Behavioral Cloning
Writeup

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 summarizing the results

2. Submission includes functional code

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

```
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 3x3 filter sizes and depths between 24 and 64(model.py lines 66-80)

The model includes RELU layers to introduce nonlinearity (code line 69-73), and the data is normalized in the model using a Keras lambda layer (code line 67).

2. Attempts to reduce overfitting in the model

The model contains two dropout layers in order to reduce overfitting (model.py lines 76,78).

The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 30). 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 84).

4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I first tired the training data provided by Udacity and got good performance.

Then I tried my own data, which is a combination of 2 laps of center lane driving, 1 lap of recovering from the left and right sides of the road and 1 lap of counter-clockwise direciton. However, the result got worse. The reason maybe is my bad driving skill, maybe is my network still to tune, I'm still working on it.

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

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to build a conv network which input is vehicle camera image, and output is steering angle.This network shall be trained by human driving data.

My first step was to use a convolution neural network model similar to the LeNet, I thought this model might be appropriate because it is a basic convolution neural network.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 tried this model on the simulator, the vehicle start driving, it just prove my step is generally correct.

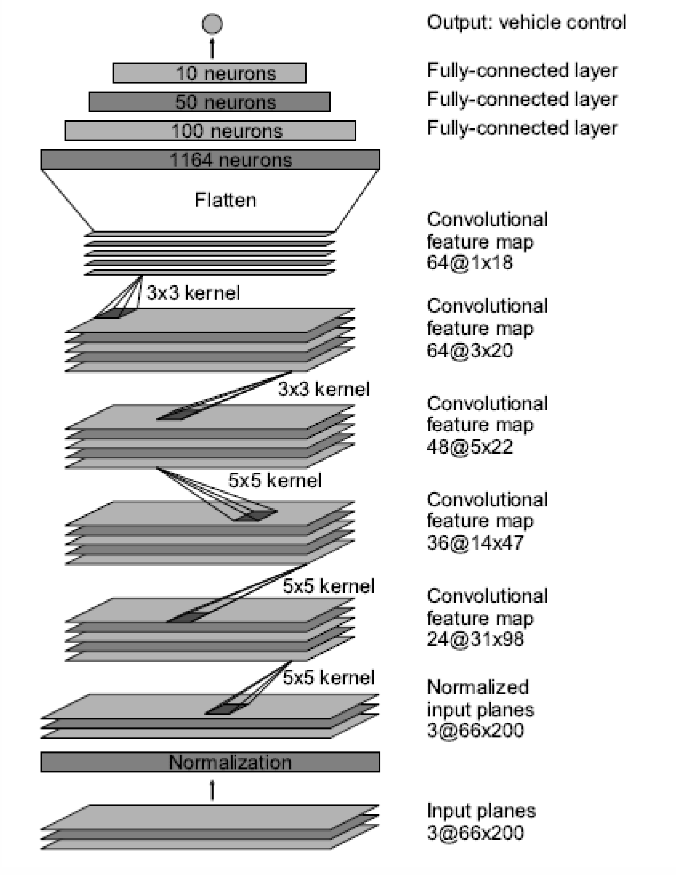
Then I improved my netwrok to Nvdia CNN Architecture to improve the performance.To combat the overfitting, I added two dropout layers.

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. To improve the driving behavior in these cases, I did as follows:
1. Using Multiple Cameras, it trippled the dataset and tell the vehicle how to do when at the edge of the track.
2. Flipping Images and Steering Measurements, it doubled the dataset.
3. Cropping Image by cropping the 50 pixels from the top and 20 pixels from the bottom.
4. Collect more data.

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

2. Final Model Architecture

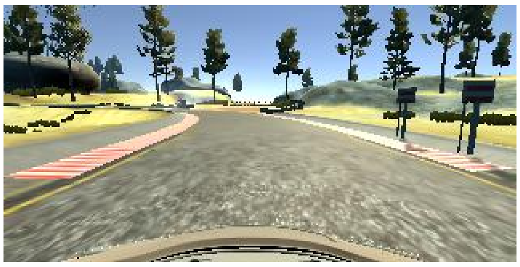
The final model architecture (model.py lines 66-80) consisted of a convolution neural network with the following layers and layer sizes:
InputLayer: 320x65@3
Convolution Layer: kerel:5x5 pooling:2,2 200x66@3
Convolution Layer: kerel:5x5 pooling:2,2 98x31@24
Convolution Layer: kerel:5x5 pooling:2,2 47x14@36
Convolution Layer: kerel:3x3 22x5@48
Convolution Layer: kerel:3x3 20x3@64
Flatten 18x1@64
Fully-connected layer 1164
Dropout layer 0.5
Fully-connected layer 100
Dropout layer 0.5
Fully-connected layer 50
Fully-connected layer 10
OutputLayer: 1



Final Architecture

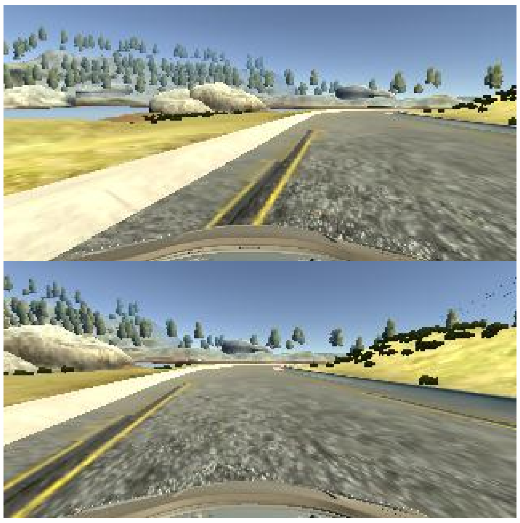
3. Creation of the Training Set & Training Process

To capture good driving behavior, I first recorded two laps on track one using center lane driving. Here is an example image of center lane driving:



Center 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 how to steering back from the edge of the lane. These images show what a recovery looks like:



I aslo record one counter-clockwise direciton lap.

Then I repeated this process on track two in order to get more data points.

After the collection process, I had 17635 number of data points. I then preprocessed this data by Lambda Layer (Lambda(lambda x: (x / 255.0) - 0.5))

Then I flip the image and use three cameras to add training data further more.

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 5 as evidenced by the model start to overfit. I used an adam optimizer so that manually training the learning rate wasn't necessary.