Behavioral Cloning

Writeup Report

Behavioral Cloning Project

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road
- Summarize the results with a written report

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

My model consists of 5 convolution neural network with three 5x5 filters and two 3x3 filters (model.py lines 71-75)

After the CNN, the model has three fully connected layers with neurons of 100, 50 and 10. Each fully connected layer uses RELU as activation function. (model.py lines 78-80)

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 76).

The model was trained and validated on different data sets to ensure that the model was not overfitting. The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually (model.py line 110).

4. Appropriate training data

Training data was collected by driving the car in the center of the road for at least one loop in the Udacity simulator. At each frame, the center, left and right camera catches an image and the vehicle driving data (steering angle, throttle, brake, speed) is recorded accordingly. The images from three cameras are the mainly input data, and the steering angle value is set as ground truth.

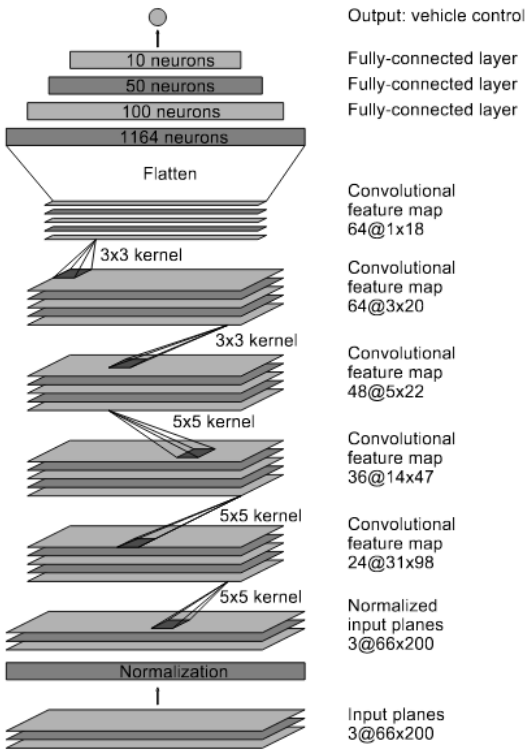
Model Architecture and Training Strategy

1. Final Model Architecture

The final model architecture is taken reference from Nvidia's research paper: "*End to End Learning for Self-Driving Cars*".

The final model architecture (model.py lines 69-82) consisted of a normalized layer, three convolution layer with 5x5 filter, two convolution layer with 3x3 filter, one fully connected layer with 100 neurons, one fully connected layer with 50 neurons, one fully connected layer with 10 neurons and the final output layer with only one output indicates the predict steering angle.

Here is the neural network architecture:



3. Creation of the Training Set & Training Process

To capture good driving behavior, I recorded one lap on drving the car in the center lane. Here is an example image of center lane driving:

Center Camera

Meanwhile the left and right camera images are both captured for teaching the car how to turn left and turn right.

Left Camera

Right Camera

With one loop of three cameras data, the model can be trained. But even with the best turned model from Nvidia example, the test result is not good enough. It is okay to record one more loop of data, but we can also do some modification on the existing first loop of data randomly (50% for example), to enlarge the information from the three camera images. The modification includes flip the image (left ← →right), while negative the steering angel. Shift the image virtually and horizontally, add shadow or brightness.

After the input data are modified randomly, we got more information for training the model. We can also preprocess input data before input to the network, which is crop, resize and convert the color. Crop is to remove some background from the image to reduce not useful information, resize is to match the input size of the neural network. Convert color from RGB to YUV is suggested in Nvidia paper. The preprocess functions can be found in the help file utlis.py.

Thanks to Siraj Raval's great job done before, I learned how to organize the code in model.py for a better coding style. Model.py has three basic functions of load data, build model and train model. These three steps can be considered as the pipeline of this project. The training parameters such as number of epochs, batch size, learning rate are packaged as arguments in the main function (line 140-149).

The call the main function to process the three steps in the pipeline to train the model and generate the model.h5 file.