Machine Learning Engineer Nanodegree

Capstone Proposal

End to end deeplearning for autonomous driving

Domain Background

With the interests of artificial intelligent raising, the self-driving vehicle for transportation industry is becoming a really high potential application as a smart hardware implementation of artificial intelligent. The most critical challenge to develop self-driving vehicle is safety. While discussing the safety, the first step is sensing, to know the status of the vehicle from both inside and outside. Redundancy is also necessary to maintain the system functional while part of sensing system crashed. Meanwhile, we need notice that redundancy is not only a copy, it is best to be a backup that can solve the problem with a different method.

Given this concept, end-to-end deep learning based self-driving solution is promoted. Comparing to the traditional sensor fusion and control theory, End-to-end solution implements the same functions using its independent theory. It shows the market a new way to build a self-driving vehicle with its impressing performance and simple architecture. In 2016, NVIDIA presents a remarkable prototype of end-to-end learning for self-driving cars. In 2017, Baidu launches Apollo project with open source solution and end-to-end data.

NVIDIA End-to-End Deep Learning for Self-Driving Cars: https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/ (https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/)

Baidu Apollo Project: http://apollo.auto/ (http://apollo.auto/)

Problem Statement

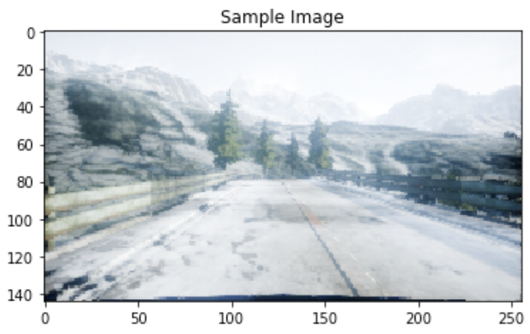
The topic of this project is to use end-to-end deeplearning model for autonomous driving, which is to replicate the result from NAVIDA self-driving project that teaching car how to drive using only cameras and deep learning. A system like this will have camera images, collected from a windshield camera of a car, as input and predict a direct steering command (steering angle) in real-time. Data collected from the AirSim simulation (Microsoft open sourced Aerial Informatics and Robotics Platform) will be used to train and test the model. The performance can also be evaluated by comparing with other published End-to-End Autonomous Steering Model, like commaai.com (Learning a Driving Simulator).

Datasets and Inputs

Datasets are Obtained form AirSim photo-realistic simulator(https://www.microsoft.com/en-us/research/project/aerial-informatics-robotics-platform/ (https://www.microsoft.com/en-us/research/project/aerial-informatics-robotics-platform/)). The provided simulator includes solid information to give users a full picture of the driving situation of the car. It helps the developers collect a large amount of data to train the autonomous driving model without having to use an actual car.

When people learn driving, we will be trained to drive in different scenarios, like driving on different kinds of roads, different lighting conditions, different traffics and so on. So we will have experience to handel complex conditions when we on the road. It is the same to train a deeplearning based self-driving system. Providing the system sufficient data is the foundation to generate a reliable performance.

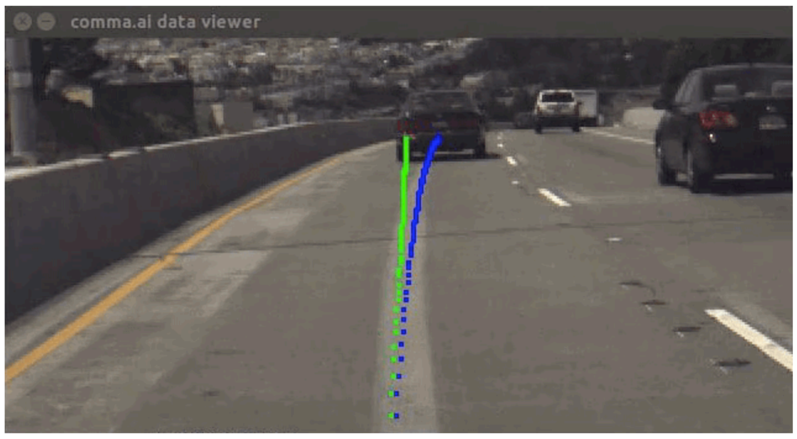
The AirSim generated dataset will have two parts, which are images from front camera and .tsv files.
An example of the image:



.tsv file consists information of Timestamp, throttle, brake, speed, steering and ImageName.

	Timestamp	Speed (kmph)	Throttle	Steering	Brake	Gear	ImageName	Folder
0	93683464	0	0.0	0.000000	0.0	N	img_0.png	data_raw/normal_1
1	93689595	0	0.0	0.000000	0.0	N	img_1.png	data_raw/normal_1

Another optional dataset is recorded from real driving published by commaai.com:
http://research.comma.ai (http://research.comma.ai)



Solution Statement

The goal of the project is to train a deeplearning model that can make steering control for a vehicle base on facing camera input and possible previous vehicle states. Considering the type of the input as the image, Convolutional Neural Networks was considered as a good option to start given its abilities on visual applications. Different architectures of CNN will be evaluated and tuned for the use case. The deeplearning model will be designed to identify the features of camera images, recognize them and build the link to the expected result - steering angles. AirSim provides the project a straightforward way to generate the training data with a variety of use cases and a safe test bed to evaluate the deeplearning model. The system will be measured using RMS error between the predicted steering angle and the actual human steering data.

Benchmark Model

A few of published results and architectures regarding the end-to-end deep learning based autonomous vehicle can be used as Benchmark. MIT project - DeepTesla - Deep Learning for Self-Driving Cars (https://selfdrivingcars.mit.edu (https://selfdrivingcars.mit.edu)) provides a model and a web-based train/test environment based on camera data received from Tesla autopilot system. NIVIA also published How End-to-End Deep Learning Steers a Self-Driving Car with its CNN-based model and test results (https://devblogs.nvidia.com/parallelforall/explaining-deep-learning-self-driving-car/ (https://devblogs.nvidia.com/parallelforall/explaining-deep-learning-self-driving-car/)). Commaai.com published its research on an RNN based self-driving model and test results(http://research.comma.ai (http://research.comma.ai)).

Evaluation Metrics

The project will adopt the RMSE as evaluation metrics. Root Mean Square Error: The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. The RMSD represents the sample standard deviation of the differences between predicted values and observed values. Wiki: https://en.wikipedia.org/wiki/Root-mean-square_deviation (https://en.wikipedia.org/wiki/Root-mean-square_deviation)

Project Design

Overview:

A theoretical workflow for this project will be

- Data prepare
- Iteration of:
    - pre-processing
    - Model training and validation
    - Parameter tuning
- Model test

Data Prepare

The first step is to get available datasets. The dataset will be downloaded from AirSim Github (https://aka.ms/AirSimTutorialDataset (https://aka.ms/AirSimTutorialDataset)). AirSim will also be used to generate another part of data for test and more data for training if needed. Comma.ai open-source data (http://research.comma.ai (http://research.comma.ai)) may be considered as another source that fits the requirements of this project.

Pre-Processing

Data pre-processing need to be done before being fed into the training network. From current point of view, a typical set of processing may include Image rescaling, Grayscale conversion, Synchronizing the image with the steering angle data and Encoding. More extra processing may be added based on the adjustment of the deep learning model. Scikit-image, numpy, pandas are ideal tools for this step.

Model training and validation

There are a few of architectures published showing a really good performance on steering angle prediction. I would like to try those architectures as a start point. There are NAVIDA CNN architecture (https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/ (https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/)) and comma ai RNN architecture (http://research.comma.ai (http://research.comma.ai)). 3D CNN architecture with its remarkable record on video processing and motion recognition may also be a high potential path to look into. In this step, tools like tensorflow and keras will be used.

Parameter tuning

Adjust hyperparameters over the course of training runs to achieve an optimal result. Considering the requirements of large computing power, cloud computing service(AWS EC2) may be adopted to accelerate the hyperparameter tuning iteration.

Model test

At last, the model will be test with an all-new test data. The evaluation metrics (ERMS) will be calculated by comparing with the ground truth (human controlled steering angle data)