#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

Rubric Points

###Here I will consider the rubric points 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 includes the following files:

- model.py containing the script to create and train the model
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- writeup_report.md or 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 track by executing

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

####3. Submission code is usable and readable

The model.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 contains comments to explain how the code works.

###Model Architecture and Training Strategy

####1. An appropriate model architecture has been employed

I have used nvidia's approach (end-to-end self driving car paper)

This architecture have layers as follows:

- input Plan 3@66x200
- input plan normalization
- conv2D 24 feature map
- conv2D 36 featuremap
- conv2D 48 feature map
- conv2D 64 feature map
- conv2D 64 feature map
- full connected 100
- full connected 50
- full connected 10

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

The model contains dropout layers in order to reduce overfitting

####3. Model parameter tuning

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

But I have set the curve and straight steering ratio as 0.5.

If I did not set the curve image not much, then training have bad result (It is fit to only striaght line)

####4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left and right sides of the road ...

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

I have augmented image as follows:

- translate image
- use left and right camera image
- change brightness
- flip image

###Model Architecture and Training Strategy

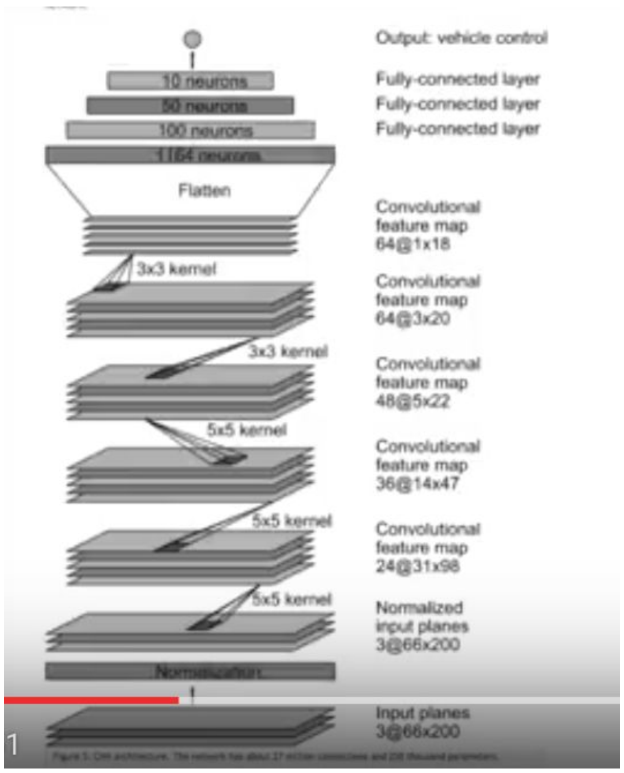
####1. Solution Design Approach

I have used nvidia's approach (end-to-end self driving car paper)

When I augmented the image by translation, I have set 0.1degree per pixel. But this is now good for curve when I have tested the simulator. So, I have changed it to 0.2 degree per pixel.

####2. Final Model Architecture

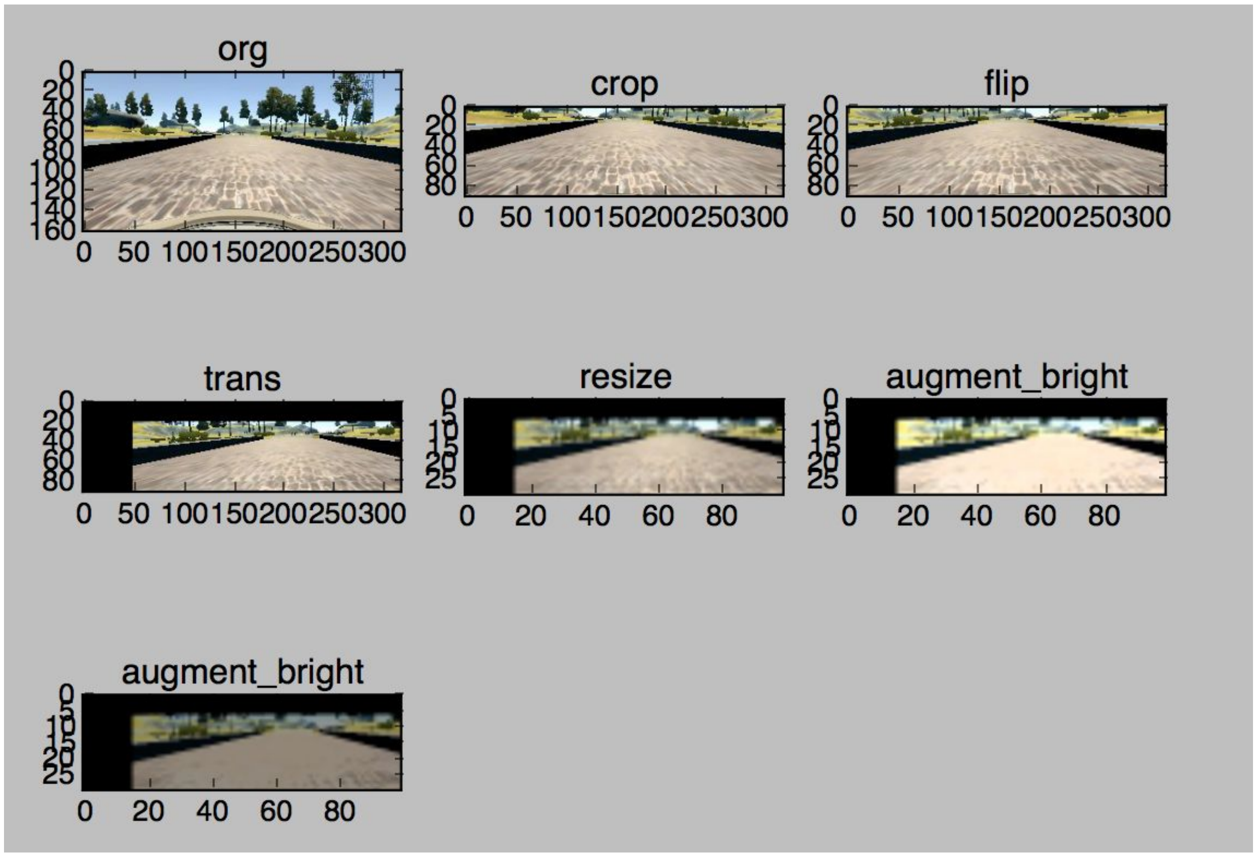
The final model architecture consisted of a convolution neural network with the following layers and layer sizes ...



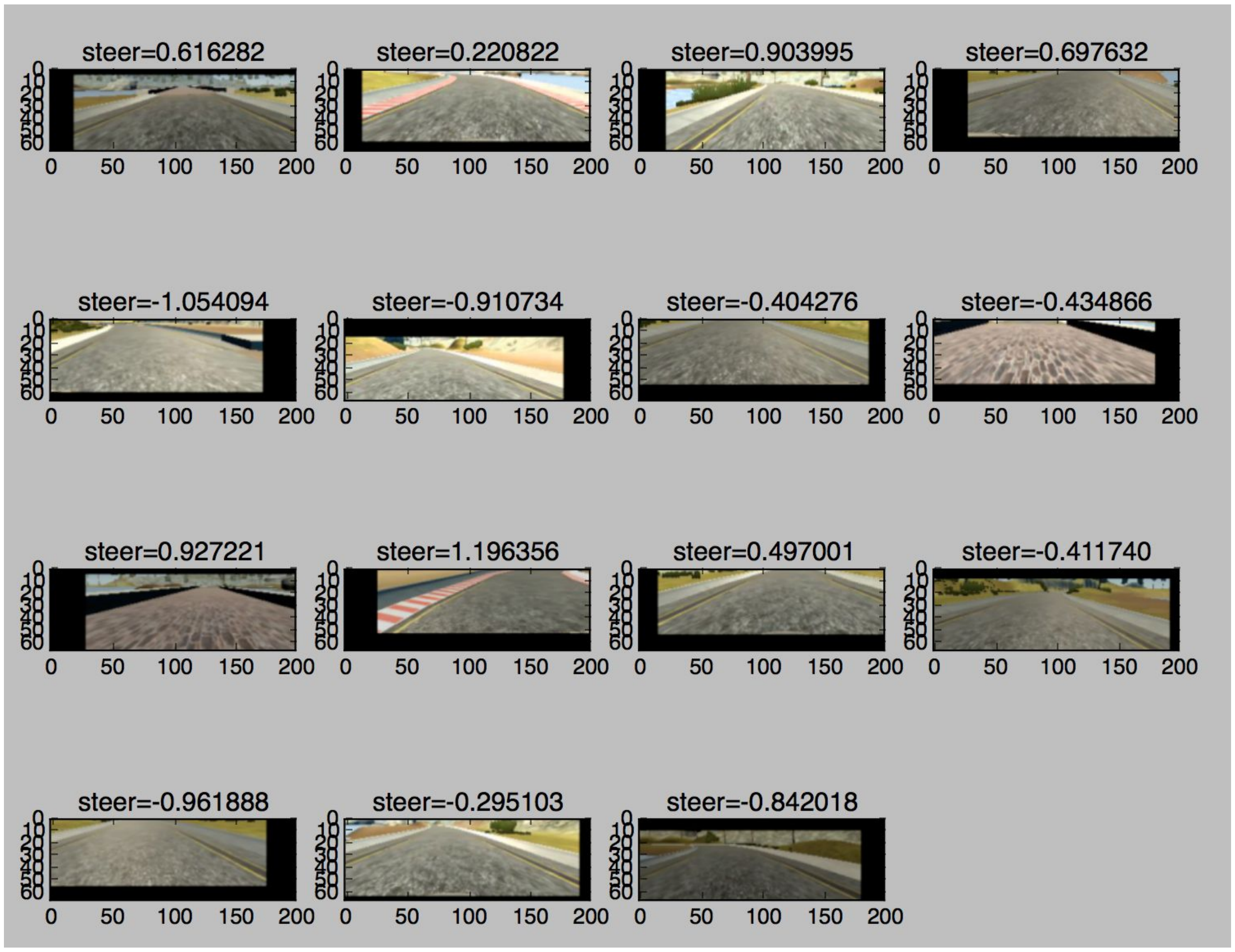
####3. Creation of the Training Set & Training Process

I have trained with image and steering data presented by udacity. I have not made my data on my own.

I have generated augmented images with as many curves (It has 0.5 probability)
I have tested 10 epochs with 25600 images per epoch.
Each image is augmented randomly.
I have augmented images as follows:



Following image is sample images when I trained :


steer=0.616282
steer=0.220822
steer=0.903995
steer=0.697632
steer=-1.054094
steer=-0.910734
steer=-0.404276
steer=-0.434866
steer=0.927221
steer=1.196356
steer=0.497001
steer=-0.411740
steer=-0.961888
steer=-0.295103
steer=-0.842018