Behavioral Cloning Project Report

Yi-Jhih Jimmy

February 12, 2017

1 Project Overview

In this project, I implemented a self-driving car which can drive itself on a track of the simulator. The self-driving car used deep neural network as a computational framework to "learn" the driving behavioral of human. That is, I have to "drive" the simulated car on the simulator to collect training data. Then, the self-driving robot learn my behavior. Although Udacity has provided a set of training data, I cannot cause the driving robot stay within yellow lane until I began to collect my own training data. At last, I use my own data completely.

The submitted files are as below:

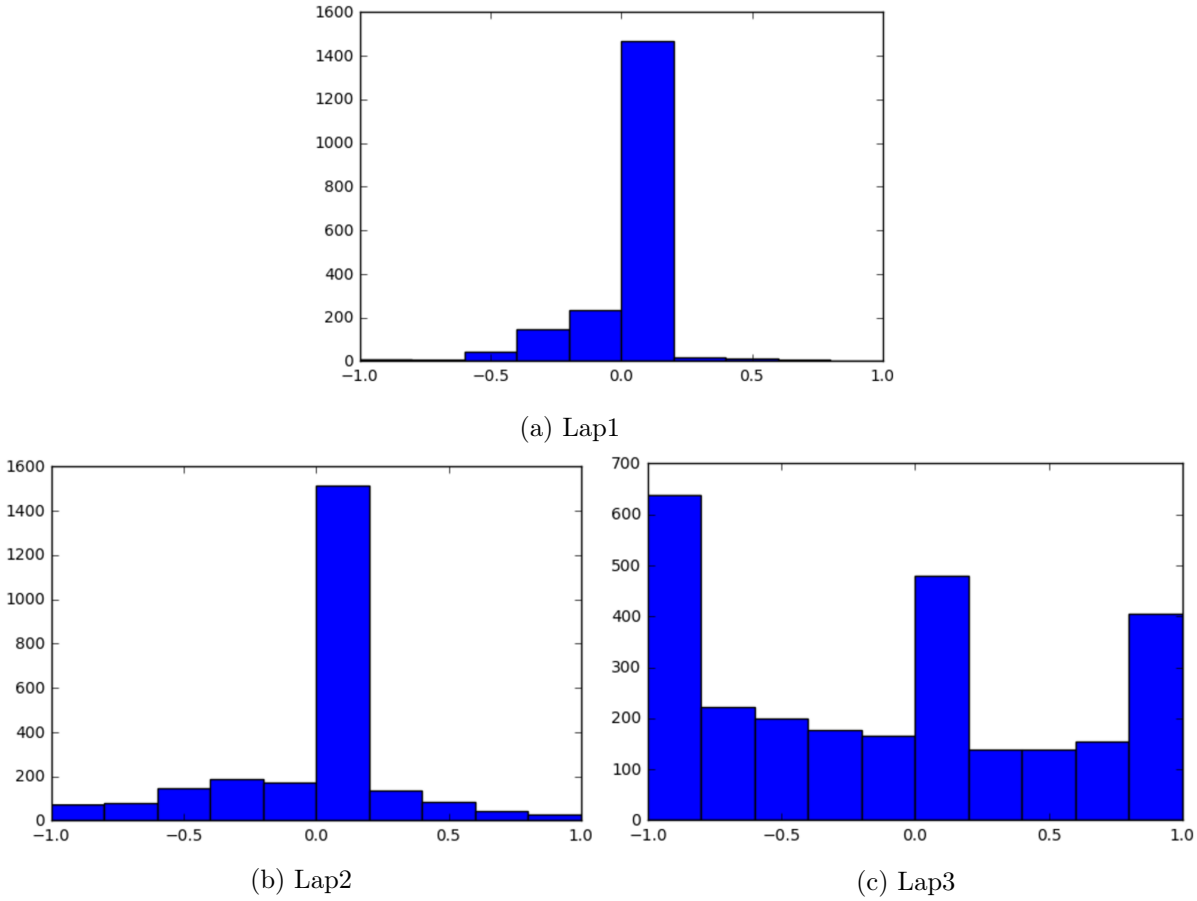
- model.py - The script used to create and train the model.
- drive.py - The script to drive the car. I make some modifications: resize and normalize images, reduce the throttle, etc.
- model.json - The model architecture.
- model.h5 - The model weights.
- report.pdf - explains how the training data were collected, the structure of my network, and training approach.
- README.md - lists the files included in this project.

2 Data Collection

I collected my own data and used the collected data completely to derive proportional coefficients for training my deep neural network. Training data is collected from total three laps as the following method:

- First lap: I let the car runs on the center of the road;
- second lap: I let the car runs on the near-center range of the road;
- third lap, I let the car runs on the extreme range of the road. That is, let the car on extreme right-hand-side and extreme left-hand-side of the road.

Figure 1 shows the histograms of the steering in the three laps.



(a) Lap1

(b) Lap2

(c) Lap3

Figure 1: The Histograms of the `steering` in three laps.

We can observe that in the first lap, there are few large angles. Besides, because in the simulator where left turns are more than right turns, so that negative steering angles are more than positive steering ones. In lap2, the range of steering angles are slightly larger than that of lap1. In lap3, there are many extreme large steering angles, +1 and -1. Using these histograms, I designed a controller similar to the concept of proportional control [5]. I divided the captured images into five classes:

- Lap 1 center images: the images indicate the car is on the center of the road, I set the steering angles to 0.

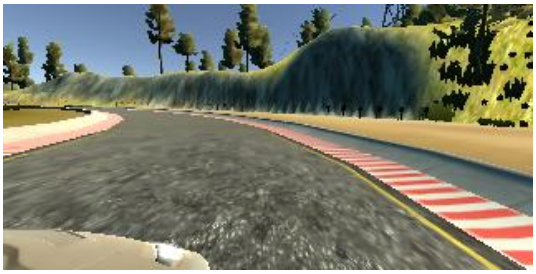
- Lap 1 left images: the images indicate the car is slightly to the left, I set the steering angles to +0.35.
- Lap 1 right images: the images indicate the car is slightly to the right, I set the steering angles to −0.35.
- Lap 3 left images: the images indicate the car is seriously to the left, I set the steering angles to +0.80.
- Lap 3 right images: the images indicate the car is seriously to the right, I set the steering angles to −0.80.

3 Data Visualization and Preprocessing

Let's have a glance at the images captured by the front cameras. Figure 2(a) is an image taken by the center camera; Figure 2(b) is an image taken by the left camera; and Figure 2(c) is an image taken by the right camera.

(a) Center

(b) Left



(c) Right

Figure 2: The images captured by the front cameras.

What follows is suggested by Denise R. James [3]. Since the front of the car is not relevant in the training data, 20% or 32 pixels are removed from the bottom of the image. Besides, 25 pixels are removed from the top of the image to not train the trees and sky but the road [3]. One of a cropped image is as shown in Figure 3.

Figure 3: A cropped image

Then, the cropped images are sent to normalizer for training deep neural network model to get better performance after training.

4 Model Architecture Design

The model architecture is inspired from the NVIDIA paper [1]. There are 10 layers as what follows.

- input planes: 3@66x208
- 1st, conv layer: 5x5 kernel, accepts input planes 3@66x208, stride 2, same padding;
- 2nd, conv layer: 5x5 kernel, convolutional feature map 24@33x104, stride 2, same padding;
- 3rd, conv layer: 5x5 kernel, convolutional feature map 36@17x52, stride 2, valid padding;
- 4th, conv layer: 3x3 kernel, convolutional feature map 48@7x24, stride 1, valid padding;
- 5th, conv layer: 3x3 kernel, convolutional feature map 64@5x22, stride 1, valid padding;

- 6th, flatten: the output of 5th conv layer is (3,20,64), which is flattened to 3840 neurons.
- 7th, fully-connected layer, 100 neurons;
- 8th, fully-connected layer, 50 neurons;
- 9th, fully-connected layer, 10 neurons;
- 10th, fully-connected layer, 1 neuron.

4.1 Dropout Layers

I have tried to add dropout layers after conv layers, but I feel this problem cannot be improved via dropout layers.

5 Architecture Characteristics

As mentioned in the previous section, the first layer accepts the three input planes. The convolutional layers were designed to perform feature extraction and were chosen empirically through a series of experiments that varied layer configurations. The first three convolutions are "strided" with a 2×2 stride and a 5×5 kernel and a non-strided convolution with a 3×3 kernel size in the last two convolutional layers [1].

Following the five convolutional layers, there are three fully connected layers leading to an output that means a control value. The fully connected layers are designed to function as a proportional controller for steering. The deepen network served not only as a feature extractor, but also a controller. However, it is not possible to make a clean break between which parts of the network function primarily as feature extractor and which serve as controller [1]. Note that for each layer, the ELUs are adopted instead of the popular ReLUs. The benefits of ELUs over ReLUs have been published in the paper [2].

As for optimization, `Adam` is adopted.

6 Model Training

For training the model, the keras `fit_generator()` function is adopted to save memory space. A user-defined generator was created to send a batch size of one to the keras `fit_generator()`. In my generator, I implemented the concept of proportional controller. The generator reads images from training data, and generates a pair of (image, control value) for model training. In this project, I divided the captured images into five classes:

- Lap 1 center images: the images indicate the car is on the center of the road, I set the steering angles to 0.
- Lap 1 left images: the images indicate the car is slightly to the left, I set the steering angles to +0.35.
- Lap 1 right images: the images indicate the car is slightly to the right, I set the steering angles to −0.35.
- Lap 3 left images: the images indicate the car is seriously to the left, I set the steering angles to +0.80.
- Lap 3 right images: the images indicate the car is seriously to the right, I set the steering angles to −0.80.

References

[1] M. Bojarski et al., "End to end learning for self-driving cars," NVIDIA Corporation, http://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

[2] Djork-Arne Clevert, "Fast and accurate deep network learning by exponential linear units (ELUS)," https://arxiv.org/pdf/1511.07289v1.pdf

[3] Denise James, "Udacity self driving car project 3 – deep learning to clone driving behavior," https://medium.com/@deniserjames/denise-james-bsee-msee-5beb448cf184#.xrhv22wje

[4] Vivek Yadav, "An augmentation based deep neural network approach to learn human driving behavior," https://chatbotslife.com/using-augmentation-to-mimic-human-driving-496b569760a9#.5lbgw1wwn

[5] Wikipedia, Proportional control, https://en.wikipedia.org/wiki/Proportional_control