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

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.pdf summarizing the results

2. Submission includes functional code

Using the Udacity provided simulator and 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

I used a convolution neural network model similar to the NVIDIA model.

The model includes ELU layers to introduce nonlinearity, and the data is normalized in the model using a Keras lambda layer.

The model looks like:

- Image normalization

- Cropping the image with this dimensions ((70,20),(0,0))
- Convolution: filter depth=24, filter size= 5x5, stride= 2x2, activation: ELU
- Convolution: filter depth=36, filter size= 5x5, stride= 2x2, activation: ELU
- Convolution: filter depth=48, filter size= 5x5, stride= 2x2, activation: ELU
- Convolution: filter depth=64, filter size= 3x3, stride= 1x1, activation: ELU
- Convolution: filter depth=64, filter size= 3x3, stride= 1x1, activation: ELU
- Flatten image from 2D to side by side
- Fully connected layer: Dense= 100, activation: ELU
- Dropout layer to avoid overfitting = 0.25
- Fully connected layer: Dense= 50, activation: ELU
- Fully connected layer: Dense= 10, activation: ELU
- Fully connected layer: 1 output

Here is a visualization of the architecture.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda_1 (Lambda)            (None, 160, 320, 3)       0
_________________________________________________________________
cropping2d_1 (Cropping2D)    (None, 70, 320, 3)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 33, 158, 24)       1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 15, 77, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 37, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 35, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 33, 64)         36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 4224)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               422500
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 559,419
Trainable params: 559,419
Non-trainable params: 0
```

2. Attempts to reduce overfitting in the model

The model contains dropout layer(0.25) in order to reduce 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

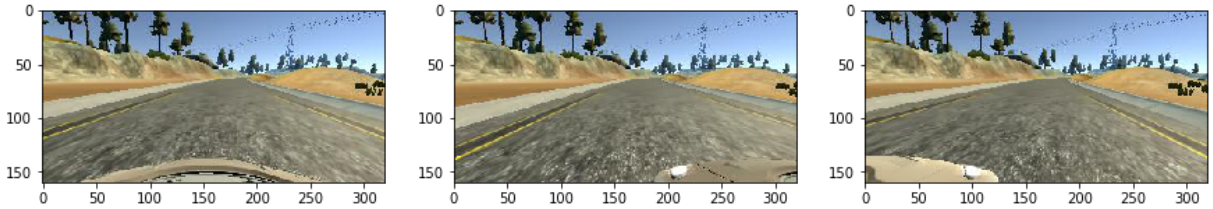
The model used an adam optimizer, so the learning rate was not tuned manually.

Model Architecture and Training Strategy

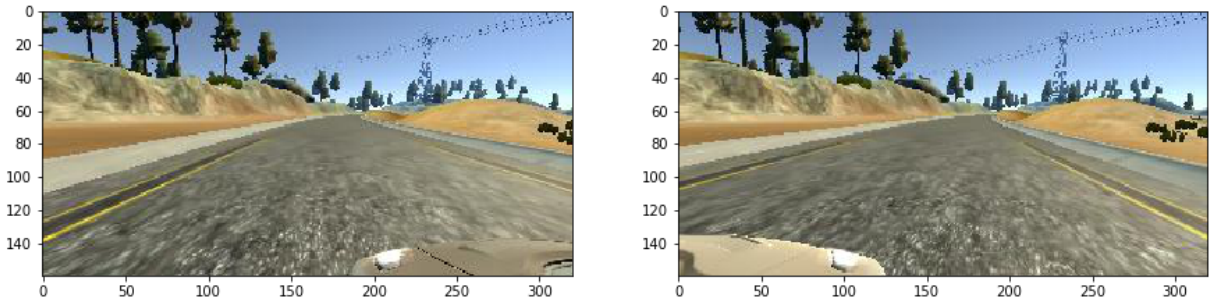
Creation of the Training Set & Training Process

I analyzed the Udacity Dataset and decided to use the data for training and validation. I splited the dataset into training and Validation set using sklearn.model_selection and used 20% of the data in Validation set.

Here is an example image of center, left and right lane driving:



To augment the data sat, I also flipped images and angles. For example, here is an image that has then been flipped:



Testing model on simulator

The final step was to run the simulator to see how well the car was driving around track. To record the video and save in mp4 file, I used

```
python video.py video
```

Here's a link to my video result (./Untitled/video.mp4)

.