Readme for the Self-Driving Cars Nanodegree Program 4th Project Behavioral Cloning

Goals / Steps of this project:

- Use the simulator to collect data of good driving behavior.
- Build, a convolution neural network with 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

Required Files & Quality of Code

Required Files

- model.py – Contains the script to create/load, train and validate the model.
- drive.py – Script used for driving the car in autonomous mode.
- model.h5 – Contains the trained CNN (Convolution Neural Network)
- writeup_report.pdf – File that summarizes the results.
- Video_lap.mp4 – Video that shows the vehicle driving 1 lap of the track #1.

Functional Code

The model provided can be used to drive autonomously the car in the simulator. To run the model use:

```
python drive.py model.h5
```

Usable and readable Code

The mode.py file contains the code used for training, validating, and saving the CNN. The code is commented to "learn" how to set some variables to create/load a model and select the folder with the dataset.

Model Architecture and Training Strategy

Model Architecture

The model used for the project is an NVIDIA architecture shown during the course.
It consists of:

- Normalization & Pre-process
    - Lines 64-65
    - I performed a normalization and a cropping of the images to eliminate zones that are not relevant for the model. Data is normalized using Keras lambda layer.
- 5 Convolutional Layers
    - Lines 66-70

    - Relu layers to introduce nonlinearity.
    - Kernel Sizes vary between 3x3 and 5x5.
    - Depths vary from 24 and 64.
- 5 Fully Connected Layers
    - Lines 71-78

Reduce Overfitting

I tried to use Dropouts as a method to reduce overfitting, after some tests I decided to eliminate the Dropouts since the behaviour of my model was better.
I used some Dropouts only on the fully connected layers since after some research I learned Dropouts on Convolutions does not help much.

Model Parameter Tuning

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

- Line 81

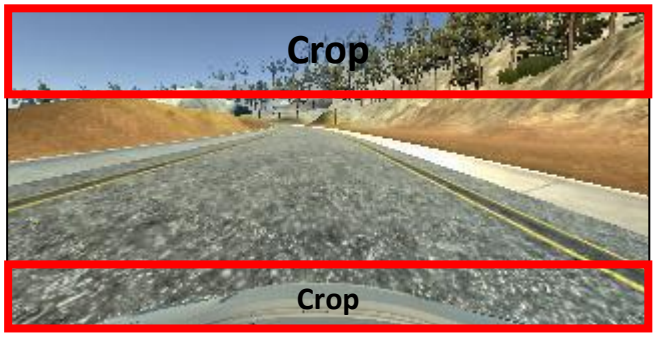
Appropriate Data Training

I decided to use the example data provided by Udacity since I was planning to save some GPU time for the next projects.
Even though I used the example data I had to create some by my own; I got data from the simulator by driving the car on the closest curves (left and Right) and recovering from the right side of the road.

Solution Design

The first step I took to approach this project was to follow same steps that were shown during the Project preparation by David Silver.

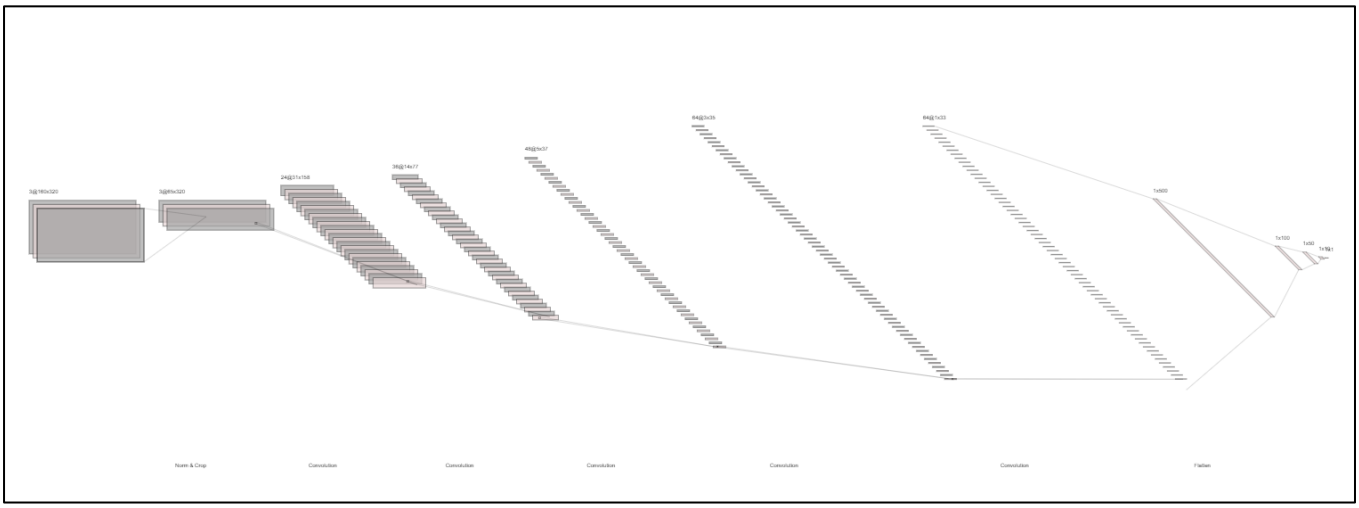
- I analysed the images that can be outputted by the simulator and perform a normalization ((x/255)-0.5) and a cropping (70 lines of pixels on the top and 25 on the bottom) of the image to eliminate the zones that were not relevant for the model (Sky, front of the car, etc).



- I selected a “base” architecture to create. For this application, the selected network architecture was the one used by the autonomous vehicle team at NVIDIA. Showed during the course.

- The Architecture of the model is the following:

Layer	Description
Input	RGB Image 160x320x3
Normalize & Crop	RGB Image 65x320x3
Convolution	Stride: 2x2, Padding: Valid, Output: 31x158x24
RELU	
Convolution	Stride: 2x2, Padding: Valid, Output: 14x77x36
RELU	
Convolution	Stride: 2x2, Padding: Valid, Output: 5x37x48
RELU	
Convolution	Stride: 1x1, Padding: Valid, Output: 3x35x64
RELU	
Convolution	Stride: 1x1, Padding: Valid, Output: 1x33x64
RELU	
Flatten	Output: 2,112
Fully-Connected	Output: 100
Dropout	
Fully-Connected	Output: 50
Dropout	
Fully-Connected	Output: 10
Dropout	
Fully-Connected	Output: 1



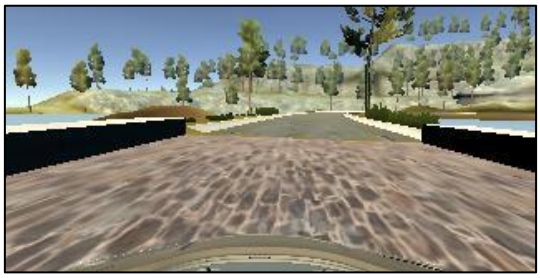
- Once the Pre-process was defined and the Architecture was selected and modified, I perform the first training with the dataset provided in the project.
    - I shuffle the data and split it in training data (80%) and validation data (20%).
    - I performed 3 epochs. (I tried with around 10 the first time but the error started to increase after the 3rd)

- After the first training the behaviour of the car in the simulator was pretty good, but it was having a hard time in close curves, so I decided to get some more data;

I prepared a data set for close curves to the right, close curves to the left and recovering for the right.

- I trained the model with the data obtain from turning right. 3 epochs.

Some images taken from the data set are shown below:



NOTE:
To drive the car in the simulator I use a Switch Controller connected and mapped in the PC. This helped a lot since controlling the car with the arrows is not really simple.

Simulation
The car can navigate correctly on the track without leaving the road. I attached a video as an example to this delivery. Video_lap.mp4.