IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 6, May 2023

Self-Driving Car Trained using Udacity's Simulator and Deep Neural Network

Pradyumna Charate[1], Rutuja Chavan[2], Sarvesh Chavhan[3], Harshal Deokar[4], Prof. Rahul Dagade[5]
Student, Department of Computer Engineering[1,2,3,4]
Professor, Department Of Computer Engineering[5]
Smt. Kashibai Navale College of Engineering, Pune, Maharashtra, India

Abstract: *Self-driving cars has become a trending subject with a significant improvement inthe technologies in the last decade. The project purpose is to train a neural network to drivean autonomous car agent on the tracks of Udacity's Car Simulator environment. Udacity has released the simulator as an open source software and enthusiasts have hosted a competition (challenge) to teach a car how to drive using only 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 behaviorin the training mode on the track. That means a dataset is generated in the simulator by user driven car in training mode, and the deep neural network model then drives the car in autonomous mode. Three architectures are compared with respect to their performance. 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. Thedataset for Track_1, which was simple with favorable road conditions to drive, was used as the training set to drive the car autonomously on Track_2 which consists of sharp turns, barriers, elevations and shadows. To tackle this problem, image processing and different 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 generalizingwell. The project aims at reaching the same accuracy on real time data in the future.*

Keywords: Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM), Time Distributed Layer.

I. INTRODUCTION

The purpose of a Self-driving car project is to build a better autonomous driver. The car should be able to drive itself without falling off the track, with accelerating and braking at appropriate places. Udacity released an open source simulator for self-driving cars to depict a real-time environment. The challenge is to mimic the driving behavior of a human on the simulator with the help of a model trained by deep neural networks. The concept is called Behavioral Cloning, to mimic how a human drives. The simulator contains two tracks and two modes, namely, training mode and autonomous mode. The dataset is generated from the simulator by the user, driving the carin training mode. This dataset is also known as the "good" driving data. This is followed by testing on the track, seeing how the deep learning model performs after being trained by that user data. Another challenge is to generalize the performance on different tracks. That means, training the model using the dataset created on one of the tracks, and testing it on the other track of the simulator.

II. MOTIVATION

We are a long way away from having a true self-driving car. By a true self-driving car we mean a car that can be essentially driven in any manner as a human driving a car. This is an incredibly hard thing to achieve. Major automobile companies have been trying to achieve true autonomous driving. Main motivations behind the idea are:

- Safer Roads
- Increase in productivity
- More economical

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 6, May 2023

- The movement will be more efficient
- More environment friendly
- Reduce accident rate

III. OBJECTIVES

The objective of the Self-Driving Car is to decide on the acceleration to apply along a given path. Two different scenarios are considered: right-turn and left-turn Source publication. Along with the turn Steering angle through which vehicle will turn is very important and to be determined by deep learning model.

IV. LITERATURE SURVEY

In a new automotive application, the authors Mariusz Bojarski, Ben Firner, Beat Flepp, Larry Jackel, Urs Muller, Karol Zieba and Davide Del Testa in paper 'End-to-End Deep Learning for Self-Driving Cars' have used convolutional neural networks (CNNs) to map the raw pixels from a front-facing camera to the steering commands for a self-driving car. This powerful end-to-end approach means that with minimum training data from humans, the system learns to steer, with or without lane markings, on both local roads and highways. The system can also operate in areas with unclear visual guidance such as parking lots or unpaved roads. They designed the end-to-end learning system using an NVIDIA DevBox running Torch 7 for training. An NVIDIA DRIVETM PX self-driving car computer, also with Torch 7, was used to determine where to drive—while operating at 30 frames per second (FPS). The system is trained to automatically learn the internal representations of necessary processing steps, such as detecting useful road features, with only the human steering angle as the training signal. According to authors, never explicitly trained it to detect, for example, the outline of roads. In contrast to methods using explicit decomposition of the problem, such as lane marking detection, path planning, and control, our end-to-end system optimizes all processing steps simultaneously. They believe that end-to-end learning leads to better performance and smaller systems. Better performance results because the internal components self-optimize to maximize overall system performance, instead of optimizing human-selected intermediate criteria, e. g., lane detection. Such criteria understandably are selected for ease of human interpretation which doesn't automatically guarantee maximum system performance. Smaller networks are possible because the system learns to solve the problem with the minimal number of processing steps.

V. METHODOLOGY

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

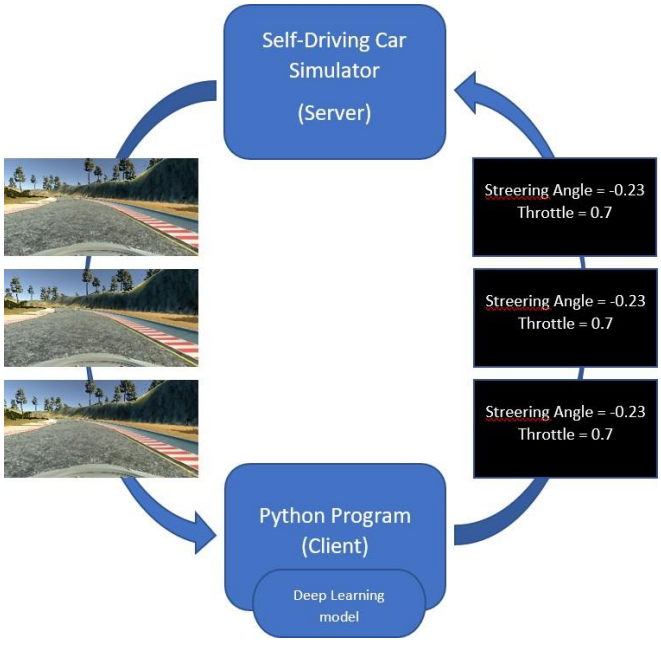

Fig. Implementation Architecture

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 6, May 2023

The simulator can be used to collect data by driving the car in the training mode using a joystick or keyboard, providing the so called "good-driving" behavior input data in form of a driving_log (.csv file) and a set of images. The simulator acts as a server and pipes these images and data log to the Python client. The client (Python program) is the machine learning model built using Deep Neural Networks. These models are developed on Keras (a high-level API over Tensorflow). Keras provides sequential models to build a linear stack of network layers. Such models are used in the project to train over the datasets as the second step. Detailed description of CNN models experimented 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.

VI. SYSTEM ARCHITECTURE

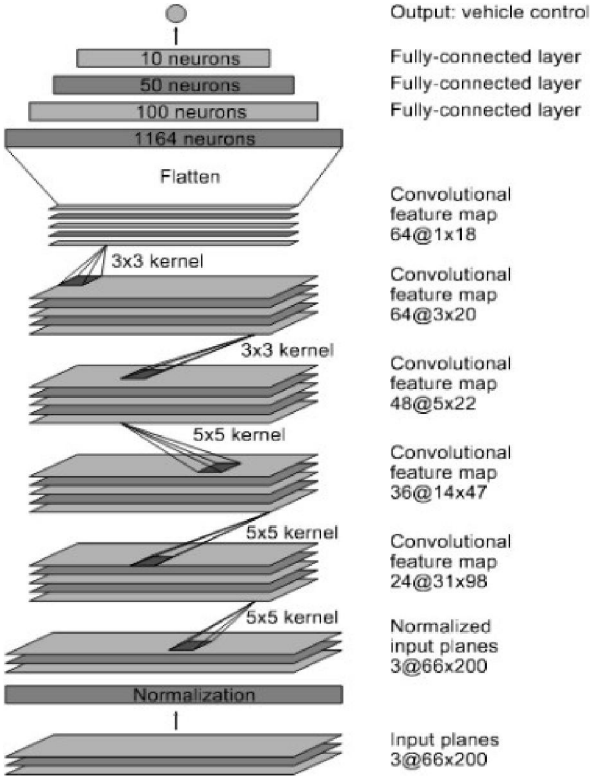

Fig. System Architecture

VII. RESULTS

```
None
Epoch 1/10
300/300 [==============================] - 448s 1s/step - loss: 0.1158 - val_loss: 0.0566
Epoch 2/10
300/300 [==============================] - 513s 2s/step - loss: 0.0710 - val_loss: 0.0470
Epoch 3/10
300/300 [==============================] - 463s 2s/step - loss: 0.0567 - val_loss: 0.0377
Epoch 4/10
300/300 [==============================] - 508s 2s/step - loss: 0.0492 - val_loss: 0.0346
Epoch 5/10
300/300 [==============================] - 507s 2s/step - loss: 0.0477 - val_loss: 0.0327
Epoch 6/10
300/300 [==============================] - 448s 1s/step - loss: 0.0458 - val_loss: 0.0310
Epoch 7/10
300/300 [==============================] - 453s 2s/step - loss: 0.0430 - val_loss: 0.0298
Epoch 8/10
300/300 [==============================] - 452s 2s/step - loss: 0.0424 - val_loss: 0.0288
Epoch 9/10
300/300 [==============================] - 511s 2s/step - loss: 0.0397 - val_loss: 0.0278
Epoch 10/10
300/300 [==============================] - 518s 2s/step - loss: 0.0387 - val_loss: 0.0297
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 6, May 2023

```
Epoch 20/30
33/33 [==============================] - 1s 26ms/step - loss: 0.1691 - val_loss: 0.1585
Epoch 21/30
33/33 [==============================] - 1s 24ms/step - loss: 0.1672 - val_loss: 0.1572
Epoch 22/30
33/33 [==============================] - 1s 25ms/step - loss: 0.1675 - val_loss: 0.1572
Epoch 23/30
33/33 [==============================] - 1s 29ms/step - loss: 0.1688 - val_loss: 0.1582
Epoch 24/30
33/33 [==============================] - 1s 29ms/step - loss: 0.1664 - val_loss: 0.1579
Epoch 25/30
33/33 [==============================] - 1s 25ms/step - loss: 0.1690 - val_loss: 0.1581
Epoch 26/30
33/33 [==============================] - 1s 26ms/step - loss: 0.1662 - val_loss: 0.1574
Epoch 27/30
33/33 [==============================] - 1s 26ms/step - loss: 0.1680 - val_loss: 0.1574
Epoch 28/30
33/33 [==============================] - 1s 26ms/step - loss: 0.1658 - val_loss: 0.1573
Epoch 29/30
33/33 [==============================] - 1s 24ms/step - loss: 0.1666 - val_loss: 0.1574
Epoch 30/30
33/33 [==============================] - 1s 26ms/step - loss: 0.1670 - val_loss: 0.1576
```

Loss

Fig. Loss Over Epochs

Fig. Track Screenshot

VIII. CONCLUSION

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. The use of CNN for getting the spatial features and RNN for the temporal features in the image dataset makes this combination a great fit for building fast and lesser computation required neural networks. Substituting recurrent layers for pooling layers might reduce the loss of information and would be worth exploring in the future projects. It is interesting to find the use of combinations of real world dataset 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

REFERENCES

[1] Yang Zhao, “End-to-End Driving Model for Steering Control of Autonomous Vehicles with Future Spatiotemporal Features”, Published on 08 Nov 2019.
[2] Dmytro Nasyrov, “Behavioral Cloning. NVidia Neural Network in Action.”, Published on 21 Aug 2017, accessed Jan 2017.
[3] Sihan Li, “Demystifying ResNet”, Published on 20 May 2017, accessed Jan 2017.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 6, May 2023

[4] Franchois Chollet, "Building powerful image classification models using very little data", Published on 5 June 2016, accessed Jan 2017.
[5] Ivan Kazakov, "Vehicle Detection and Tracking", Published on 14 May 2017, accessed Feb 2017.
[6] Liu W. et al. (2016) SSD: Single Shot MultiBox Detector. In: Leibe B., Matas J., Sebe N., Welling M.(eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905.Springer, Cham
[7] Jonathan Hui, "Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3" [Online]. Available: [Online]. Available: "https://medium.com/@jonathan_hui/realtimeobject-detection-with-yolo-yolov2-28b1b93e2088"