Behavioral Cloning

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

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Files Submitted & Code Quality

1. Submission includes all required files and can be used to run the simulator in autonomous mode

My project includes the following files:
* model.py containing the script to create and train the model
* drive.py for driving the car in autonomous mode
* model.h5 containing a trained convolution neural network
* writeup_report.md or writeup_report.pdf summarizing the results

2. Submission includes functional code

Using the Udacity provided simulator, drive.py file and my model.h5 , the car can be driven autonomously around the track by executing

```
sh
python drive.py model.h5
```

3. Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

My model consists of a convolution neural network with 5x5 & 3x3 filter sizes and depths between 24 and 64 (model.py lines 95-116)

The model includes RELU layers to introduce nonlinearity (code line 20), and the data is normalized in the model using a Keras lambda layer (code line 18).

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 98,102,106,110).

The model was trained and validated on different data sets to detect that if the model was overfitting (code line 10-16). 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 132).

4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left & right sides of the road,

- 5 laps of center lane driving(clock wise and counter-clock wise)
- 2 lap of recovery driving from the sides
- 2 lap focusing on driving smoothly around curves
- some examples of passing the bridge

For details about how I created the training data, see the next section.

Model Architecture and Training Strategy

1. Solution Design Approach

My first step was to build a 1 layer linear model to build up the environment.

The second step was to use a convolution neural network model similar to the LeNet5. I thought this model might be appropriate in some degree because it wording well in simple CV recognition task.

Firstly, my strategy was training on the dataset Udacity provided, which actually was a 4 laps of center lane driving(clock wise and counter-clock wise). After trained that model, save it, and continue to train in the datasets of special situations, such as sharp curves, passing the bridge. However, it works poorly after reinforce the model with special situations. Then I find that the model was overfitting to those small dataset.

Secondly, I merge the special situation dataset to the original 4 laps dataset, which is about 10k samples in total. But even after 5~15 epochs, the model didn't become overfitting, the MSE of training and validation set are fluttering in small ranges. As the architecture is already more complicating than the NVIDIA architecture, so I thought there are too many tricks to combat overfitting, as too many pooling layer and too high dropout ratio.

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. I found that my first model had a low mean squared error on the training set but a high mean squared error on the validation set. This implied that the model was overfitting.

The final step was to run the simulator to see how well the car was driving around track one. There were a few spots where the vehicle fell off the track:

- After 1st curve, it often rush out to the lake
- On the bridge, it sometimes crush to the side
- The sharp curve

  to improve the driving behavior in these cases, I

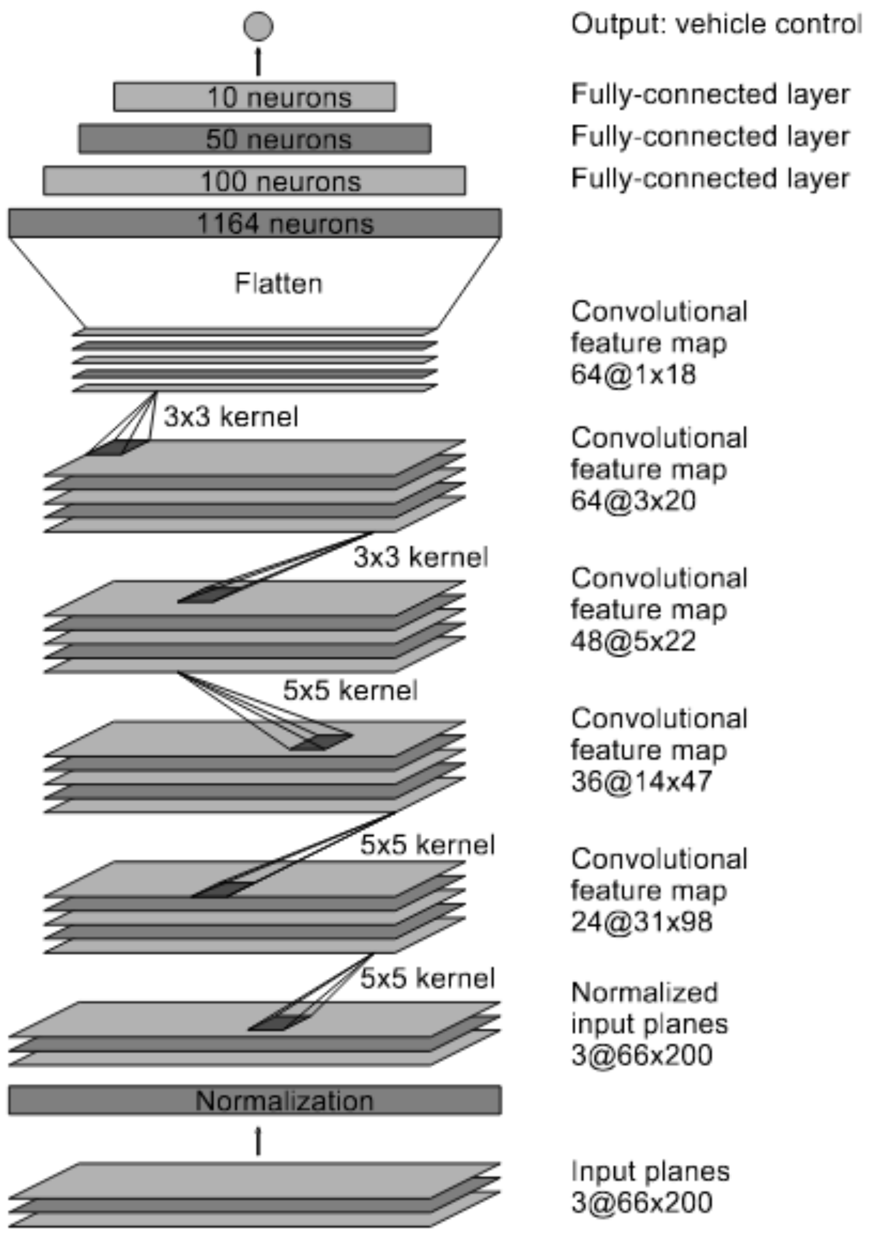
- 5 laps of center lane driving(clock wise and counter-clock wise)
- 2 lap of recovery driving from the sides
- 2 lap focusing on driving smoothly around curves

- some examples of passing the bridge

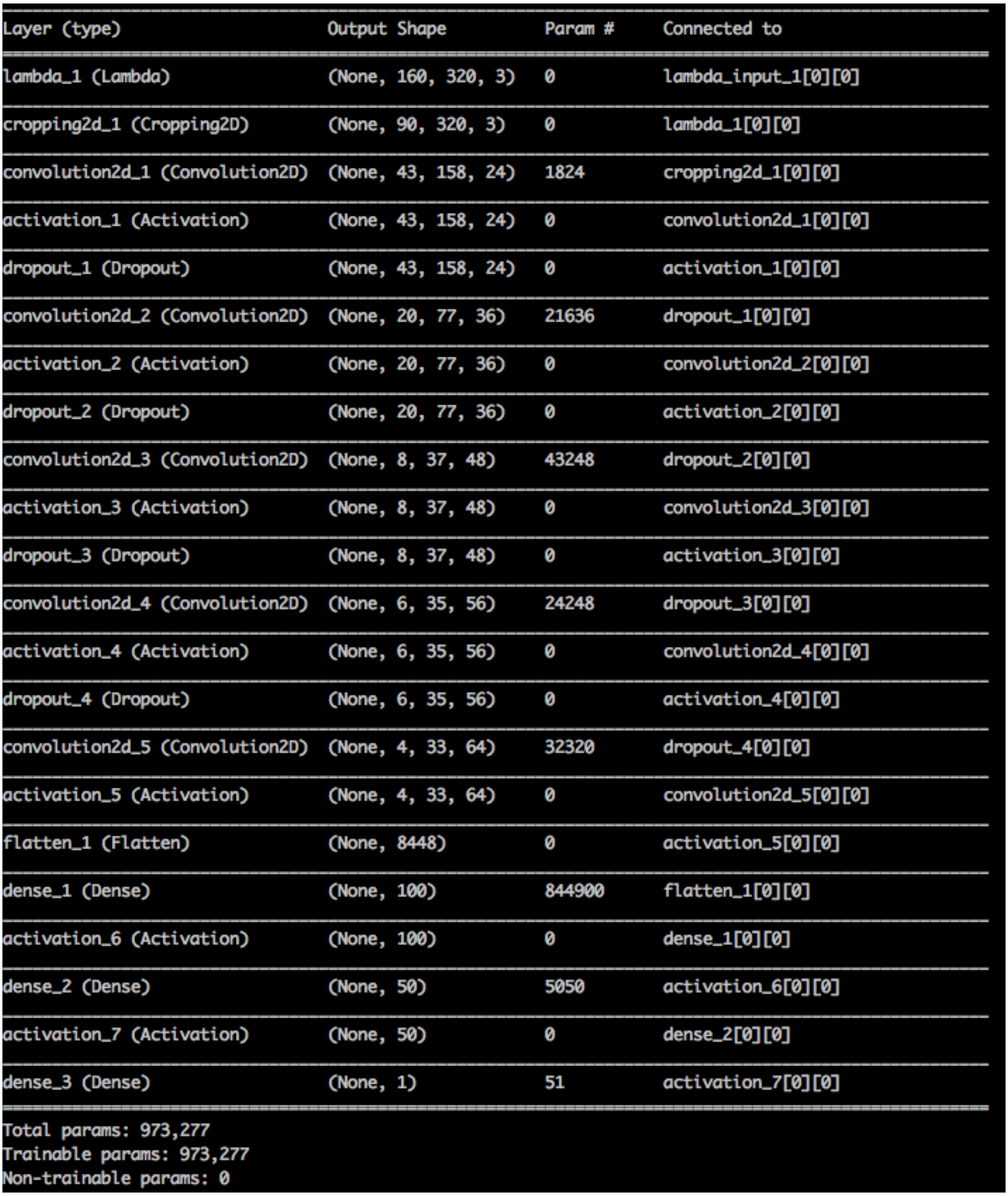
At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

The final model architecture (model.py lines 96-127) was build based on the NVIDIA paper End to End Learning for Self-Driving Cars.



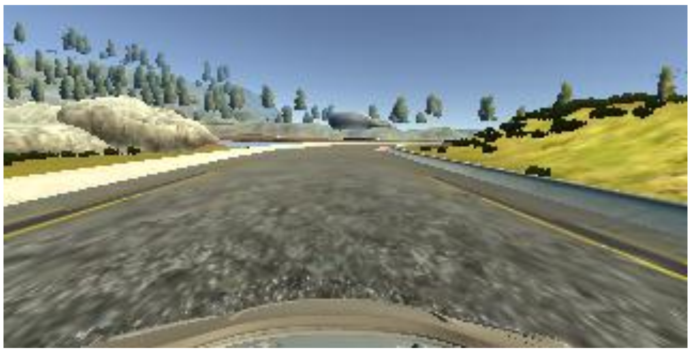
Here is a visualization of the architecture (note: visualizing the architecture is optional according to the project rubric)



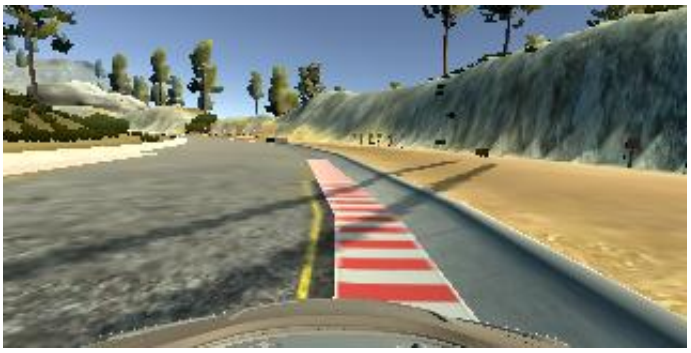
```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
lambda_1 (Lambda)                (None, 160, 320, 3)   0           lambda_input_1[0][0]
____________________________________________________________________________________________________
cropping2d_1 (Cropping2D)        (None, 90, 320, 3)    0           lambda_1[0][0]
____________________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 43, 158, 24)   1824        cropping2d_1[0][0]
____________________________________________________________________________________________________
activation_1 (Activation)        (None, 43, 158, 24)   0           convolution2d_1[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 43, 158, 24)   0           activation_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 20, 77, 36)    21636       dropout_1[0][0]
____________________________________________________________________________________________________
activation_2 (Activation)        (None, 20, 77, 36)    0           convolution2d_2[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 20, 77, 36)    0           activation_2[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 8, 37, 48)     43248       dropout_2[0][0]
____________________________________________________________________________________________________
activation_3 (Activation)        (None, 8, 37, 48)     0           convolution2d_3[0][0]
____________________________________________________________________________________________________
dropout_3 (Dropout)              (None, 8, 37, 48)     0           activation_3[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 6, 35, 56)     24248       dropout_3[0][0]
____________________________________________________________________________________________________
activation_4 (Activation)        (None, 6, 35, 56)     0           convolution2d_4[0][0]
____________________________________________________________________________________________________
dropout_4 (Dropout)              (None, 6, 35, 56)     0           activation_4[0][0]
____________________________________________________________________________________________________
convolution2d_5 (Convolution2D)  (None, 4, 33, 64)     32320       dropout_4[0][0]
____________________________________________________________________________________________________
activation_5 (Activation)        (None, 4, 33, 64)     0           convolution2d_5[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 8448)          0           activation_5[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 100)           844900      flatten_1[0][0]
____________________________________________________________________________________________________
activation_6 (Activation)        (None, 100)           0           dense_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 50)            5050        activation_6[0][0]
____________________________________________________________________________________________________
activation_7 (Activation)        (None, 50)            0           dense_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 1)             51          activation_7[0][0]
====================================================================================================
Total params: 973,277
Trainable params: 973,277
Non-trainable params: 0
```

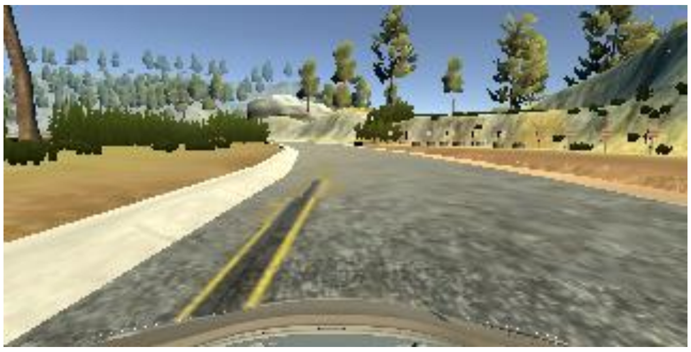
3. Creation of the Training Set & Training Process

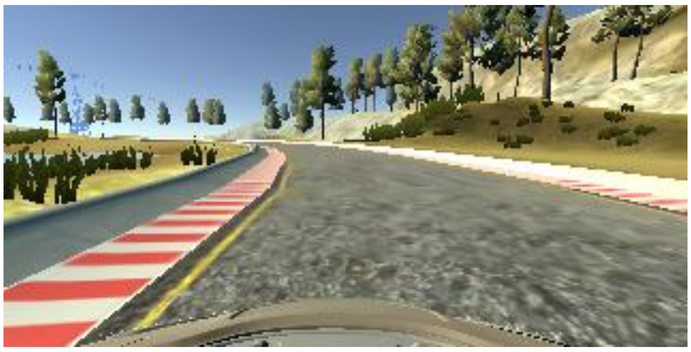
To capture good driving behavior, I first recorded five laps both clockwise and counter-clockwise on track one using center lane driving. Here is an example image of center lane driving:



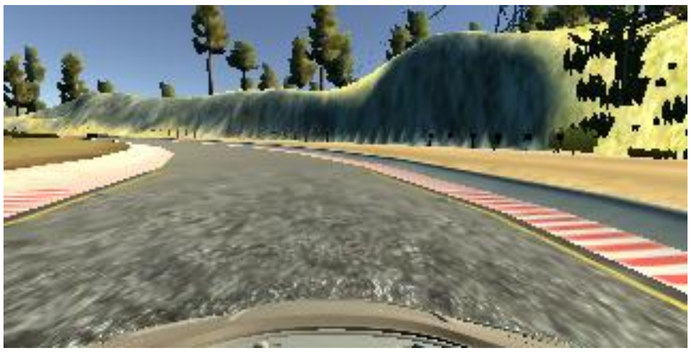
I then recorded the vehicle recovering from the left side and right sides of the road back to center so that the vehicle would learn to keep straight, go through the bridge, turn the curve smoothly, and go back to the road center. These images show what a recovery looks like starting from ... :

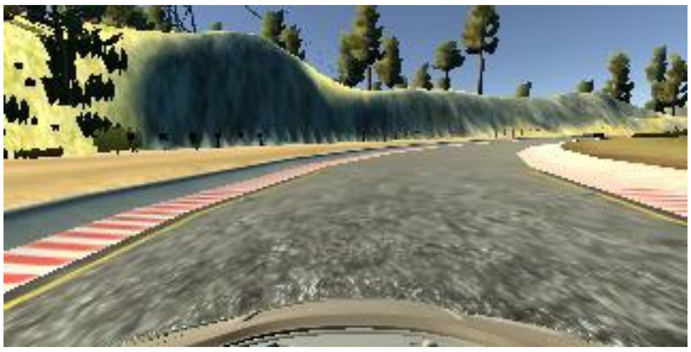






To augment the data sat, I also flipped images and angles thinking that this would reduce the impact of fixed direction. For example, here is an image that has then been flipped:





After the collection process, I had 16,816 number of data points. I then preprocessed this data by normalized, reduce mean and cropping.

I finally randomly shuffled the data set and put 20% of the data into a validation set.

I used this training data for training the model. The validation set helped determine if the model was over or under fitting. The ideal number of epochs was 10 as evidenced by large mount of experiments. I used an adam optimizer so that manually training the learning rate wasn't necessary.

Others

I use AWS GPU following Udacity AWS tutorial at first, then I try to finish the work in the server of our lab,which is equipped with 2 Intel E5 CPU,4 TITAN X, 256GB RAM, it was speed up at least 40%.

What's more

There is some kind of bug makes me extremely crazy and angry, even makes me a strong feeing of hopelessness.

My model always fail when I test in autonomous mode as I choose the "FAST" Graphics Quality. I spent all day to change model and collect more data again and again and again, what's more, nothing seems help.

I change the simulator Graphics Quality to "FANTASTIC" by chance as it was boring when the server training the model, my model just become work.