#Behavioral Cloning
##Writeup Template
###You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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
###Model Architecture and Training Strategy
####1. Solution Design Approach

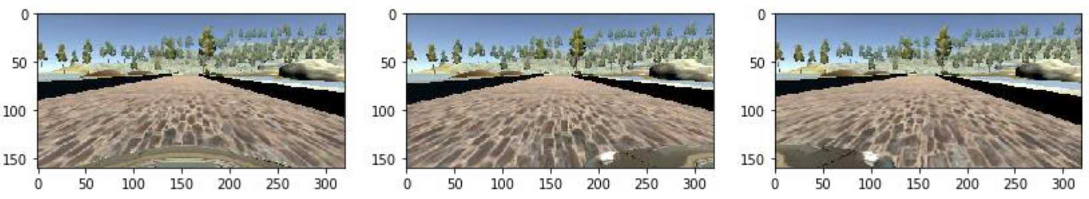
This project is entirely dependent on two main factors
How well the network architecture is defined How good the model has been trained (with a considerable number of images including data augmentation) I have faced many challenges in carrying out this project. Since the errors or issues that I was facing is quite entirely new to me, I did not receive proper support from mentor and tried to post my errors in slack community, but got less response. I have to thank personally to fellow nano-degree student Mr.Joao Sauso Pinto for helping me to solve the unnecessary tool/AWS/command errors and at least run my model.

The recorded data has the car driving in the center, several random recordings, car driving in reverse direction (one lap). I tried with a lot of images (to be specifi 80K). In total, for the project submission, I am running with 38568 images including equally flipped images of all cameras.

Data visualization:
This section can be found in Notebook.ipynb

1. Showing center, left, right sample images

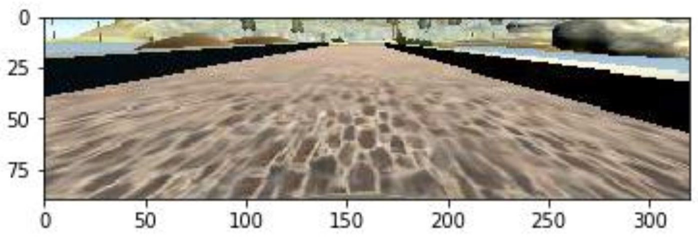
```
Out[17]: <matplotlib.image.AxesImage at 0x7f3e38195fd0>
```



2. Cropping images (taken sample as center image here)

```
In [20]: #Cropping images
         plt.imshow(SampleImg[50:140,:,:])  #To remove sky and hood part

Out[20]: <matplotlib.image.AxesImage at 0x7f3e380c76a0>
```

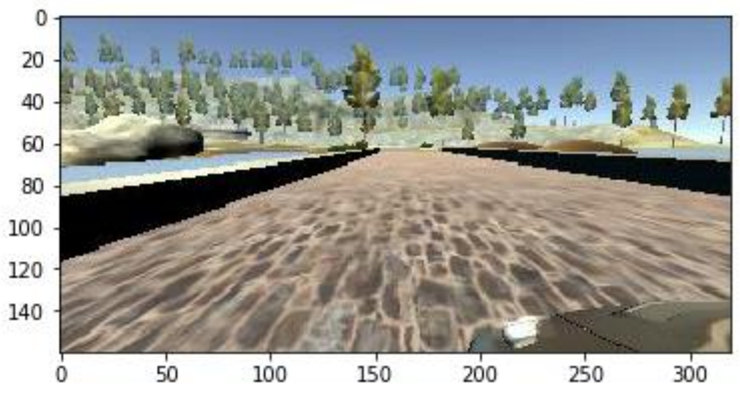


3. Flipping images (right image flipped to see as left image as example)

```
In [21]: #Flipping images
         print('Flipped image')
         plt.imshow(np.fliplr(rightImg))

         Flipped image

Out[21]: <matplotlib.image.AxesImage at 0x7f3e380a95f8>
```



I do not know the reason why the simulator is not running in Autonomous mode to test my trained model. Spent a lot of time on finding the reason, but unfortunately did not succeed. The simulator hangs up when I run the command "python drive.py model.h5"
But with this network architecture, it is the best what I could achieve now. But played around with steering angle values. Finally settled with 0.35.

Corrections made:
This time I trained with solid 358602 images with steering correction of 0.2. But sometimes with higher speeds, its going off the track. So to achieve this project, I set the speed to 7mph in the drive.py file. It depends entirely how the model sees the image at that time! Though it goes very close to the end of road at times, but recovers back to center of the road again after few seconds. Now car runs perfect! Stays on the track all the time.

Files included :
- Notebook.ipynb
- model.h5
- writeup.pdf
- drive.py (set speed to 7mph)
- video.py (python file provided by Udacity)
- ProjectVideo.mp4

To sum up the architecture:

The input image is (160,320,3)
The network architecture is defined as follows :
* A lambda layer for normalization
* Cropping layer to remove hood and sky part
* Convolution layer (24 filter depth, 5x5 kernel, 2x2 stride)
* Convolution layer (36 filter depth, 5x5 kernel, 2x2 stride)
* Convolution layer (48 filter depth, 5x5 kernel, 2x2 stride)
* Convolution layer (64 filter depth, 3x3 kernel, 2x2 stride)
* Convolution layer (64 filter depth, 3x3 kernel, 2x2 stride)
* Flatten layer
* Dropout (0.5)
* Dense layer (100 output units)
* Dense layer (50 output units)
* Dense layer (10 output units)
* Dense layer (1 output units)
* Steering outputs

I also flip images which I do it in generator_data. The ADAM optimizer is used. I tried with Nvidia architecture in mind but wanted to run at least one successful build with minimum error. Later did some changes to the architecture of Nvidia net.

Problems :
* AWS instance access took a lot of time to get resolved
* Meantime when I run the model in my PC (which do not have high graphics processor), I had to reduce the data set to 12K. For such image data set, it took a day to run. So, I managed to conclude with the best out of me in finally running it on AWS instance after resolving the issues associated with AWS(near to the end of project deadline. In the end, the project aimed at defining the network architecture and how much good and number of image data set that it is trained on.

Problem : I train the model in AWS. Download it to my PC. Run the command "python drive.py model.h5". Everything works fine. waiting for simulator to run. I run the simulator. I select the display and resolution. Then select Autonomous mode. Simulator hangs. It has been difficult to complete the project with available resources, I have submitted with my best of what I could achieve.