Machine Learning Engineer Nanodegree

Capstone Proposal

End to end deeplearning for self-driving steering model

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 to predict Steering Angles, 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. Datasets from Udacity Open Source Self-driving Car project will be used to train the medel. The performance of the model will be measured in simulation, given a test dataset. The performance can also be evaluated by comparing with other published results of the Udacity Self-driving Car CHALLENGE 2.

Datasets and Inputs

Datasets are Obtained form Udacity Open Source Self-driving car project. The provided dataset includes solid information to give users a full picture of the driving situation of the car. In this project, we will mainly use two categories of data - camera imagery and steering wheel angle information to train the CNN.

Datasets: https://github.com/udacity/self-driving-car/tree/master/datasets/CH2 (https://github.com/udacity/self-driving-car/tree/master/datasets/CH2)

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 CNN based self-driving system. Providing the system sufficient data is the foundation to generate a reliable performance.

The proposed dataset contains camera data on multiple driving conditions. Description of the data:

- HMB_1: 221 seconds, direct sunlight, many lighting changes. Good turns in beginning, discontinuous shoulder lines, ends in lane merge, divided highway
- HMB_2: 791 seconds, two lane road, shadows are prevalent, traffic signal (green), very tight turns where center camera can't see much of the road, direct sunlight, fast elevation changes leading to steep gains/losses over summit. Turns into divided highway around 350s, quickly returns to 2 lanes
- HMB_4: 99 seconds, divided highway segment of return trip over the summit
- HMB_5: 212 seconds, guardrail and two lane road, shadows in beginning may make training difficult, mostly normalizes towards the end
- HMB_6: 371 seconds, divided multi-lane highway with a fair amount of traffic

The above data is provided in ROSbag format.

Solution Statement

As I mentioned before, the target of the project is to predict the steering angles for vehicles by using camera data. The main task is to create a model to link the input - camera data to the output - steering angles. Considering the type of the input as the image, Convolutional Neural Networks was selected given its abilities on visual applications. The problem itself can be looked as a regression task, because of the continuity of the steering wheel movement. The CNN will be designed to identify the features of camera images, recognize them and build the link to the expected result - steering angles. The system will be measured using RMS error between the predicted steering angle and the actual human steering data.

Benchmark Model

The open sourced solutions of Udacity self-driving car CHALLENGE 2 will be used as Benchmark model. The best model for the test dataset shows its performance at 0.0482910511682 RMSE. The model on 10th rank shows its performance at 0.1216643438960 RMSE. The system was evaluated with a test set. RMSE is calculated from the predicted steering angle against the actual steering angle from the test set.

Leaderboard of the Udacity self-driving car CHALLENGE 2 can be found at:
https://github.com/udacity/self-driving-car/tree/master/challenges/challenge-2
(https://github.com/udacity/self-driving-car/tree/master/challenges/challenge-2)

Evaluation Metrics

The project will adopt the same evaluation metrics as the Udacity self-driving car CHALLENGE 2. 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 main Dataset will download from Udacity Self-driving car CHALLENGE 2 (https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo/data (https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo/data)). 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. Considering the format difference of the data, some pre-processing need to be done to transform the data to lists of jpg images, which is the type of training input for the Neural Network. rwightman's udacity-driving-reader （https://github.com/rwightman/udacity-driving-reader） (https://github.com/rwightman/udacity-driving-reader）) is an ideal tool to convert the ROSBAG to jpg.

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 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)