Behavioral Cloning

Writeup Template

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

Rubric Points

Here I will consider the [rubric points](https://review.udacity.com/#!/rubrics/432/view) individually and describe how I addressed each point in my implementation.

Files Submitted & Code Quality

1. Submission includes all required files and can be used to run the simulator in autonomous mode

My project consists the following files:
* train_Rober_v08.py containing the script to create and train the model
* drive.py for driving the car in autonomous mode
* model-p3-v08.h5 containing a trained convolution neural network
* writeup_report.pdf summarizing the results

2. Submission includes functional code

Using the Udacity provided simulator and my drive.py file, the car can be driven autonomously around the **easy** track by executing.

```
*python drive.py model-p3-v08.h5*
```

3. Submission code is usable and readable

The train_Rober_v08.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 includes comments to explain how the code works.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

My model implements the model presented by NVIDIA in the paper 'End to End Learning for Self-Driving Cars'

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda_3 (Lambda)            (None, 160, 320, 3)       0
_________________________________________________________________
cropping2d_3 (Cropping2D)    (None, 90, 320, 3)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 43, 158, 24)       1824
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 20, 77, 36)        21636
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 8, 37, 48)         43248
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 6, 35, 64)         27712
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 4, 33, 64)         36928
_________________________________________________________________
flatten_3 (Flatten)          (None, 8448)              0
_________________________________________________________________
dense_9 (Dense)              (None, 100)               844900
_________________________________________________________________
dense_10 (Dense)             (None, 50)                5050
_________________________________________________________________
dense_11 (Dense)             (None, 10)                510
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 11
=================================================================
Total params: 981,819.0
Trainable params: 981,819.0
Non-trainable params: 0.0
```

2. Attempts to reduce overfitting in the model

The model NOT contains dropout layers to reduce overfitting.

The model was trained and validated on random data sets to ensure that the model was not overfitting using **ImageDataGenerator** from Keras.

The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

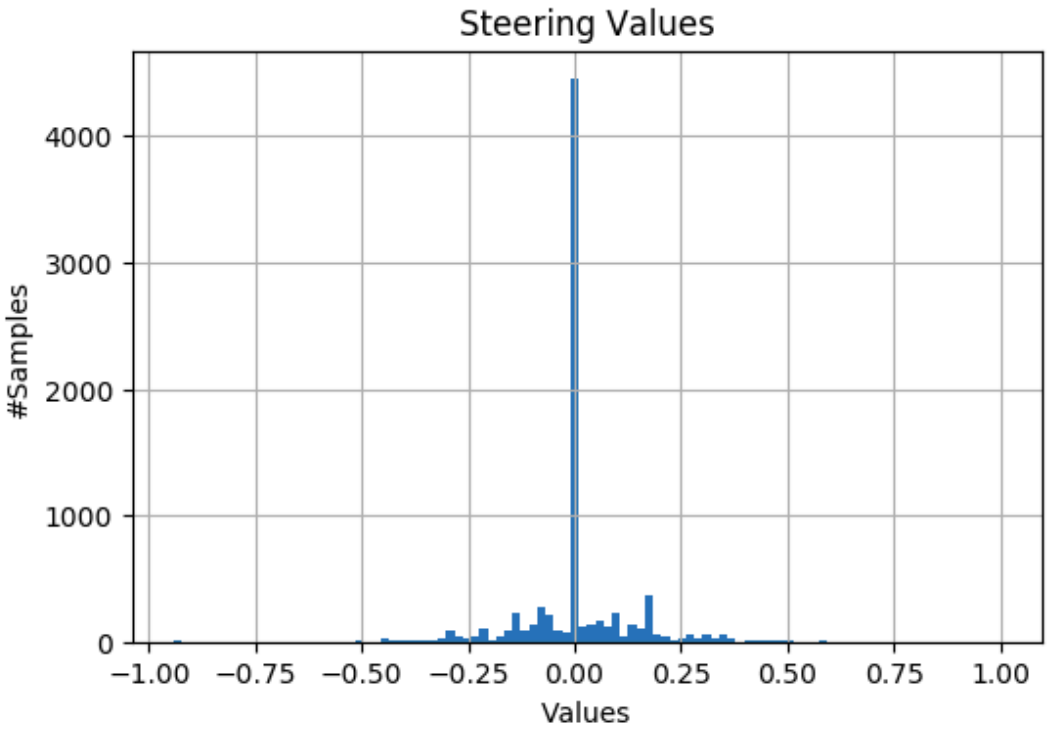
In a second phase, I try to train the same model with more data from left and right cameras. But I get worse results.

3. Model parameter tuning

The model used Adam optimizer, so the learning rate was not tuned manually

4. Visualizing training data

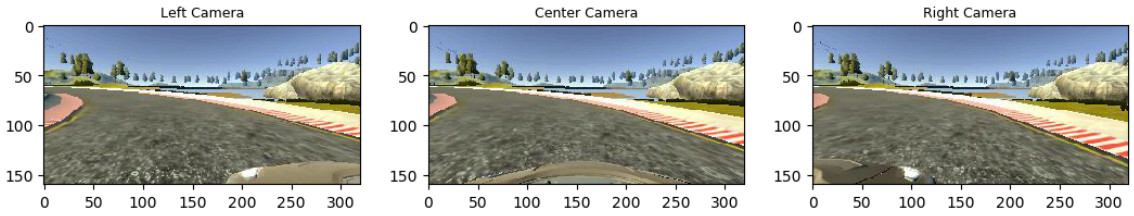
The original dataset is very imbalanced with many values near to zero from the right.




```
(8036,)
```

The images are clear, but we are sending to the model not useful info.

From the tutorial:
“Not all of these pixels contain useful information, however. In the image above, the top portion of the image captures trees and hills and sky, and the bottom portion of the image captures the hood of the car.”



For details about how I created the training data, see the next section.

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving the model architecture was to follow the steps of tutorial.

My first step was to use the CNN network form Project: “CarND-Traffic-Sign-Classifier-Project”. I thought this model might be appropriate because it works very well in the last project.

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set.

To combat the overfitting, I use **ImageDataGenerator** from Keras

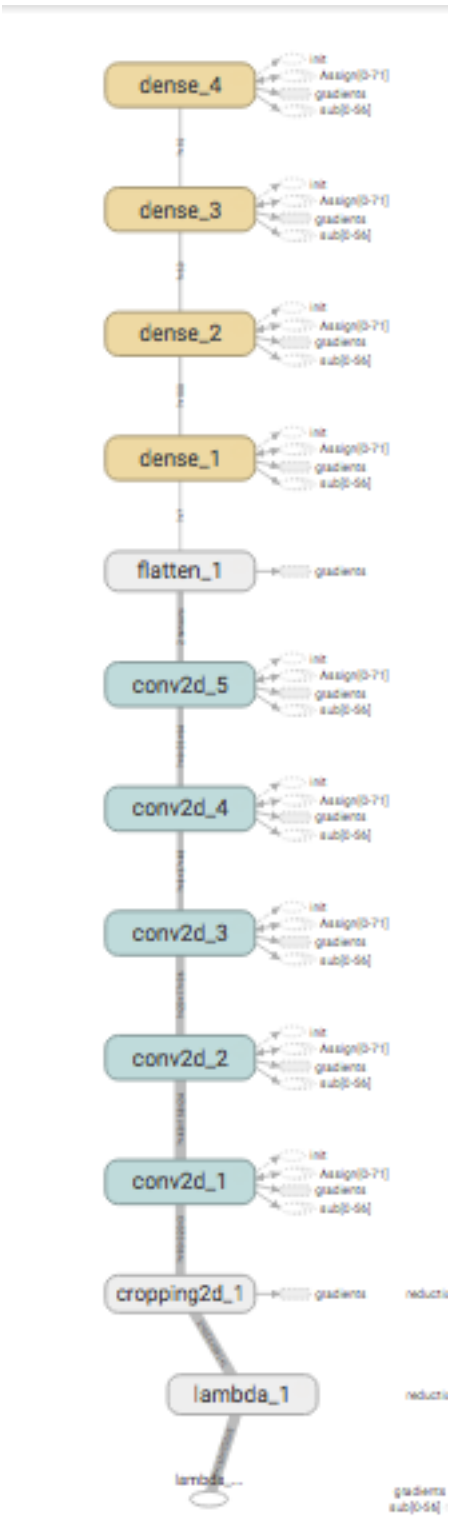
The final step was to run the simulator to see how well the car was driving around track one.

At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

My model implements the model presented by NVIDIA in the paper ‘End to End Learning for Self-Driving Cars’

Here is a visualization of the architecture from Tensorboard:

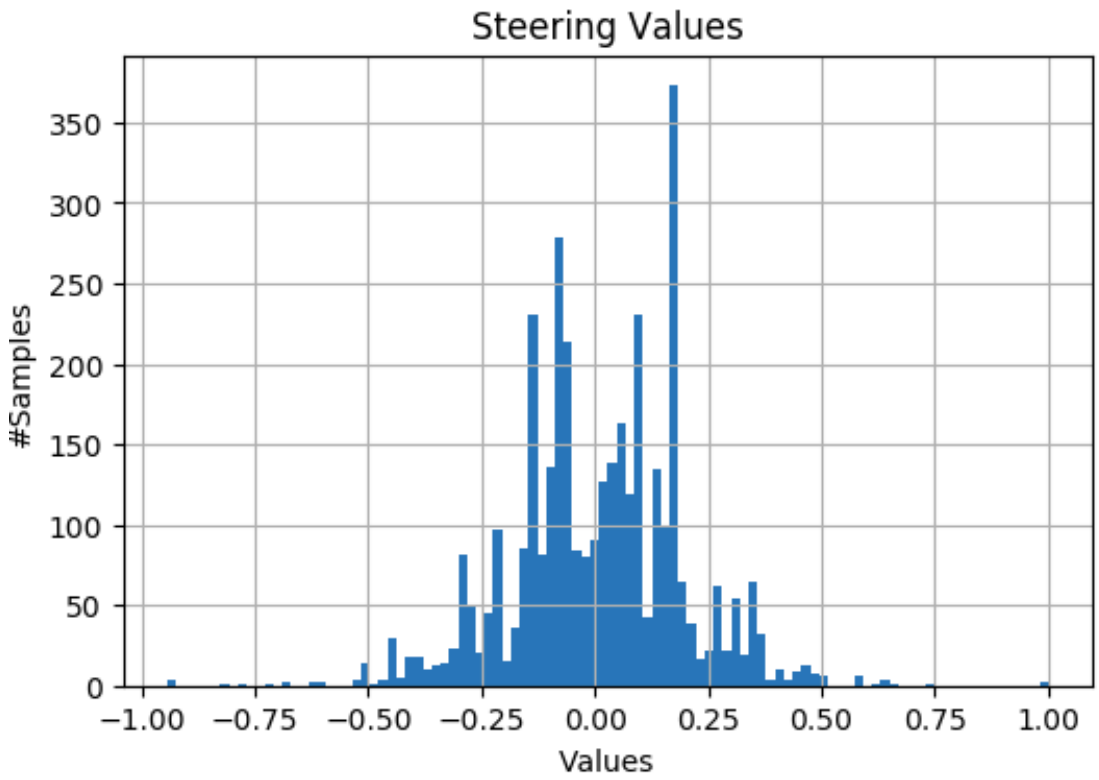

dense_4
dense_3
dense_2
dense_1
flatten_1
conv2d_5
conv2d_4
conv2d_3
conv2d_2
conv2d_1
cropping2d_1
lambda_1

Layer (type)	Output Shape	Param #
lambda_3 (Lambda)	(None, 160, 320, 3)	0
cropping2d_3 (Cropping2D)	(None, 90, 320, 3)	0
conv2d_11 (Conv2D)	(None, 43, 158, 24)	1824
conv2d_12 (Conv2D)	(None, 20, 77, 36)	21636
conv2d_13 (Conv2D)	(None, 8, 37, 48)	43248
conv2d_14 (Conv2D)	(None, 6, 35, 64)	27712
conv2d_15 (Conv2D)	(None, 4, 33, 64)	36928
flatten_3 (Flatten)	(None, 8448)	0
dense_9 (Dense)	(None, 100)	844900
dense_10 (Dense)	(None, 50)	5050
dense_11 (Dense)	(None, 10)	510
dense_12 (Dense)	(None, 1)	11

Total params: 981,819.0
Trainable params: 981,819.0
Non-trainable params: 0.0

3. Creation of the Training Set & Training Process

I use the default data set.
I balance the original dataset.
I remove samples with values minor or equal to abs(0.003).



Data augmentation:

A: Before of training, I flip the images:

a. Flip horizontal
b. Flip vertical

B: During training time, I use **ImageDataGenerator** from Keras With theses parameters:

```
datagen = ImageDataGenerator(
        samplewise_center=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.1,
        zoom_range=0.1,
        fill_mode='nearest',
        horizontal_flip=False,
        vertical_flip=False,
        rescale=1.)
```

Refs:

#https://keras.io/preprocessing/image/
#https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
#ZCA whitening
#http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

I randomly shuffled the data set and put 20% of the data into a validation set.

After the collection process, I had these number of data points:
Number of training examples = 9709
Number of valid examples = 1079

Training Process:

During training, I process the images in the first layers of the model:
+ Image normalization: Using **Lambda** Layer from Keras.
+ Image cropping: Using **Cropping2D** Layer from Keras.

I normalize and crop the image to use only the portion of the image that is useful for predicting a steering angle.

I use augmented training data and Keras Generator.
The keras generator produces batchs of train & validation data with little transformations.

I use **Adam** optimizer so that manually training the learning rate wasn't necessary.

I set epoch to 50 and to stop the train and I use **EarlyStopping** from Keras.
To save the best model during training, I use **ModelCheckpoint** from Keras.