DALHOUSIE UNIVERSITY

Halifax, Nova Scotia, Canada

CSCI 6505: Machine Learning

Project Proposal

Instructor
Prof. Sageev Oore

Submitted By:

Mukund Sharma	B00893013
Mayank Sareen	B00899565
Mihir Sanchaniya	B00814744
Sidharth Mahant	B00899439

Introduction

It is no secret that self-driving cars are among the hottest areas of research and business for tech giants. The concept which seemed like science-fiction only a few years ago is now quickly becoming a reality. Despite the fact that companies such as Tesla, Nissan, BMW, and Cadillac have self-driving car assistance software, they still require a human to take control on occasion and keep an eye on the road. Nevertheless, it is amazing to see how far we have come in terms of innovation and how rapidly technology is developing. Therefore, we can now, with the help of deep learning and neural networks, construct our own pipeline for autonomous vehicles.

To accomplish this, we plan on creating a Convolutional Neural Network (CNN) based on the famous research paper from Nvidia (End-to-End Learning for Self-Driving Cars) [1] and replicating a smaller version of this architecture and comparing the performance of the model with different layer architectures.

Model Architecture

For any deep learning project to succeed, the model architecture needs to be highly efficient. Considering that the Nvidia model works so well for behavior cloning and for self-driving cars, our goal is to see if we can replicate or find a smaller version that we can use with our limited data and computing power. A total of 9 layers are present in the network. These include a normalization layer, 5 convolutional layers, and 3 fully connected layers. The image input is divided into YUV planes and passed to the network.

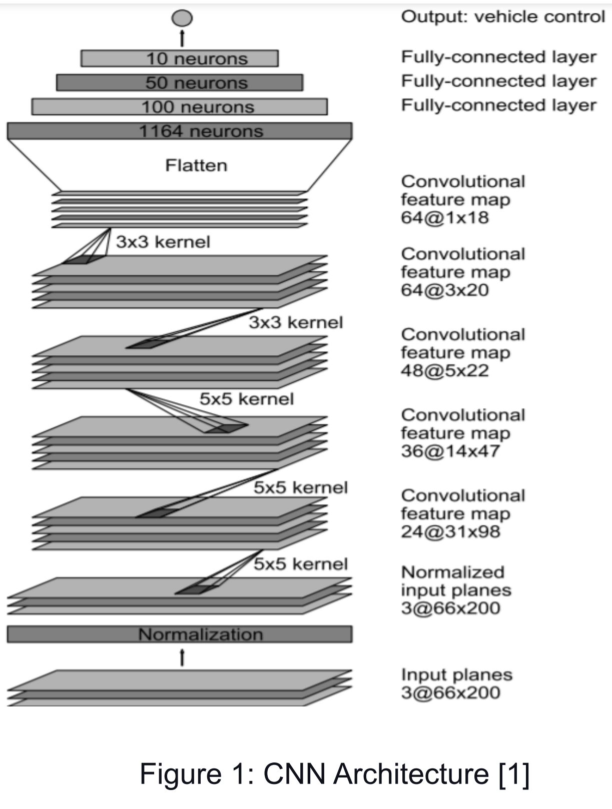

Figure 1: CNN Architecture [1]

Currently, we are planning on reducing the number of convolutional layers and linear layers and proceeding as the network performs. Using Keras to build this neural network will be pretty straightforward, but we will be using PyTorch to better understand the framework and how it processes data.

Gathering Data

Although the Udacity dataset is useful, it is not enough to make the car run on different tracks. Udacity simulator [2] comes with an option of training mode, using which we can easily generate a sufficient amount of data to train our model. It creates a directory of images and a CSV file with attributes like steering angle, throttle, left, right, brake, speed, etc. We also found a dataset for this on Kaggle but we are planning to simulate on our own, in which we will try to keep the car between the lines even during the turns, which will eventually make a good model.

Training the Neural Network:

Images are input into a CNN, which calculates a steering command. The proposed command for the image is compared to the desired command, and the CNN weights are changed to get the CNN output closer to the desired output, to make the change in weights back propagation is used.

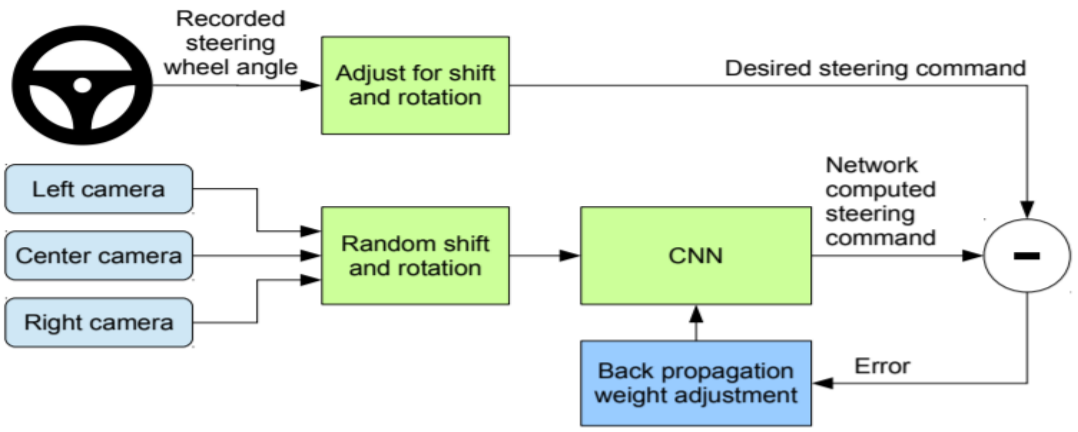
Recorded steering wheel angle
Adjust for shift and rotation
Desired steering command
Left camera
Center camera
Right camera
Random shift and rotation
CNN
Network computed steering command
–
Back propagation weight adjustment
Error

Figure 2: Training the neural network[1]

The trained network will be able to generate steering commands from the video images of a single-center camera.

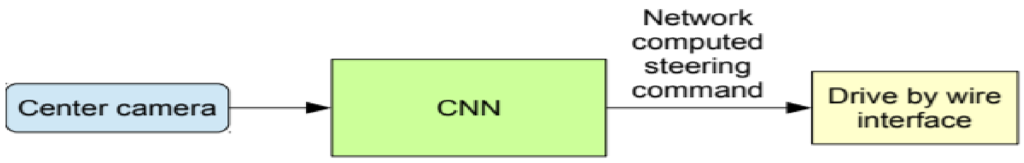

Figure 3: Trained network [1]

Simulation Test:

To test the feasibility of our model, the Udacity simulator comes with an autonomous mode, we need to have a python code that will load our model and read frames of the track and provide it to our model and send the predicted parameters to the simulator.

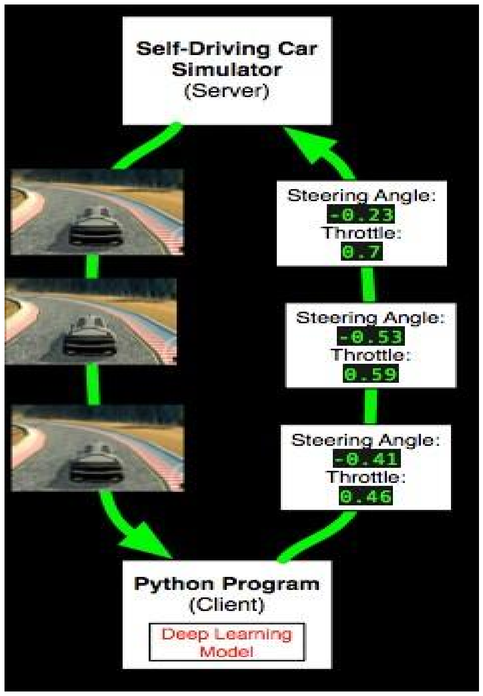
Self-Driving Car Simulator (Server)
Steering Angle: -0.23 Throttle: 0.7
Steering Angle: -0.53 Throttle: 0.59
Steering Angle: -0.41 Throttle: 0.46
Python Program (Client)
Deep Learning Model

Figure 4: Testing the model [3]

Learning involved

Although to implement this CNN model, we will be referring to the Nvidia research paper and a few GitHub repositories, our goal is to take maximum learning from this project, exploring OpenCV and understand what it is capable of, using Pytorch to implement CNN that predicts steering angle from images, how to do hyperparameter optimization, training the model on a GPU machine and how to efficiently use CUDA, how the result differ if we train the same convolutions using Keras, creating data loader to fit our Pytorch model, understanding how to comprehend research papers. Try similar and smaller convolutional networks and compare their results.

Timeline

Week 1: Gather a sufficient amount of data using the Udacity simulator and create a data loader. Understand how a Convolutional neural network works and try creating a simple CNN using Pytorch and Keras to understand the frameworks.

Week 2: Explore OpenCV and understand feature extraction like lane detection, sign detection, grayscale conversion, edge detection, region of interest, etc. Try to replicate the given Nvidia CNN architecture and observe how it behaves on the provided dataset.

Exit Point: At this point, we will be having an initial self-driving model that will detect lines in an image frame. We will try to add a maxpool layer to the architecture or gather more data to further increase the accuracy of the model and make sure the car moves between the lines at least on a straight track.

Week 3: Try to build smaller CNN models that can better perform on the dataset and evaluate their test results. Try to capture the performance of those CNNs with PyTorch and Keras.

Exit Point: After the third week, we will be having a smaller version of the architecture which will work better with a limited dataset and the car performs well even during the turns. A decent amount of comparison will be made between different approaches.

Week 4: Fine-tune the model which best fits the dataset, perform hyperparameter optimization, and make a presentable demo.

Training Resources:

We have gone through some Udemy courses and tried training some large datasets on google colab and managed to get pretty decent accuracy. We also have 2 Nvidia 1060ti GPU machines which we will be using extensively throughout this project.

References

[1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars." *Nvidia* [Online]. Available: https://arxiv.org/pdf/1604.07316v1.pdf. [Accessed: Feb.17, 2022]
[2] "Dataset Simulator." *Github* [Online]. Available: https://github.com/udacity/self-driving-car-sim. [Accessed: Feb.17, 2022]
[3] "Car-behavioral-cloning-with-pytorch." *Github* [Online]. Available: https://github.com/hminle/car-behavioral-cloning-with-pytorch . [Accessed: Feb.17, 2022]