Drive to Survive

Sai Varun Reddy, Mullangi
Ira A. Fulton Schools of Engineering
Arizona State University
Tempe, AZ USA
smullan2@asu.edu

ABSTRACT

We are seeing how self-driving cars are taking over the market and have the potential to replace all human drivers in the near future. In this project, I have trained multiple machine learning model which predicts steering angle and throttle of a car in an Udacity simulated environment. The inputs to the model are images from the camera mounted on the car. I will compare the performance of the multiple models by looking at the loss and convergence rate. I will also evaluate the models based on how well models' outputs are able to autonomously drive in the car in the simulated environment and further deep dive on how CNNs are interpreting the images.

KEYWORDS

Convolutional Neural Networks, End to End Learning, Deep Learning

1 Introduction

The first idea for a self-driving car was designed by Leonardo da Vinci in the 1500s, using high-tension springs. Today, the concept of an autonomous car has come a long way, thanks to advances in technology.

Until early 2000's self-driving cars research was majorly around using sensors like LiDAR, radar, ultrasonic and GNSS; After the CNN revolution [1], researchers started to realize the efficiency of CNNs in pattern recognition and usage of vision in autonomous driving. There are 6 levels of driving automation (Figure 1) and to achieve last level, many companies like Waymo, Tesla and Cruise are spending billions of dollars and they are doubling down on vision. We can see many cars from the above mentioned companies collecting data with 3d cameras mounted on them in day to day life.

In this project, I will try to investigate how self-driving cars can make an informed decision (steering angle, and throttle) based on vision. Idea is to implement end to end learning for self-driving car.

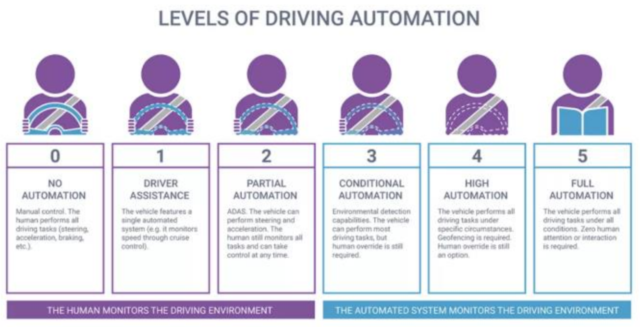
LEVELS OF DRIVING AUTOMATION
0
NO AUTOMATION
Manual control. The human performs all driving tasks (steering, acceleration, braking, etc.).
1
DRIVER ASSISTANCE
The vehicle features a single automated system (e.g. it monitors speed through cruise control).
2
PARTIAL AUTOMATION
ADAS. The vehicle can perform steering and acceleration. The human still monitors all tasks and can take control at any time.
3
CONDITIONAL AUTOMATION
Environmental detection capabilities. The vehicle can perform most driving tasks, but human override is still required.
4
HIGH AUTOMATION
The vehicle performs all driving tasks under specific circumstances. Geofencing is required. Human override is still an option.
5
FULL AUTOMATION
The vehicle performs all driving tasks under all conditions. Zero human attention or interaction is required.
THE HUMAN MONITORS THE DRIVING ENVIRONMENT
THE AUTOMATED SYSTEM MONITORS THE DRIVING ENVIRONMENT

Figure 1: 6 levels of driver automation. source

2 Related Work

Adri et al [2] provided a survey of end to end learning techniques for autonomous driving. In the paper they discussed various learning techniques like Imitation learning, Reinforcement Learning and Transfer from simulation to real word.

This project is inspired from the work of M. Bojarski et al [3] at Nvidia, where they have developed a convolutional neural network (CNN) model to convert the unprocessed images from a single front-facing camera into steering commands (Figure 2).

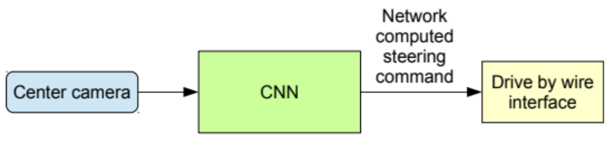

Figure 2: Self Driving car configuration. source

The simulation environment for the evaluation of the self-driving car and the python client setup were taken from the Udacity self-driving car simulation GitHub [4]. Figure 3 displays the basic structure of the environment.

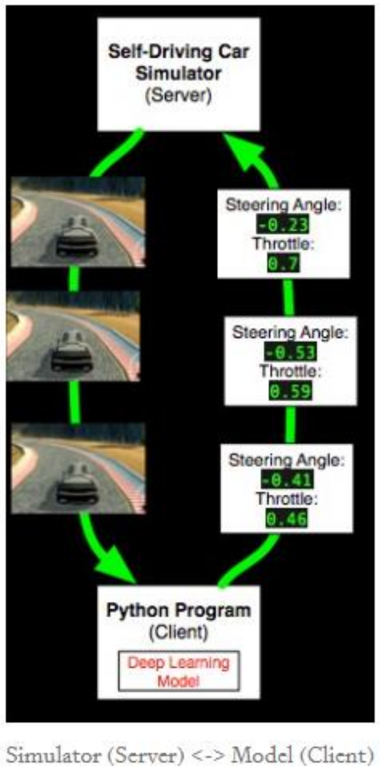
Self-Driving Car Simulator (Server)
Steering Angle: -0.23 Throttle: 0.7
Steering Angle: -0.53 Throttle: 0.59
Steering Angle: -0.41 Throttle: 0.46
Python Program (Client)
Deep Learning Model
Simulator (Server) <-> Model (Client)

Figure 3: Simulation environment structure. source

3 Data

Datasets for training the deep learning model are collected from 2 different data sources. First dataset is collection of car dash cam of images from the streets of California [5]. Second dataset is the Kaggle dataset [6] of simulated images from the same simulation environment. In the Udacity self-driving environment we have two options, training mode and autonomous mode. We can drive the car in training mode and at every frame left, right and center images are generated along with a csv of steering command and throttle.

	GitHub [4]	Kaggle [5]
Dataset Size	63000	3400

3.1 Preprocessing

Images are initially normalized before sending them to the model for training. ImageNet Mean and standard deviation values are used for normalization. I am using only center image for predicting the steering and throttle. Before sending the image to model, image is trimmed such that the front part of the car and sky will be removed. This is important in convergence as we are sending only the relevant information.

4 Model Architecture

I have implemented the CNN architecture followed by the Nvidia group [3] for end to end self-driving. The input images are first normalized and then series of 3 CNN layers with kernel size 5 and then a series of 2 CNN layers with kernel size 3 are applied. After this, flatten our image and apply 4 fully connected layers in series. RELU is the activation layer used between layers. See figure 4 for architecture details

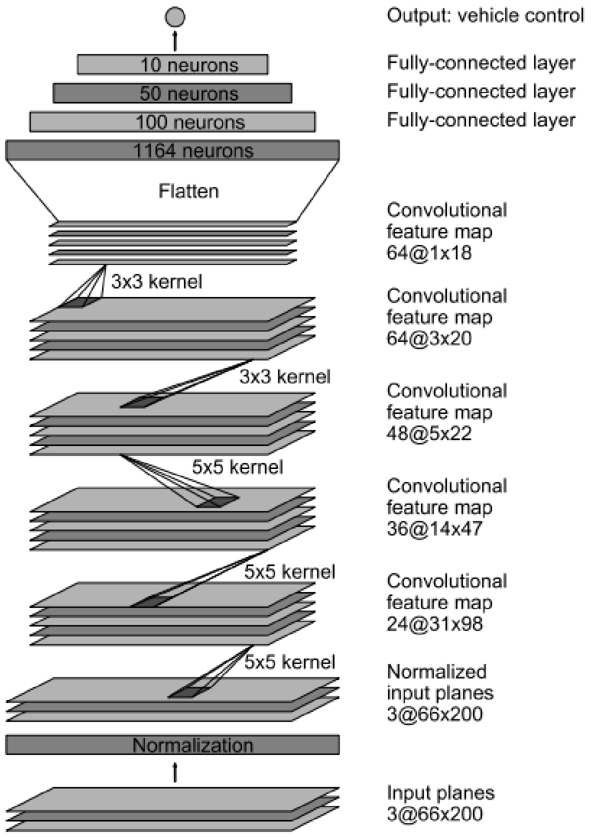

Figure 4: CNN architecture with 250 thousand trainable parameters. source

In the Nvidia CNN architecture presented above we can only predict steering angle and just cap the throttle so that the car does not move out of the track. To use both training angle and throttle, I have used Multitask learning approach. In this architecture, there is a shared CNN layers for steering angle and throttle to learn the common low level features and then the individual layers learn the task specific features. The loss is back propagated individually for steering angle and throttle. Refer fig 5 for the detailed view of the architecture.

I have experimented with both the architectures on both simulated and real world images datasets

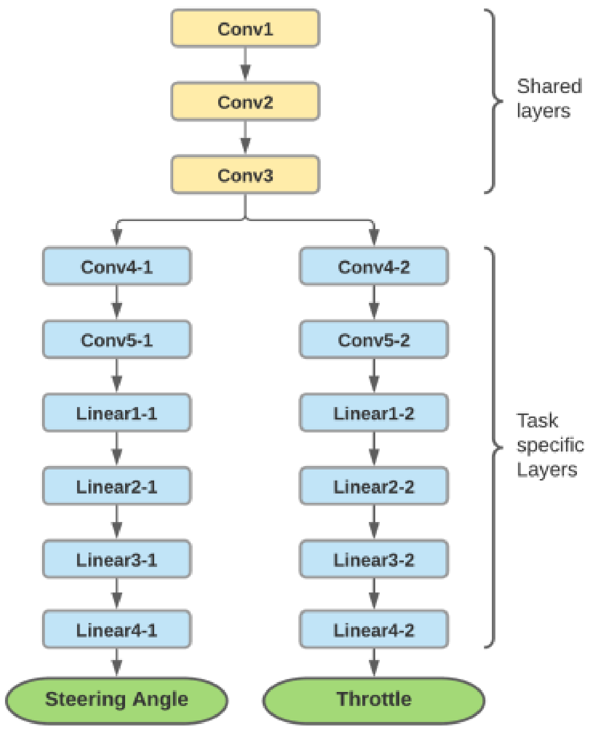

Figure 5: Multitask learning architecture

5 Training

For the CNN Nvidia architecture (figure 4), Mean squared error loss between actual steering angle and predicted steering angle is used to propagate loss backward. Adam is used as the optimizer. I trained the model for 20 epochs and can see the training and validation loss decrease at constant rate as shown in the fig.6.

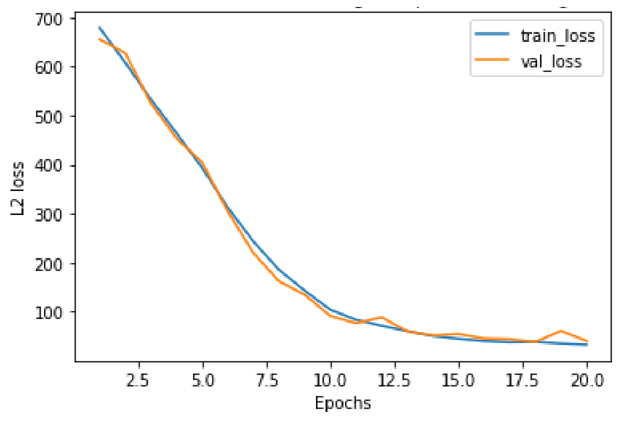

Figure 6: MSE Loss of training and validation for the real word dataset using Nvidia CNN architecture.

Further, I have trained Nvidia CNN architecture on Kaggle/Simulated dataset to match the training and testing environment. The model converges in just 10 epochs as we can see in the fig 7.

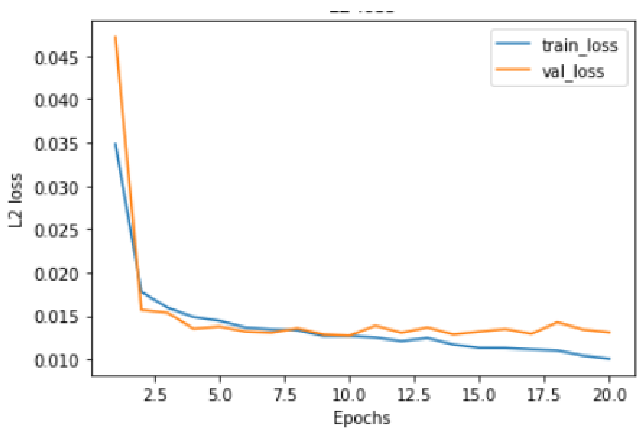

Figure 7: MSE Loss of training and validation for the Kaggle dataset using Nvidia CNN architecture.

I have used Kaggle/Simulated data for training on multitask learning architecture. Only the center camera images were used, resulting in a dataset of 3400 images. During the training process, a problem was encountered whereby the training and validation losses stagnated after a single epoch. This could be an indication that the dataset size used is too small. Further investigation into the dataset size is necessary in order to determine the optimal size for training.

Training and validation comparison among different implementations.

Architecture	Dataset	Epochs	Training loss	Validation loss
Nvidia CNN	Real Life Images	20	32.66	40.19
Nvidia CNN	Kaggle	10	0.012	0.013
Multitask learning	Kaggle	10	2.3e-04	2.6e-04

5 Results

I have used Udacity simulation environment to test the Nvidia CNN model. In the first experiment, as the dataset I have used to train is dashcam images of real word, it did not perform well on the simulated environment. For most of the input images the models output is to turn steering towards left this makes the car in the simulated environment to go outside the track as shown in the fig.8.

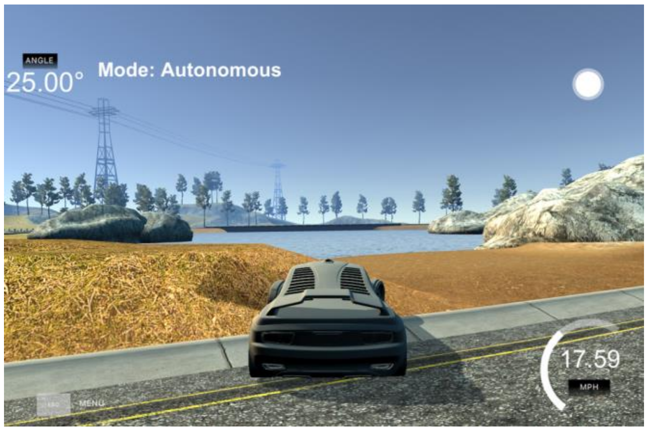

Figure 8: Car in autonomous mode after taking input from the CNN model trained on real word dataset

This behavior is corrected when I have used the Nvidia CNN model trained on Kaggle/Simulated dataset. I found that the model was successful in controlling the car and allowing it to move within the track boundaries. The model is able to identify turns accurately, however a left bias is observed in the steering angles of the car. This bias is likely due to the fact that the simulated track was circular, and the car was moving in an anticlockwise direction, resulting in the majority of the steering angles being towards the left side.

Multitask learning model is also tested in order to assess its ability to accurately identify curves while controlling the speed of a car. The results of the experiment showed that the model was able to accurately identify the curves but unable to control the speed during the curves, resulting in the car overshooting and getting out of the track.

6 CNN layers Interpretability

Figure 9 depicts a comparison between the original images and the outputs of the 5th channel of the 1st convolution layer of the Nvidia CNN model, which was trained on a real-world dataset. The results demonstrate that the convolutional neural network is attempting to differentiate between road and grass. The outputs of the convolution layer are activated using the Rectified Linear Unit (ReLU) activation function. The results show that the model is able to successfully identify the road and grass in the images.

The similar analysis is done on the simulated images as well. In the figure 10 we can see the comparison between input image and the 23rd channel of the 1st convolution layer of the Nvidia CNN model.

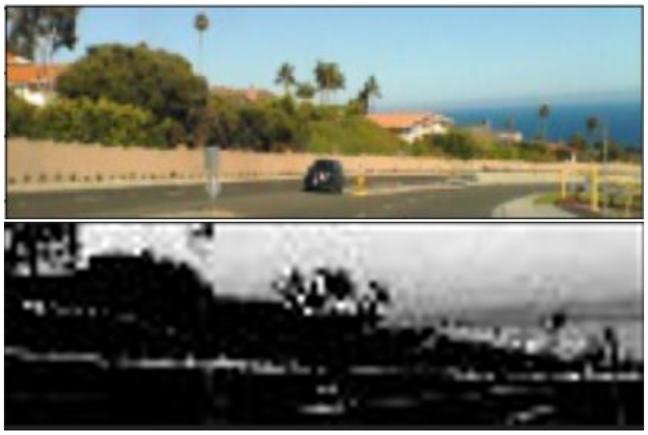

Figure 7: Input image vs 1st CNN layer 5th channel output

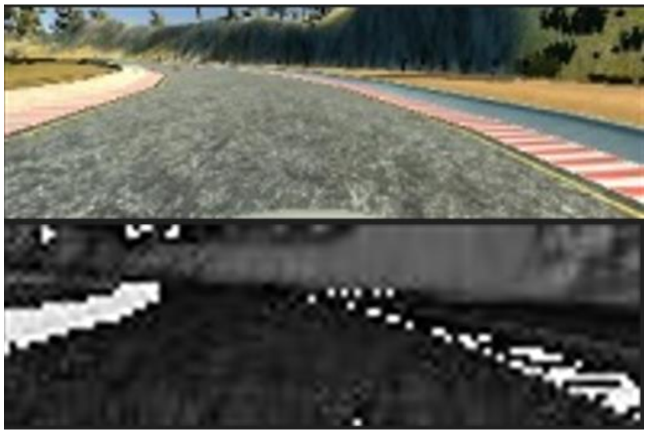

Figure 10: Input image vs 1st CNN layer 23rd channel output

7 Discussion and Future work

The Nvidia CNN model trained on Kaggle dataset performed the best in the simulated environment. In the section 6, we have seen that Convolutional Neural Networks (CNNs) are a powerful tool for computer vision applications. This is demonstrated by their capacity to learn and capture detailed features with a single convolution layer. Further layers of a CNN then attend to task-specific features, showing the importance of multiple layers in a CNN. This has been used to great success in many computer vision tasks, such as image recognition and object detection.

We also must think about the downsides of using vision in the autonomous driving cars, in many of the cases computer vision models are not robust enough to detect all the signs if we add any kind of adverse data, fig 11 is one of such examples. We need to explore more ways to make the models robust.

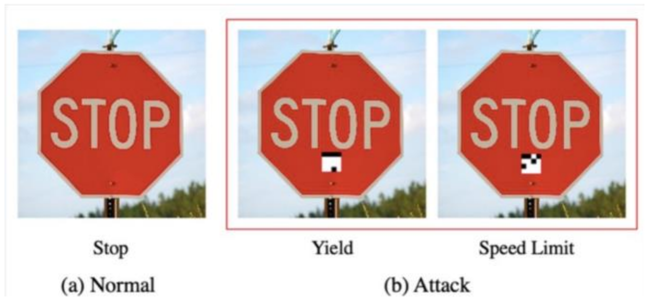

Figure 9: Automotive ML systems have been tricked into mistaking stop signs for speed limit signs

There are many parts of this projects that are yet to be explored like how to use three images (center, left and right) for training, how to control the throttle, how to make the models trained on real world work in simulated environments (transfer learning).

8 Conclusion

I have achieved the goal to autonomously drive a vehicle based on steering angles and throttle provided my machine learning model. I have performed 3 experiments, two of them are on Nvidia CNN architecture and one on Multitask learning architecture.

The model trained on only steering angle performed best in the simulation environment. I have also analyzed the training data and found that the dataset is skewed towards left turns and the track is counterclockwise. I have further shown that CNNs are a powerful tool in the computer vision tasks.

ACKNOWLEDGMENTS

I would like to express my gratitude to Prof. Hannah Kerner and Teaching Assistant Krishna Sree G for continuous guidance and support throughout this project.

REFERENCES

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551, Winter 1989. URL: http://yann.lecun.org/exdb/publis/pdf/lecun-89e.pdf.
[2] Ardi Tampuu. 2020. A Survey of End-to-End Driving: Architectures and Training Methods. arXiv:2003.06404. Retrieved from https://arxiv.org/abs/2003.06404
[3] Mariusz Bojarski. 2016. End to End Learning for Self-Driving Cars. arXiv:1604.07316. Retrieved from https://arxiv.org/abs/1604.07316
[4] Aaron Brown, self-driving-car-sim, (2018), GitHub repository, https://github.com/udacity/self-driving-car-sim
[5] Sully Chen, driving-datasets, (2018), GitHub repository, https://github.com/SullyChen/driving-datasets
[6] Andy8744, Udacity Self Driving Car - Behavioural Cloning, (2022), Kaggle, https://www.kaggle.com/datasets/andy8744/udacity-self-driving-car-behavioural-cloning