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

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 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 is based on Nvidia model “End to End Learning for Self-Driving Cars”. The model includes RELU layers to introduce nonlinearity (code line 20), and Cropping2D method from Keras is applied o reduce parameters and training time of the model. (model.py line 96), by removing the sky and the small area in front of vehicle.

2. Attempts to reduce overfitting in the model

The model was trained and validated on different data sets(left,right and center) to ensure that the model was not overfitting (code line 65,66). The model was tested by running it , with 0.2 as the correction factor of steering angles , through the simulator and ensuring that the vehicle could stay on the track. Further more, I used sklearn library function shuffle() to randomize training data.(code lineline 73) and train_test_split() to create validation data from training data (code lineline 82).

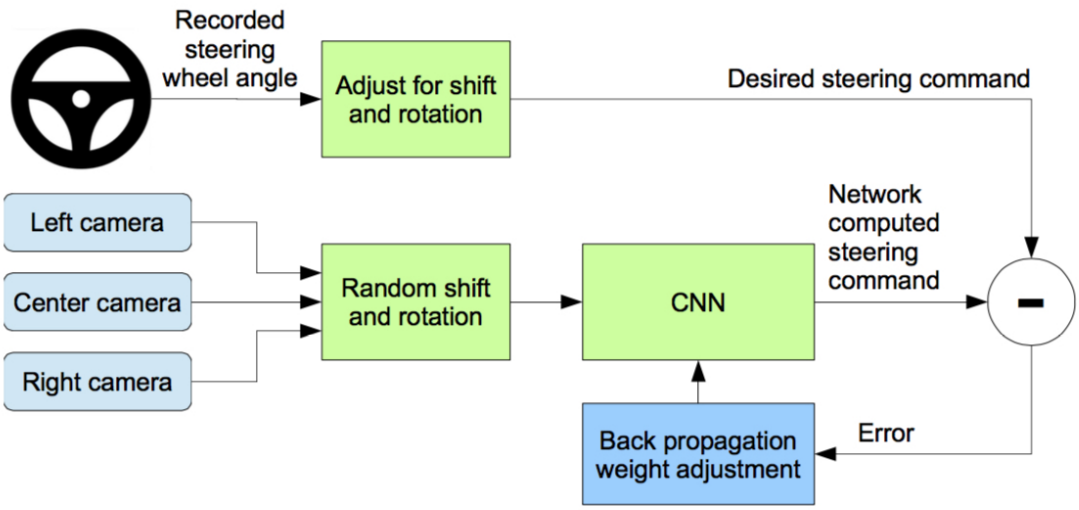

Figure 2: Training the neural network.

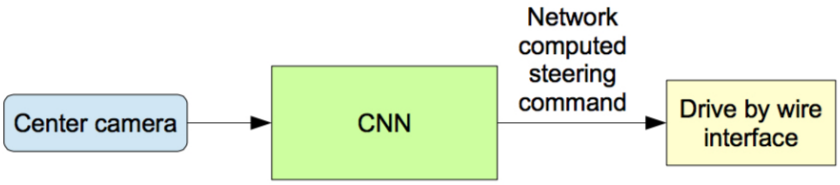

Figure 3: The trained network is used to generate steering commands from a single front-facing center camera.

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 chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left and right sides of the road. I added left and right images to train the vehicle recover from the left and right sides of the road.

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to keep all the pre-processing technology and trained Nvidia model . That is, use a convolution neural network model similar to the Nvidia model. Because I thought this model might be appropriate because the result from their paper sounds great.
By using all the dataset(left, center and right), the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

The final model architecture Nvidia (model.py lines 88-108) consisted of a convolution neural network with the following layers and layer sizes 24@31x98, 36@14x47, 48@5x22, 64@3x20, 64@1x18. Then it has a flatten layer followed by 3 fully connected layers outputting 100 neurons, 50 neurons and 10 neurons.

3. Creation of the Training Set & Training Process

After several tries, I finally trained it with 3 epoch. One result video is included in the folder.