Behavioral Cloning Project

Introduction:

There is two approaches to make cars drive autonomously :

first, robotic approach by using methods for localization,mapping,path planning and control. second,using deep learning.

The goal of this project is to use deep learning specially the power of convolution networks to learn the car how to drive autonomously.

By getting data from the simulator we got :

Input : images come from 3 cameras (center,left and right),Each image represent a situation on the road .

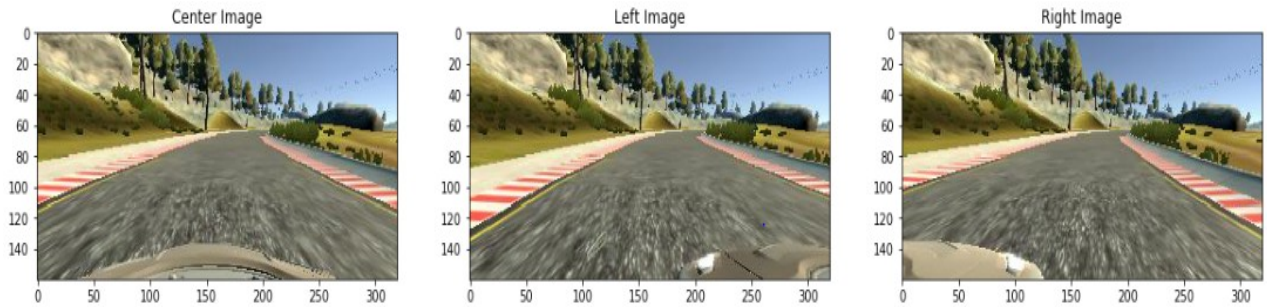
output : steering angel which represent the car behavior at each image.

Using keras framework we can build a model that can mimic our behavior in driving and make the car learn from it.

Model Architecture and Training Strategy:

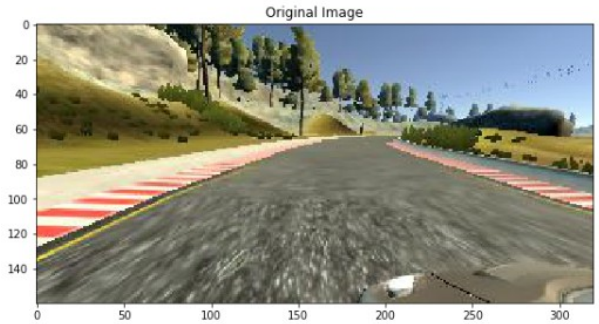
1- Loading Data

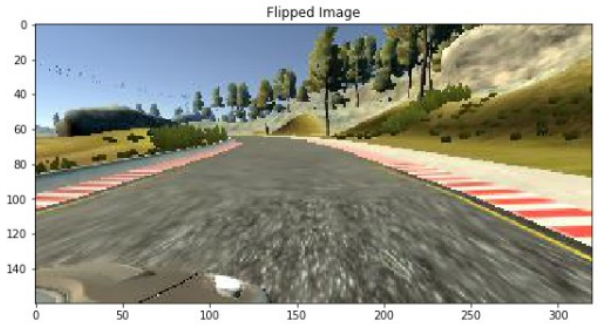
- I used the the data-set provided by Udacity.
- I used `matplotlib.image` to load the images in `RGB` as in `drive.py` it is processed in `RGB` format.
- Because we want the model to deal with left and right images as they were taken from the center .I decided to introduce a `correction factor` of `0.2`
- For the `left images` I `increase` the `steering angle` by `0.2` and for the `right` `images` I `decrease` the `steering angle` by `0.2`



2- Preprocessing

- Shuffling the images.
- Image augmentation (flipping image horizontally)

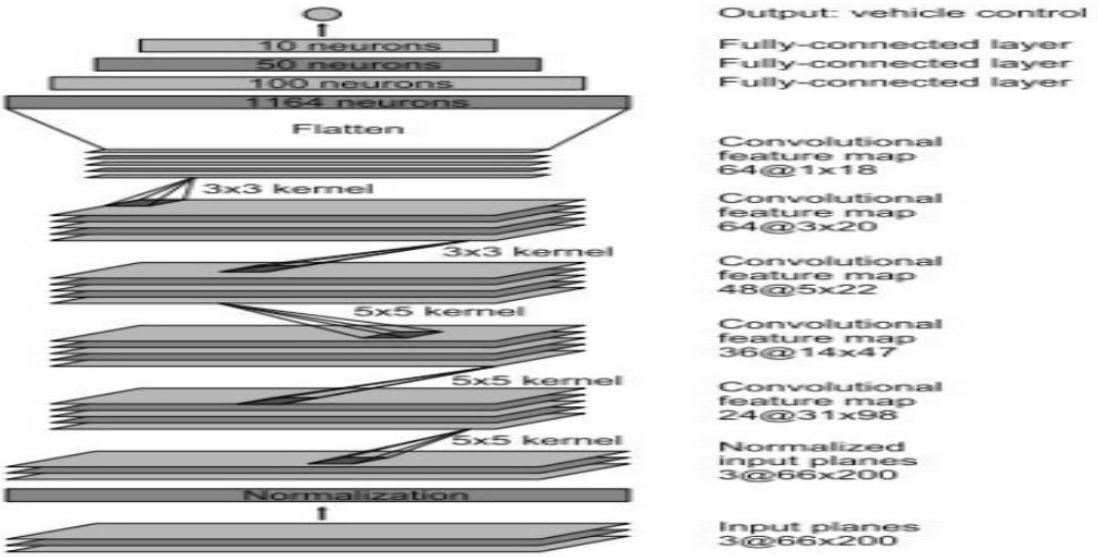




- Image cropping to reduce unnecessary parts of the image.

3- Model Architecture

I've decided to use the model provided by NVIDIA as suggested by Udacity. The model architecture is described by NVIDIA consists of :



- Normalization layer.
- First conv layer with num of filters = 24 and filter size =(5,5) with stride=(2,2) followed by relu activation.
- Second conv layer with num of filters = 36 and filter size =(5,5) with stride=(2,2) followed by relu activation.
- Third conv layer with num of filters = 48 and filter size =(5,5) with stride=(2,2) followed by relu activation.

- Two conv layer with num of filters = 64 and filter size =(3,3) with stride=(1,1) followed by relu activation.
- Flatten the feature map to prepare it for the fully connected layers.
- apply first fully connected layer with 100 neurons.
- Apply second fully connected layer with 50 neurons.
- Apply third fully connected layer with 10 neurons.
- And finally the layer with one output.

4-Model parameter tuning

- Optimizer : Adam
- Loss Function : MSE
- Learning Rate : 0.001
- No of epochs : 5
- Batch size= 32
- Validation Data split : 0.2
- Correction factor : 0.2

Conclusion:

- Using deep learning to teach cars how to drive will depend basically with the quality and the quantity of data we know will know if these criteria achieved by answering :

  - Does our data represent a good drive ?

  - Does our data cover all different situations that could happen while driving ?