#**Behavioral Cloning**

Files Included which need to be considered:
1. Model.py
2. readingCsv.py
3. img_manipulation.py
4. data_p3.tar.xz
5. Model.h5
6. Model.json
7. runnning.mp4
8. Folder running

NOTE: Additionally folder first_run, Model1.h5, model1.json have been included but are not to be used for evaluation.

###Model Architecture and Training Strategy

####1. An appropriate model architecture as used my NVIDIA for their End to End Learning for Self-Driving Cars has been used. Link : http://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

My model consists of 4 convolution neural networks and 4 fully connected neural networks. (model.py lines 33-50)

The model uses RELU and TANH layers to introduce nonlinearity , and the data is normalized and zero meaned in the model using a Keras lambda layer (model.py line 34).

####2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting with a dropout probabilty of 25% . The value 25% was choose as it seems to give best result.

The model was trained and validated on different data sets to ensure that the model was not overfitting (model.py line 60). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track. The output can car running autonomously can be seen in video running.mp4.

####3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually (model.py 25).
Other parameters for batch size and number of epochs where experimentally determined and finally batch size of 1000 and 10 epochs were used.
Mse was used to evaluate the performance.

####4. Appropriate training data

Training data was collected in the following ways:
1. 2 laps tarck1 driving in the center clockwise direction.
2. 2 laps track1 driving in the center anticlockwise direction.
3. 2 laps tarck1 recovering clockwise.
4. 2 laps tarck2 driving in the center clockwise direction.
5. 2 laps tarck2 driving in the center anticlockwise direction.
6 1 lap track2 recovering clockwise.

###Model Architecture and Training Strategy

####1. Solution Design Approach

The model in terms of its architecture was not much experimented with and NVIDA architecture was followed by adding additional dropout layers to avoid over fitting.

As the first measure I tested my system only with track1 clockwise and track 1 anti clockwise data which let to car drifting at the first curve and not recovering. Then I introduced the track 2 data to generalize but it didn't improve my system by rather the accuracies dropped to 45% which was surprising.

To finally train the model training data from track 2 was not included at all and only data from track 1 clockwise, anticlockwise and recovering was used.

At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.
When the same model was tested on track 2 it didn't perform as well.

####2. Final Model Architecture

The final model architecture with dropout layers is as shown in model.py line (33-50)

####3. Creation of the Training Set & Training Process

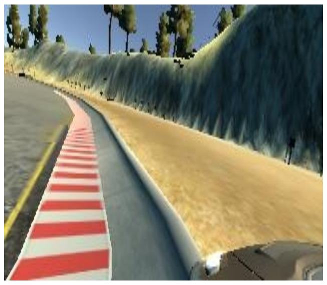
To capture good driving behavior, Training data was collected in the following ways:
1. 2 laps tarck1 driving in the center clockwise direction.
2. 2 laps track1 driving in the center anticlockwise direction.
3. 2 laps tarck1 recovering clockwise.
4. 2 laps tarck2 driving in the center clockwise direction.
5. 2 laps tarck2 driving in the center anticlockwise direction.
6 1 lap track2 recovering clockwise.

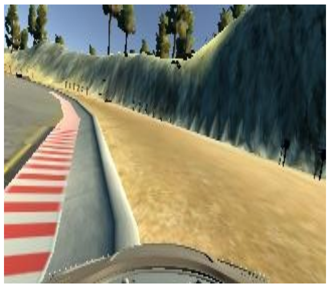
Here are set of example images collected track 1
i.e. first 3 cases as mentioned above.

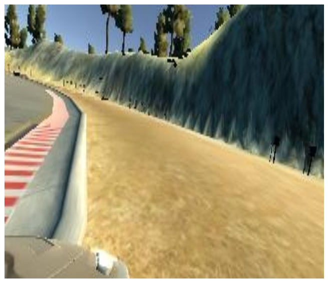
1.   LEFT   0.0   CENTER 0.0   RIGHT 0.0

2.   0.0   0.0   0.0

3. -0.850001 -0.850001 -0.850001







To augment the data set during the training process , I also flipped images horizontally and also changed the steering angle accordingly.
I also introduced random brightness change in images to to generalize the model to different condition.

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 were 10 which was experimentally determined. I used an adam optimizer so that manually training the learning rate wasn't necessary.

OBSERVATION AND Future Work:

1. The most time in the project was consumed in training the data uploading the data to ec2 instance.
2. Use of track 2 data didn't help generalize the model , which was surprising and further analysis needs to be done in this field.
3. Use of Kears made live easier.
4. the NVDIA paper input the image is fed in YUV color space which need to be experimented with.
5. Other network architecture like comma.ai can also be tested or incorporated to improve the accuracies.