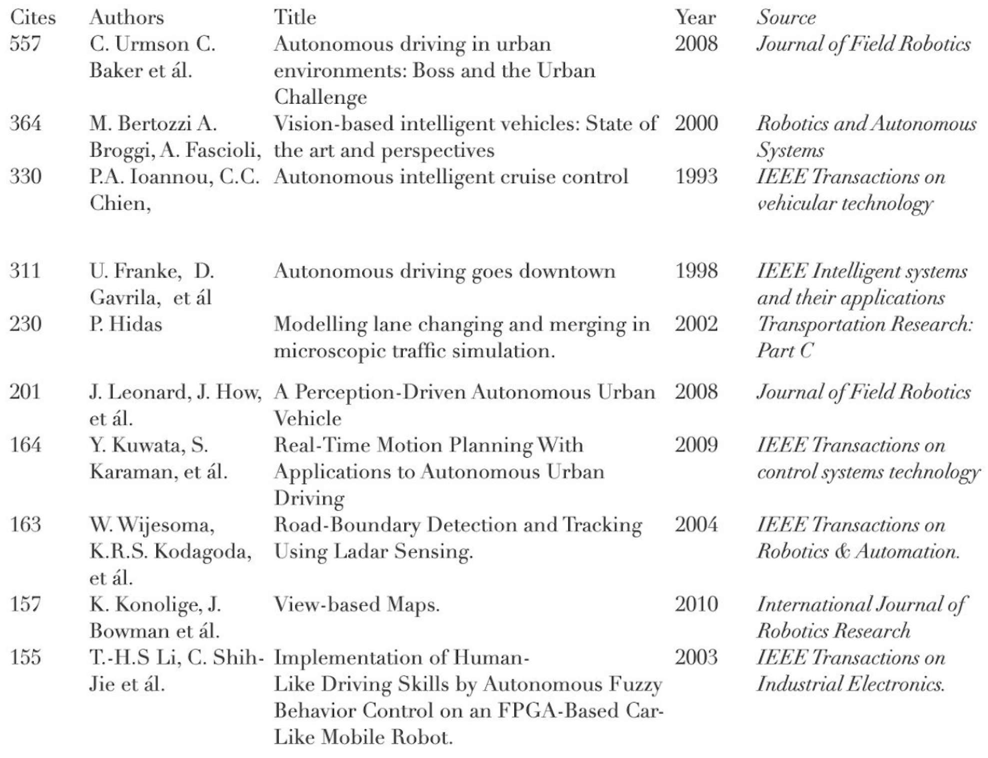


Figure 2: Journal Publication Count

Citation Analysis

In addition and in order to examine the impact of published articles in a field (Leone, et al., 2012) , the well-established procedure of citation analysis was conducted, known as one of the most effective tools to analyze research performance (Chubin & Hackett , 1990; Martin, 1996) it was conducted by analyzing all 399 publications found, through the Publish or Perish tool (Harzing, 2007), the search yielded the citation count of 374 of the 399 publications leading to 25 results (6.2%) which were not found by Publish or Perish. The top results are reflected in the following Table.



Cites	Authors	Title	Year	*Source*
557	C. Urmson C. Baker et ál.	Autonomous driving in urban environments: Boss and the Urban Challenge	2008	*Journal of Field Robotics*
364	M. Bertozzi A. Broggi, A. Fascioli,	Vision-based intelligent vehicles: State of the art and perspectives	2000	*Robotics and Autonomous Systems*
330	P.A. Ioannou, C.C. Chien,	Autonomous intelligent cruise control	1993	*IEEE Transactions on vehicular technology*
311	U. Franke, D. Gavrila, et ál	Autonomous driving goes downtown	1998	*IEEE Intelligent systems and their applications*
230	P. Hidas	Modelling lane changing and merging in microscopic traffic simulation.	2002	*Transportation Research: Part C*
201	J. Leonard, J. How, et ál.	A Perception-Driven Autonomous Urban Vehicle	2008	*Journal of Field Robotics*
164	Y. Kuwata, S. Karaman, et ál.	Real-Time Motion Planning With Applications to Autonomous Urban Driving	2009	*IEEE Transactions on control systems technology*
163	W. Wijesoma, K.R.S. Kodagoda, et ál.	Road-Boundary Detection and Tracking Using Ladar Sensing.	2004	*IEEE Transactions on Robotics & Automation.*
157	K. Konolige, J. Bowman et ál.	View-based Maps.	2010	*International Journal of Robotics Research*
155	T.-H.S Li, C. Shih-Jie et ál.	Implementation of Human-Like Driving Skills by Autonomous Fuzzy Behavior Control on an FPGA-Based Car-Like Mobile Robot.	2003	*IEEE Transactions on Industrial Electronics.*

.

Chapter 3
System Analysis

3.1 Functional Requirements

Major functionalities associated with the user are:

- Lets the users run the simulation in training and autonomous modes.
- Enable users to record the training data.
- Access to two different maps to record the training data.
- Access to two different maps to run the model.
- Enable users to save the data to your local device.

Major functionalities associated with the system are as follows:

- Generate values for acceleration, steering angle and brakes.
- Generate images from cameras at 3 different angles.
- Displays the car simulation.

Interface Requirements:

1. Press R to record the training data.
2. Select autonomous mode to train the training model.

3.2 Non-Functional Requirements

3.2.1 Performance

The computer running the software must have a powerful CPU and GPU so that the data can be processed faster and there won’t be lag in the entire process of summarizing the news articles.

3.2.2 Reliability

The system takes in the inputs without any errors and predicts the summary accurately or produces the desired summary of the CNN model used. The data can be very reliable if the user runs the car in training mode at the center of the road properly.

3.2.3 Usability

The system is easy to handle and navigates in the most expected way with no delays. The system interacts with the user in a very friendly manner making it easier for the users to use the system. The training data can be used for a different model.

3.2.4 Maintainability

There will be no maintenance requirement for the software. The entire process is automated.

3.3 Specific Requirements

3.3.1 User Interfaces

- Udacity Self Driving Car Simulation

3.3.2 Hardware Requirements

- CPU Type: Intel i7 or above
- Clock speed: 2.7GHz
- Ram size: 8GB and above
- Hard Disk capacity: 100GB and above
- System: 64bit OS, x64 processor
- Working keyboard

3.3.3 Software Requirements

- Operating System: Windows 10
- Python 3.5 or above

3.4 Use-Case Diagrams and Description

Use Case Diagram:

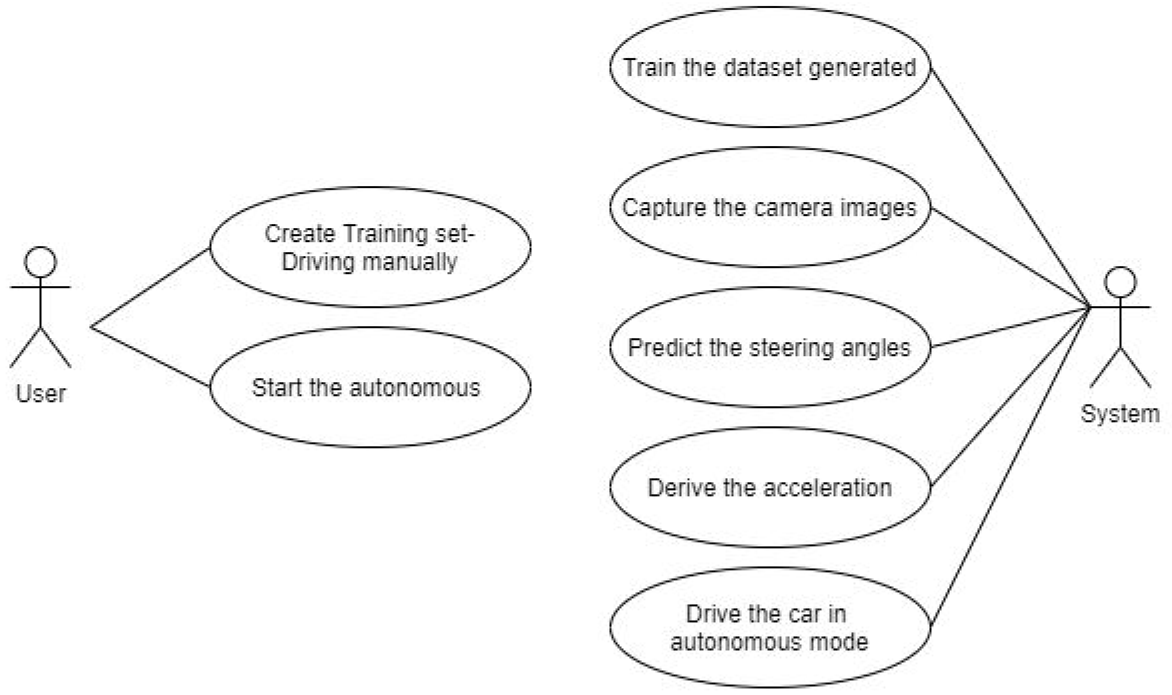

Train the dataset generated
Capture the camera images
Create Training set-Driving manually
Predict the steering angles
Start the autonomous
User
System
Derive the acceleration
Drive the car in autonomous mode

Fig 3: Use Case Diagram for Self driving Car Simulation.

A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams model the functionality of a system using actors and the use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "System" is something being developed and operated, such as a website. The "Actors" are people or entities operating under defined roles within the system.

Chapter 4
Analysis Modelling

4.1 Class Diagram and Activity Diagram

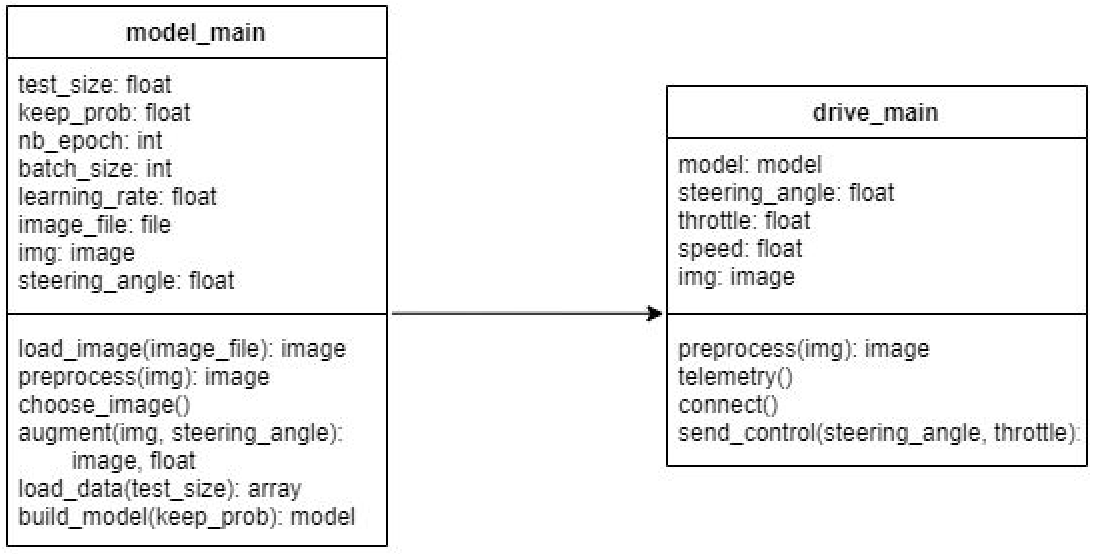


Fig 4: Class Diagram for Self Driving Car Simulation

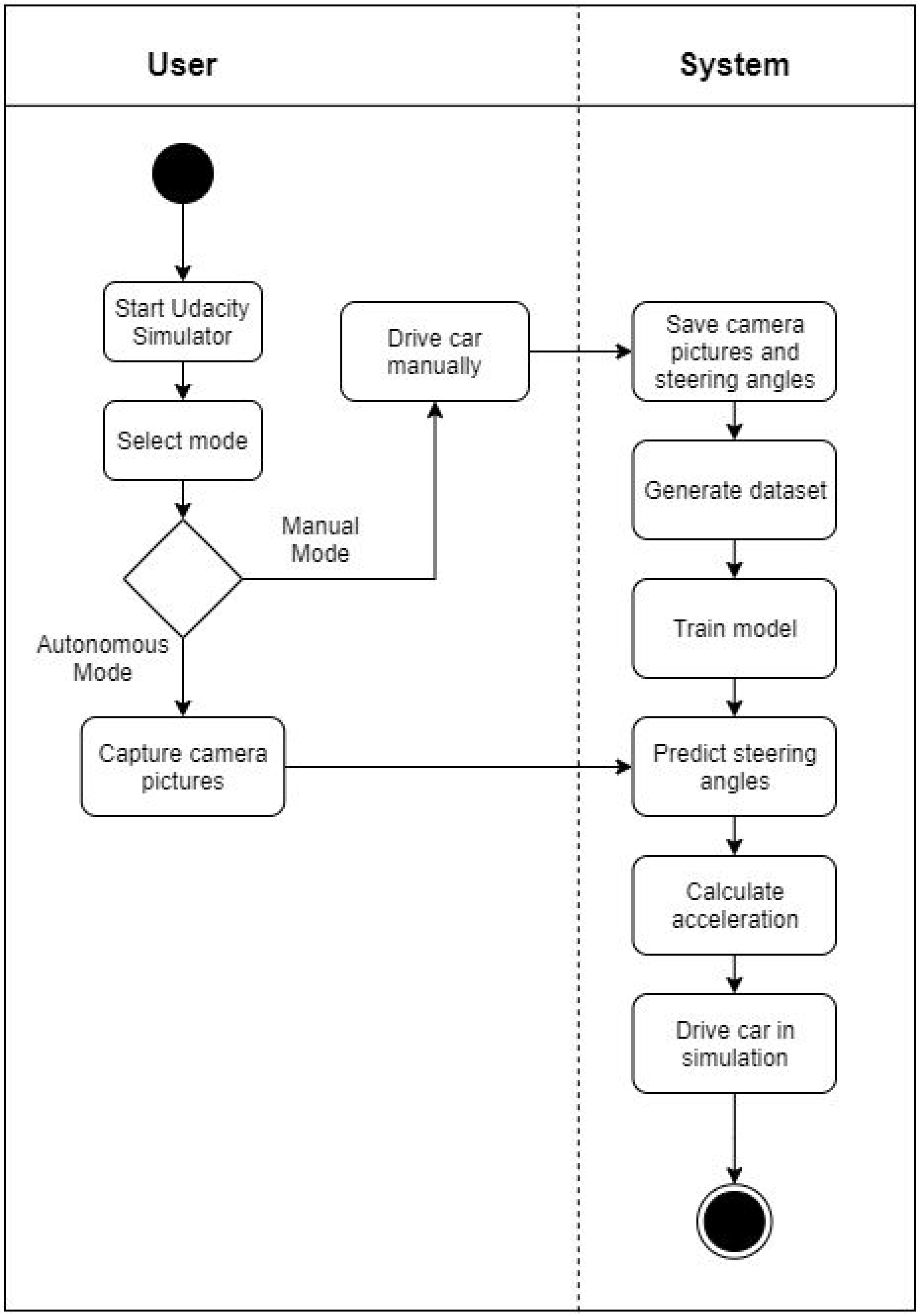


Fig 5: Activity Diagram for Self Driving Car Simulation

- User decides which model to select out of the two models (BART or Seq2Seq).
- After choosing any one model the user has to give a data input in form of text to obtain its summarization.

- If the Seq2seq model is selected then the system loads the data, generates vocabulary, and loads Glove (Global vector) for word embeddings. After word embedding is done, load the seq2seq + attention model, train the model, test it and after inference the user will obtain a suitable summary of their input text.
- If a BART pretrained model is selected, the system loads the data and tokenizes it, the model encodes the input evaluates it and decodes after inference to obtain a summary of the given input text.

4.2 Functional Modelling

Data Flow Diagram

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on flow of information, where data comes from, where it goes and how it gets stored.

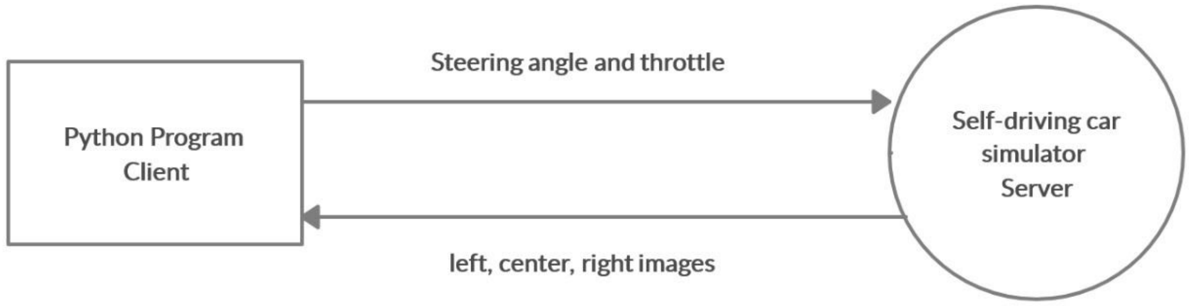

Steering angle and throttle
Python Program Client
Self-driving car simulator Server
left, center, right images

Fig 6: Level 0 DFD for Self Driving Car Simulation.

Level 0 DFD:

This Level is called the Level 0 DFD. It is a basic overview of the whole system or process being analysed or modelled. Here the basic flow of the system is shown. The user gives input which is stored by the system. Based on the input given the system accordingly processes and gives then output to the user.

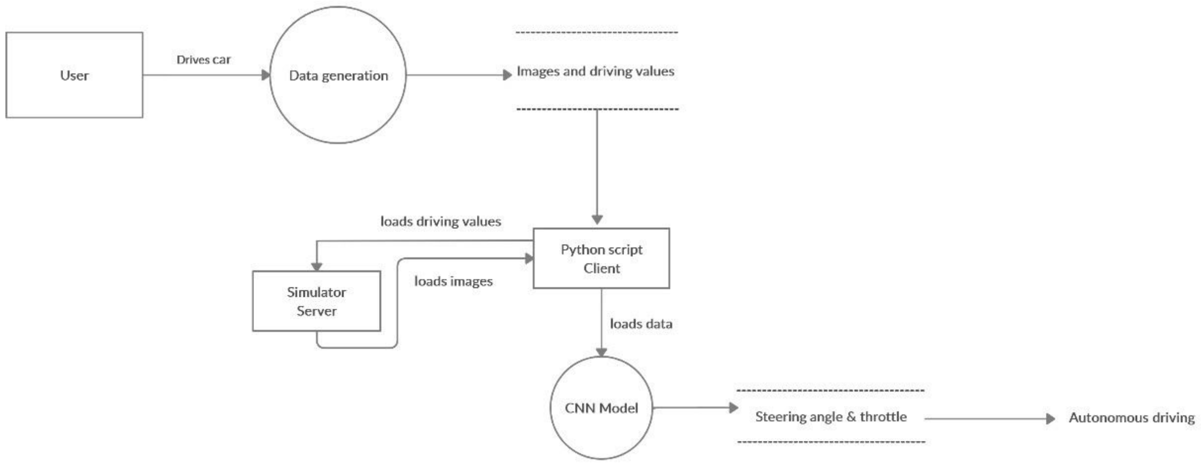


Fig 7: Level 1 DFD for Self Driving Car Simulation.

Level 1 DFD:

DFD Level 1 provides a more detailed breakout of pieces of Context Level DFD. It basically explains the system more in detail.

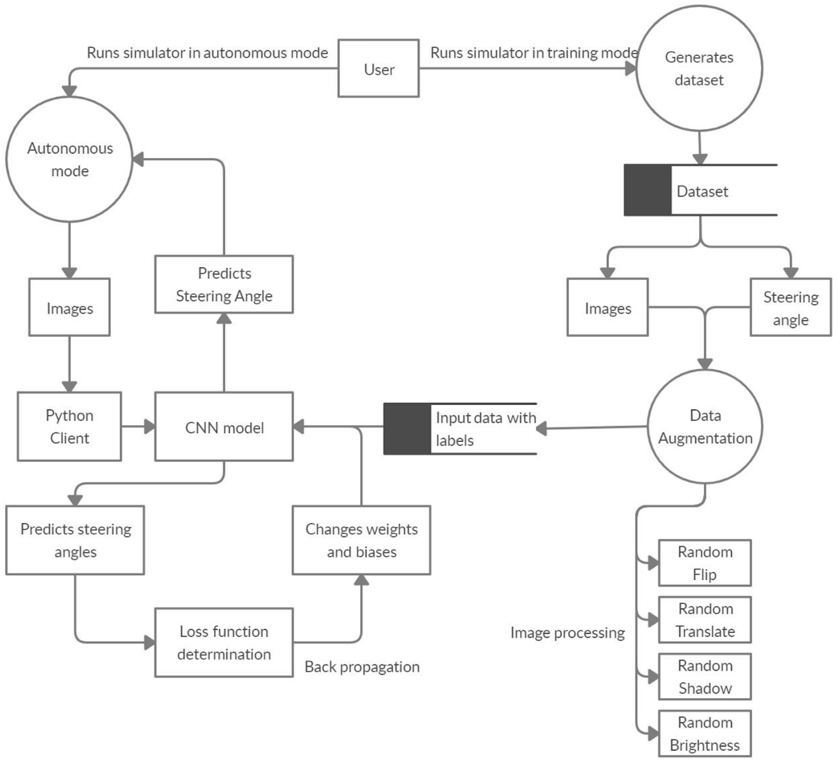


Fig 8: Level 2 DFD for Self Driving Car Simulation.

Chapter 5
Design

5.1 Architectural Design

There were various combinations of architectures tried,predicting the steering angle and input for the car to drive in autonomous mode. Neural Network layers were organized in series and various combinations of Time-Distributed Convolutional layers, MaxPooling, Flatten, Dropout, Dense and so on are used in architectures. The best performing ones are shown in detail. Refer To The modellistingsfor the parameters used to build them.The high-level view of layers used to build the models is shown in the accompanying architecture figures.

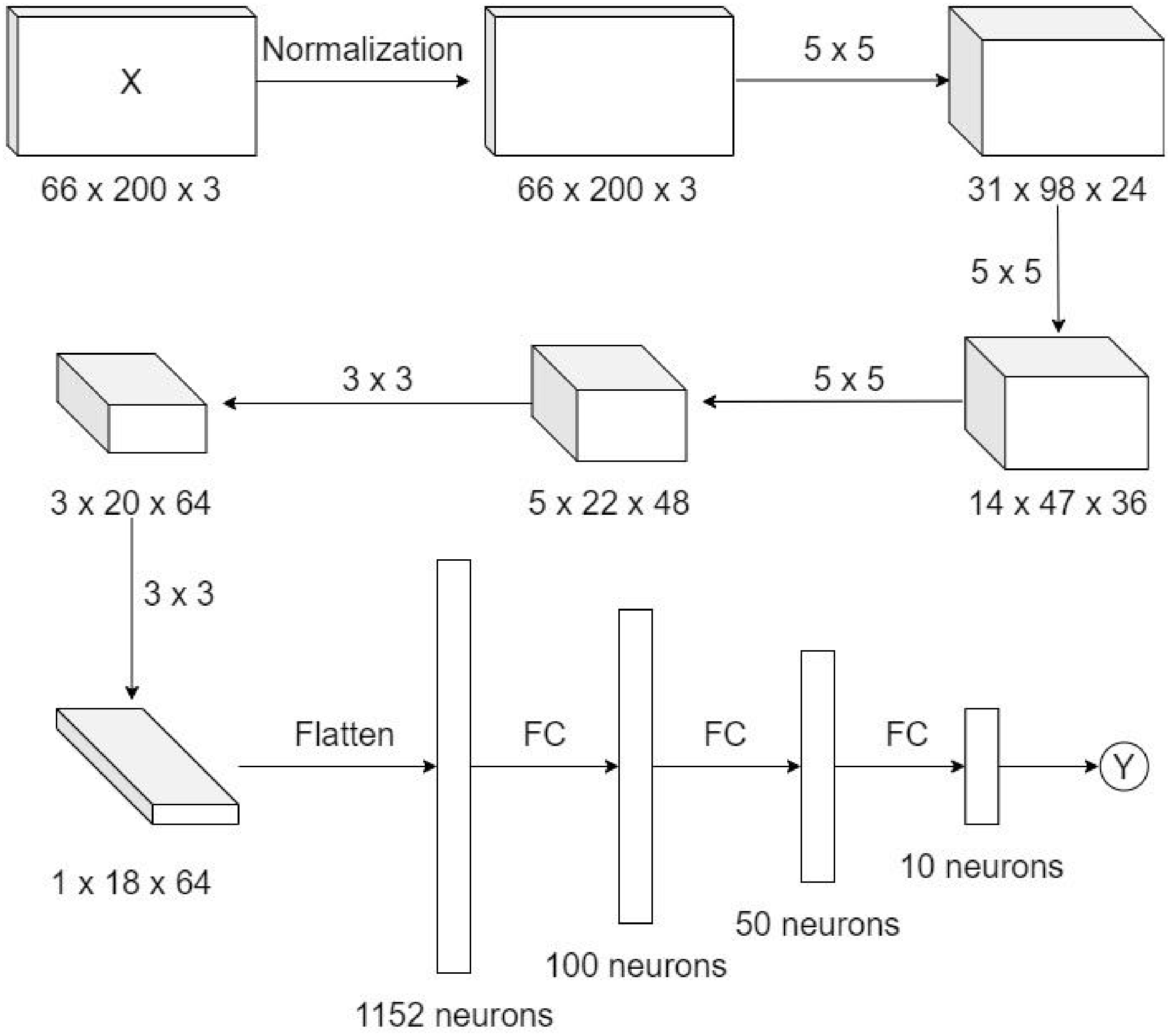


Fig 9: Neural Network design

NVIDIA released an architecture for self-driving cars and it is used in the project for reference to solve the problem and for comparing with the various other architectures tried. Different architectures have been standardized over the years for building sequential models of CNN like AlexNet,VGG-Net, GoogLeNet, ResNet and so on. Model_2 is an architecture similar to AlexNet, with a slight variation by tweaking parameters to suit the problem in the project.

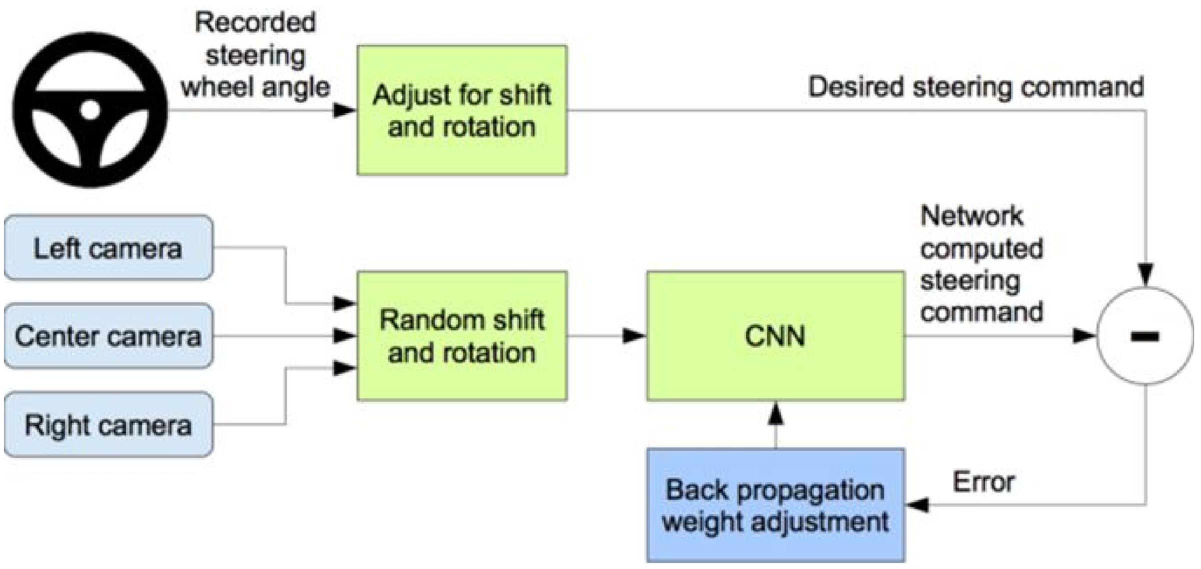

Recorded steering wheel angle
Adjust for shift and rotation
Desired steering command
Left camera
Center camera
Right camera
Random shift and rotation
CNN
Network computed steering command
-
Back propagation weight adjustment
Error

Fig 10: Block Diagram of Training Model

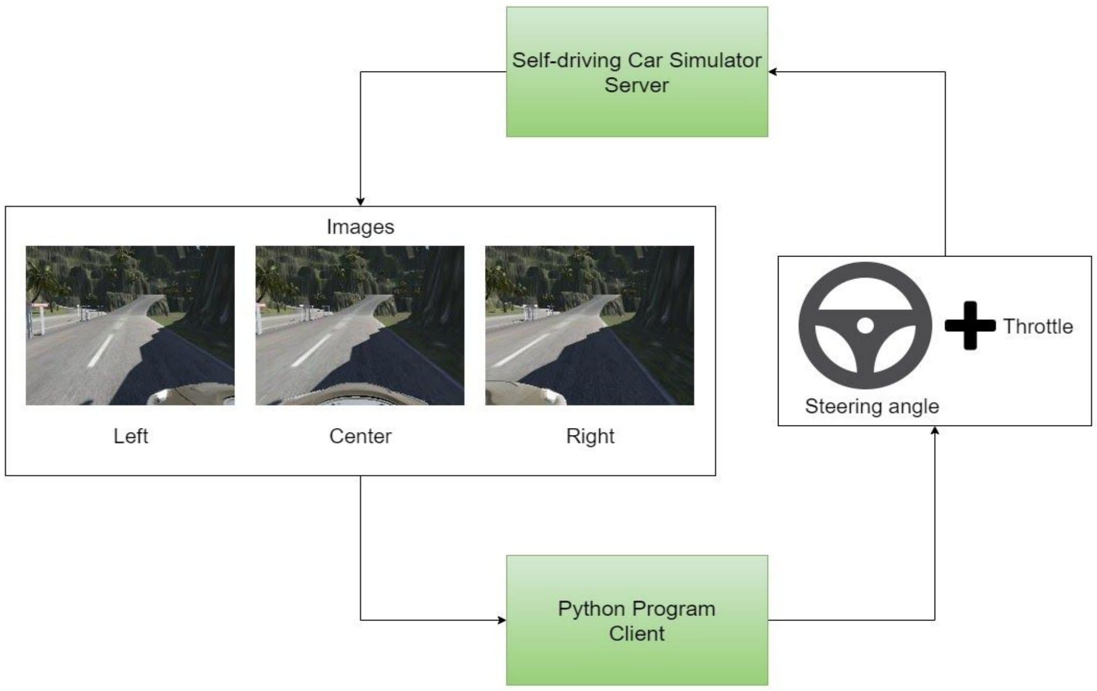


Fig 11: Block Diagram of Prediction Model

5.2 User Interface Design

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

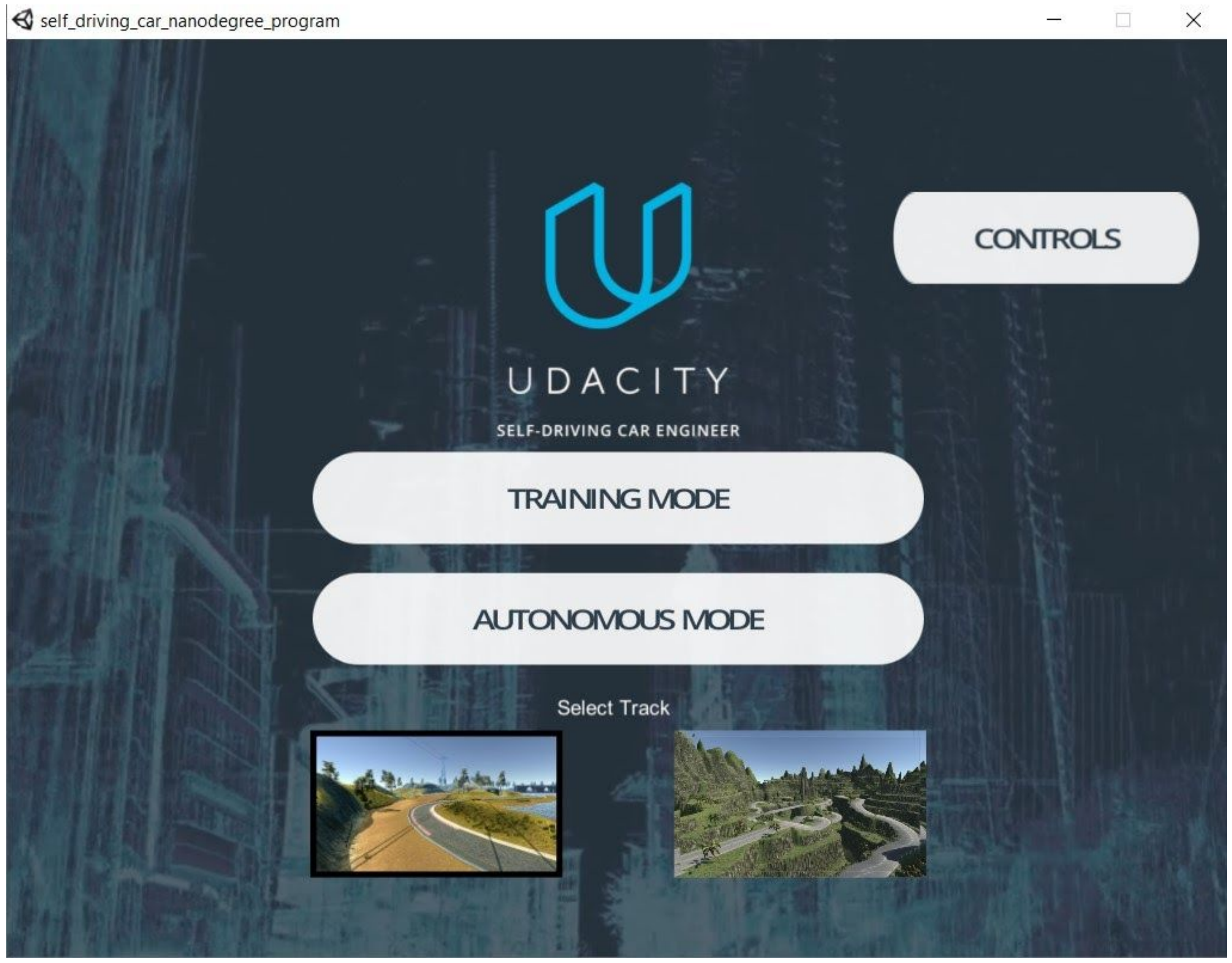

self_driving_car_nanodegree_program
UDACITY
SELF-DRIVING CAR ENGINEER
CONTROLS
TRAINING MODE
AUTONOMOUS MODE
Select Track

Fig 12: Main Menu of Simulator

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 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.

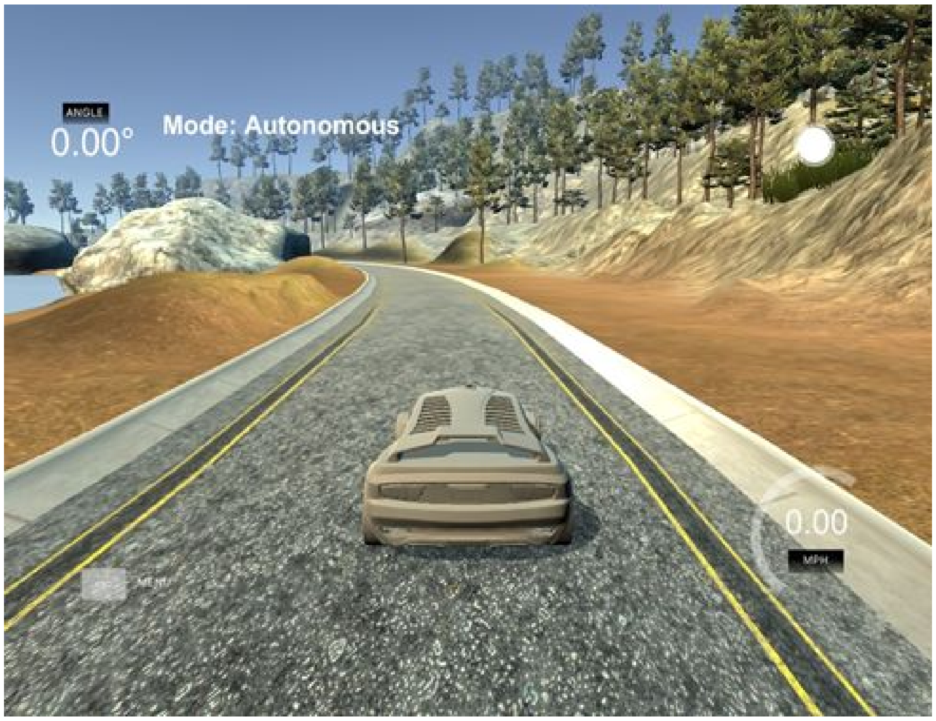


Fig 13: GUI of Track 1

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

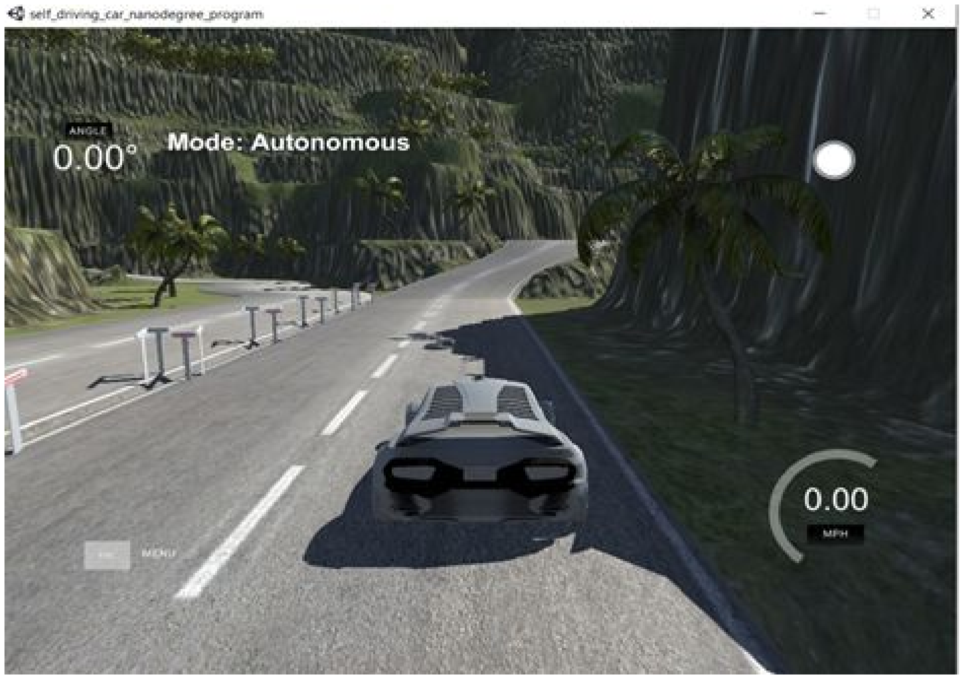


Fig 14: GUI of Track 2

The simulator 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 sets 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.

Chapter 6
Implementation

6.1 Algorithms Used

6.1.1. Convolutional Layer

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels. Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

6.1.2. ELU Layer

The most successful non-linearity for CNN’s is the Exponential Linear unit (ELU), which combats the vanishing gradient problem occurring in sigmoids. ELU is easier to compute and generates sparsity (not always beneficial).

6.1.2. Max Pooling

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

6.1.3. Flattening

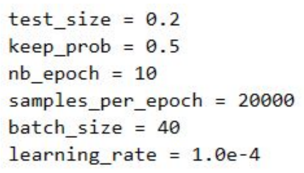
Flattening is the process of converting the resultant 2 dimensional arrays from the pooled feature map to a single long continuous linear vector.

6.1.4. Fully connected layer

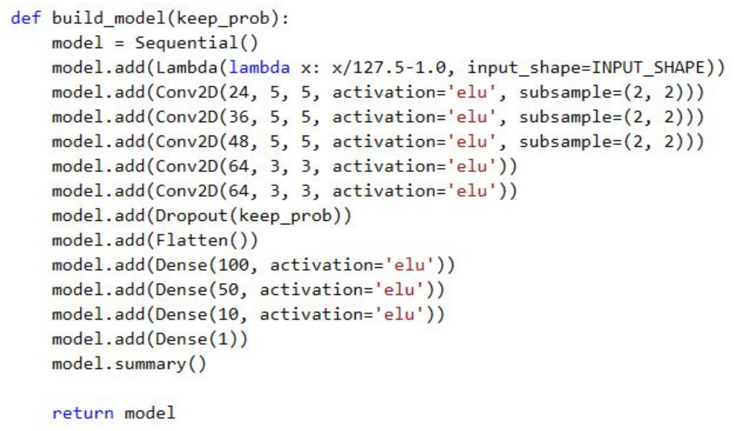
Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

6.2. Code Snippets

6.2.1 Training

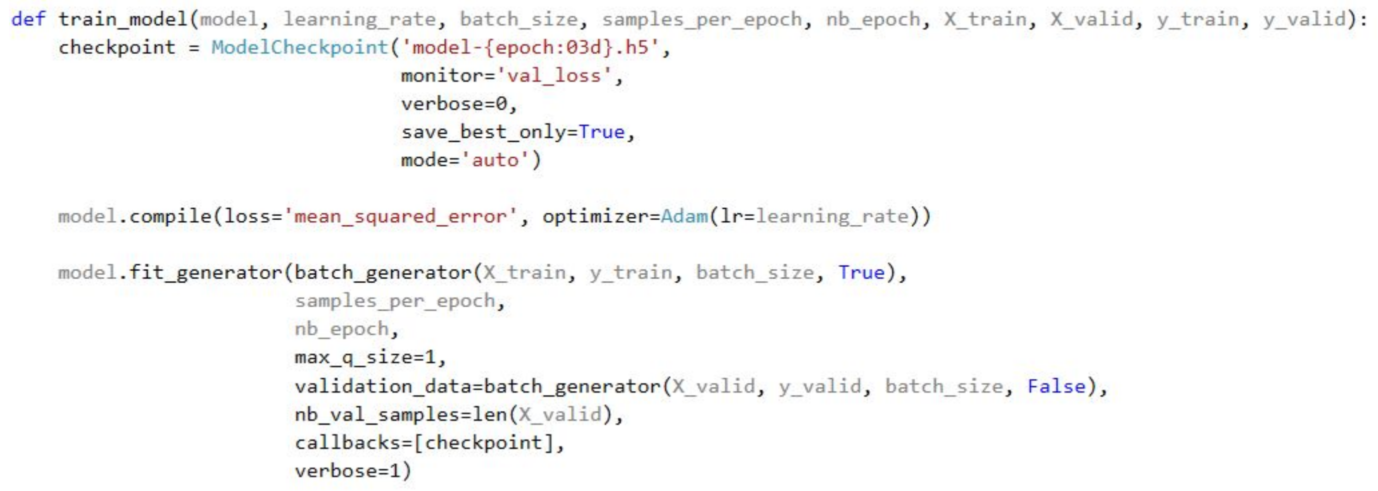


```
test_size = 0.2
keep_prob = 0.5
nb_epoch = 10
samples_per_epoch = 20000
batch_size = 40
learning_rate = 1.0e-4
```



```
def build_model(keep_prob):
    model = Sequential()
    model.add(Lambda(lambda x: x/127.5-1.0, input_shape=INPUT_SHAPE))
    model.add(Conv2D(24, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(36, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(48, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Dropout(keep_prob))
    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.summary()

    return model
```

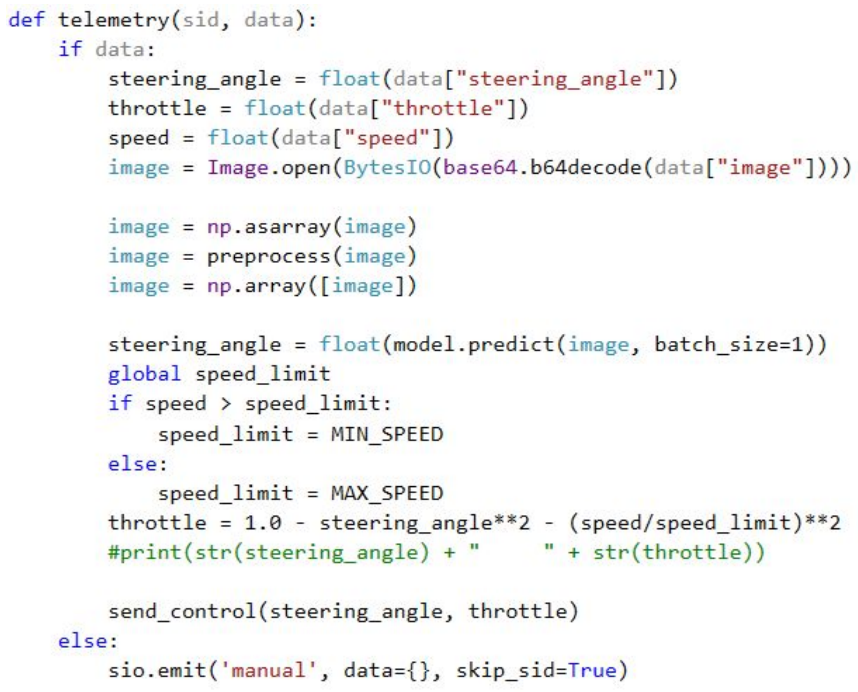


```
def train_model(model, learning_rate, batch_size, samples_per_epoch, nb_epoch, X_train, X_valid, y_train, y_valid):
    checkpoint = ModelCheckpoint('model-{epoch:03d}.h5',
                                 monitor='val_loss',
                                 verbose=0,
                                 save_best_only=True,
                                 mode='auto')

    model.compile(loss='mean_squared_error', optimizer=Adam(lr=learning_rate))

    model.fit_generator(batch_generator(X_train, y_train, batch_size, True),
                        samples_per_epoch,
                        nb_epoch,
                        max_q_size=1,
                        validation_data=batch_generator(X_valid, y_valid, batch_size, False),
                        nb_val_samples=len(X_valid),
                        callbacks=[checkpoint],
                        verbose=1)
```

6.2.2 Prediction



```
def telemetry(sid, data):
    if data:
        steering_angle = float(data["steering_angle"])
        throttle = float(data["throttle"])
        speed = float(data["speed"])
        image = Image.open(BytesIO(base64.b64decode(data["image"])))

        image = np.asarray(image)
        image = preprocess(image)
        image = np.array([image])

        steering_angle = float(model.predict(image, batch_size=1))
        global speed_limit
        if speed > speed_limit:
            speed_limit = MIN_SPEED
        else:
            speed_limit = MAX_SPEED
        throttle = 1.0 - steering_angle**2 - (speed/speed_limit)**2
        #print(str(steering_angle) + "     " + str(throttle))

        send_control(steering_angle, throttle)
    else:
        sio.emit('manual', data={}, skip_sid=True)
```

Chapter 7
Conclusion

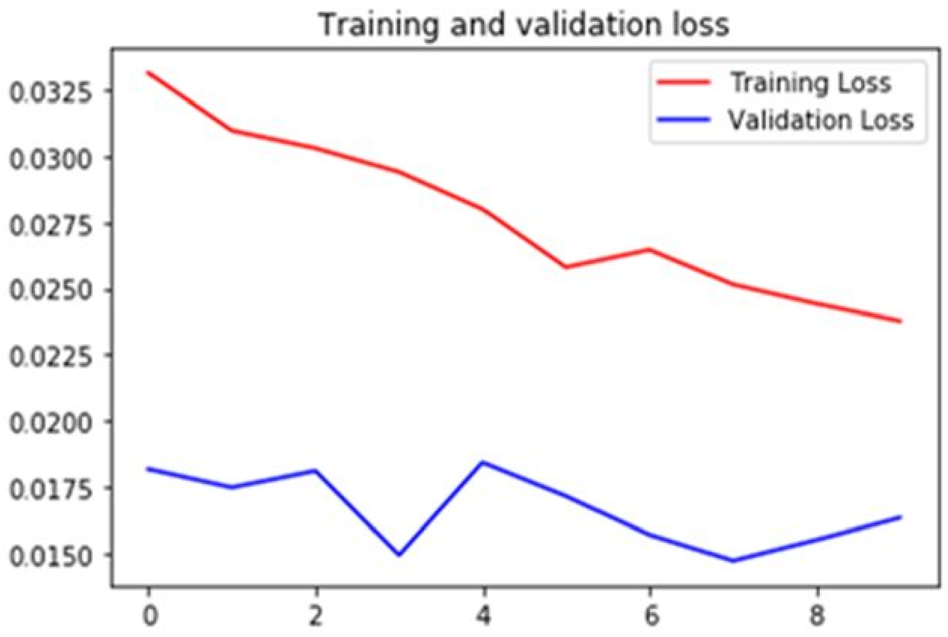


Fig 15: Training And Validation Loss

This project started with training the models and tweaking parameters to get the best performance on the tracks and then trying to generalize the same performance on different tracks.The models that performed best on 1 track did poorly on Track_2, hence there was a need to use image augmentation and processing to achieve real time generalization. Substituting recurrent layers for pooling layers might reduce the loss of information and wouldbe worth exploring in the future projects.It is interesting to find the use of combinations of real world datasets and simulator data to train these models. Then I can get the true nature of how a model can be trained in the simulator and generalized to the real world or vice versa.There are many experimental implementations carried out in the field of self-driving cars and this project contributes towards a significant part of it.

We have empirically demonstrated that CNNs are able to learn the entire task of lane and road following without manual decomposition into road or lane marking detection, semantic abstraction,path planning, and control. A small amount of training data from less than a hundred hours of driving was sufficient to train the car to operate in diverse conditions, on highways, local and residential roads in sunny, cloudy, and rainy conditions. The CNN is able to learn meaningful road features from a very sparse training signal (steering alone).The system learns for example to detect the outline of a road without the need of explicit labels during training.More work is needed to improve the robustness of the network, to find methods to verify the robust-ness, and to improve visualization of the network-internal processing steps.

References

1. Oliver Cameron, “Challenge #2: Using Deep Learning to Predict SteeringAngles”, Published on 11 Aug 2016, https://medium.com/udacity/challenge-2-using-deep-learning-to-predict-steering-angles-f42004a36ff3, accessed Jul 2017

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

3. Andrej Karpathy, “The unreasonable effectiveness of Recurrent Neural Networks”, Published on 21 May,2015, http://karpathy.github.io/2015/05/21/rnn-effectiveness , 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. Jason Brownlee, “How to use TimeDistributed Layers for LSTM”, https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/, accessed Nov 2017

6. Lucas Weist,“Recurrent Neural Networks -Combination of RNN and CNN”, Published on 7 Feb 2017, https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+Combination+of+RNN+and+CNN ,accessed Nov 2017

7. Dmytro Nasyrov, “Behavioral Cloning. NVidia Neural Network in Action.” , Published on 21 Aug 2017, https://towardsdatascience.com/behavioral-cloning-project-3-6b7163d2e8f9 ,accessed Jan2017

8. Sihan Li, “Demystifying ResNet”,Published on 20May2017, https://arxiv.org/abs/1611.01186 , accessed Jan2017

9. Franchois Chollet, “Building powerful image classification models using very little data”, Published on 5June2016, https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html , accessed Jan201710.Ivan Kazakov, “Vehicle Detection andTracking”, Published on 14 May2017,https://towardsdatascience.com/vehicle-detection-and-tracking-44b851d70508 , accessed Feb 2017

Acknowledgements

We take the opportunity to thank all those people who have helped and guided us through this project and make this experience worthwhile for us. We wish to sincerely thank our reverend **Bro. Jose Thuruthiyil** and principal **Dr. Sincy George** for giving us this opportunity for making a project in the Final Year of Engineering. We would also like to thank HOD of the Computer department **Dr. Kavita Sonawane** and all teaching and non-teaching staff for their immense support and cooperation.

Last but not the least we would like to thank **Mr. Rupesh Mishra** for guiding us throughout the project and encouraging us to explore this domain.