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

Here is the link to my project code.

Files Submitted & Code Quality

1. Submission includes all required files

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.pdf 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
```

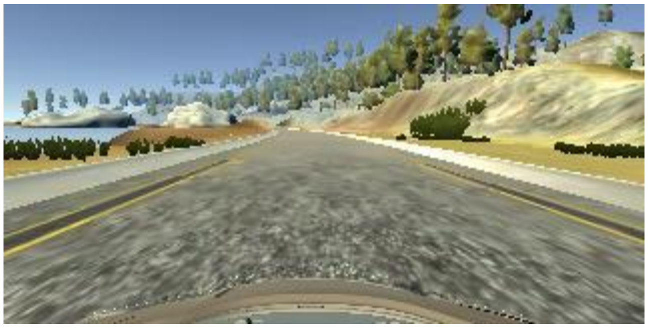
3. Submission code is usable and readable

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.

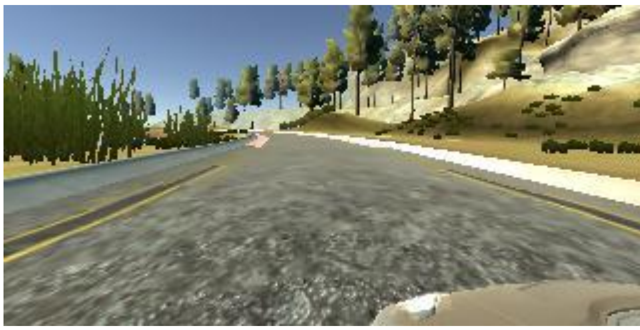
Training Data and Model Architecture

1. Training data

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:



Then I used left and right camera images by adding a correction of 0.25 and subtracting the same correction respectively. As result, I obtained three times as many training data as I only used the center camera images. Here are the examples of left and right camera images:



Finally, I had 10605 examples in the data set and I randomly shuffled it before put 20% of the data into a validation set.

2. Model architecture

My final model consisted of the following layers:

Layer	Description
Input	160×320×3 images
Lambda Layer	Normalize the image data
Cropping Layer	Crop the image to 90×320×3
Convolution 24@5×5	2×2 stride, valid padding, outputs 43×158×24
Batch Normalization	
RELU	
Convolution 36@5×5	2×2 stride, valid padding, outputs 20×77×36
Batch Normalization	
RELU	
Dropout	dropout rate 0.3
Convolution 48@5×5	2×2 stride, valid padding, outputs 8×37×48
Batch Normalization	
RELU	
Dropout	dropout rate 0.4
Convolution 64@3×3	1×1 stride, valid padding, outputs 6×35×64
Batch Normalization	
RELU	
Dropout	dropout rate 0.5

Convolution 64@3×3	1×1 stride, valid padding, outputs 4×33×64
Batch Normalization	
RELU	
Dropout	dropout rate 0.5
Flatten	
Fully connected	inputs 8448, outputs 100
Fully connected	inputs 100, outputs 50
Fully connected	inputs 50, outputs 10
Fully connected(output)	inputs 10, outputs 1

I used dropout and batch normalization layer to avoid overfitting. The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 88-90). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

I used **adam** optimizer, **mean square error** as loss function and batch size of 64 to train the model for 40 epochs.