Project 3: Using Deep Learning to Clone Driving Behavior

Behavior 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

Overview

The goal of this project is to learn the good driving behavior of human driver using a simulator provided by Udacity and use the learnt data in autonomously driving the car with the same simulator.

The project helped in understanding the usage of Keras functional APIs, how simple it can be used in deep learning. Also introduces to concepts of avoiding overfitting the images, data augmentation, how to collect data in real environment, and making recovery data to the stable environment (e.g. starting from edge of the lane to center).

Finally using fit generator, visualizing loss metrics, also how to maximize the usage of memory using Generators/yield routine.

Files Submitted & Code Quality

Required Files:

####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.pdf summarizing the results
- video.mp4 the recorded video of the automonous driving

####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

The drive.py code was developed on a Windows machine with Nvidia Quadro M4000 GPU. With the Epoch of 10, every iteration ran for 3-4 minutes, overall every run took me less than 40 minutes. Earlier there were bunch of issues in setting CUDA and running in the Conda environment, took almost a week to get the GPU environment running.

Also, ran this code on my MAC with just CPU, one iteration was taking around 20-30 minutes. Spend almost a week in setting my GPU/Windows machine.

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

Used the same file name conventions and added appropriate comments, explanation in the documentation.

Model Architecture and Training Strategy

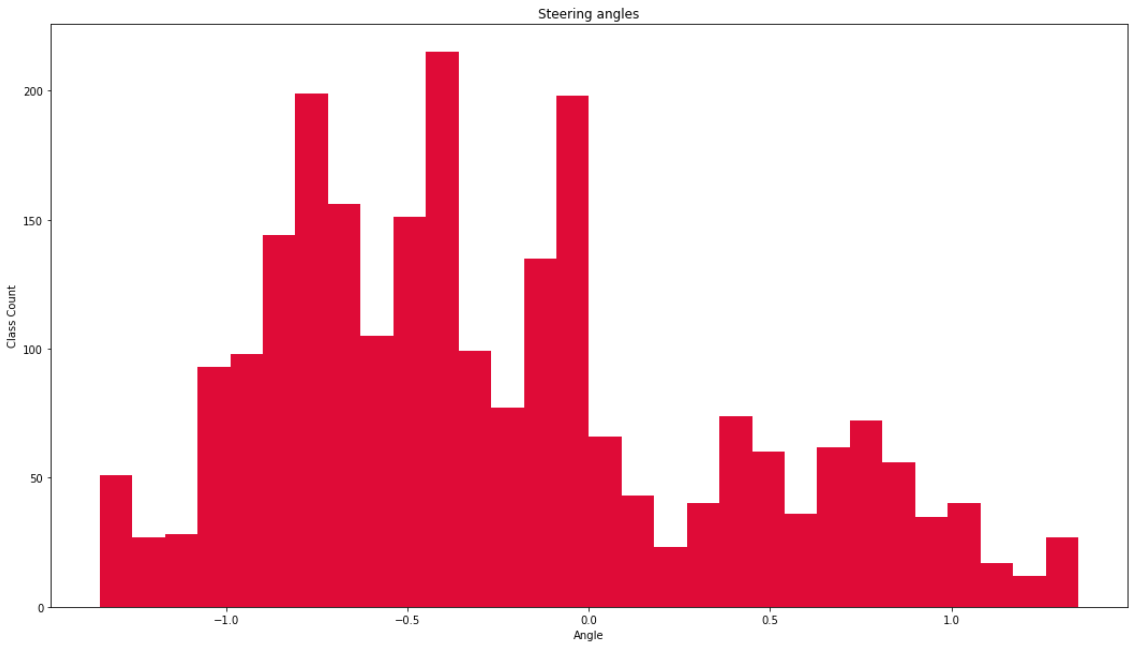
Sample simulation Data:

Here is the sample data from one of the run. The CSV was parsed correctly for st eering, throttle, speed, break, left, center, and right images.

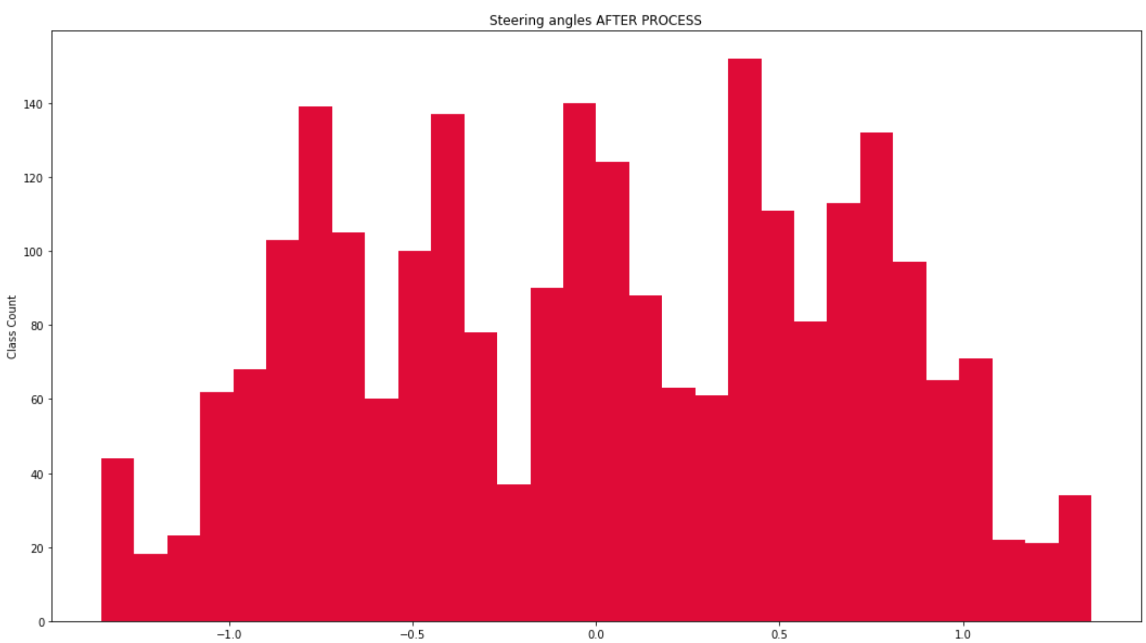

```
{'steering': ' 0', 'throttle': ' 0', 'speed': ' 7.83842E-05', 'brake
': ' 0', 'left': ' /Users/rthirumu/CarND-Behavioral-Cloning-P3/IMG/l
eft_2017_05_13_14_12_51_959.jpg', 'center': '/Users/rthirumu/CarND-B
ehavioral-Cloning-P3/IMG/center_2017_05_13_14_12_51_959.jpg', 'right
': ' /Users/rthirumu/CarND-Behavioral-Cloning-P3/IMG/right_2017_05_1
3_14_12_51_959.jpg'}
```

Steering Data Visualization:

This is the visualization of Steering angle data, this data is collected from the Udacity simulator, from one of the runs.



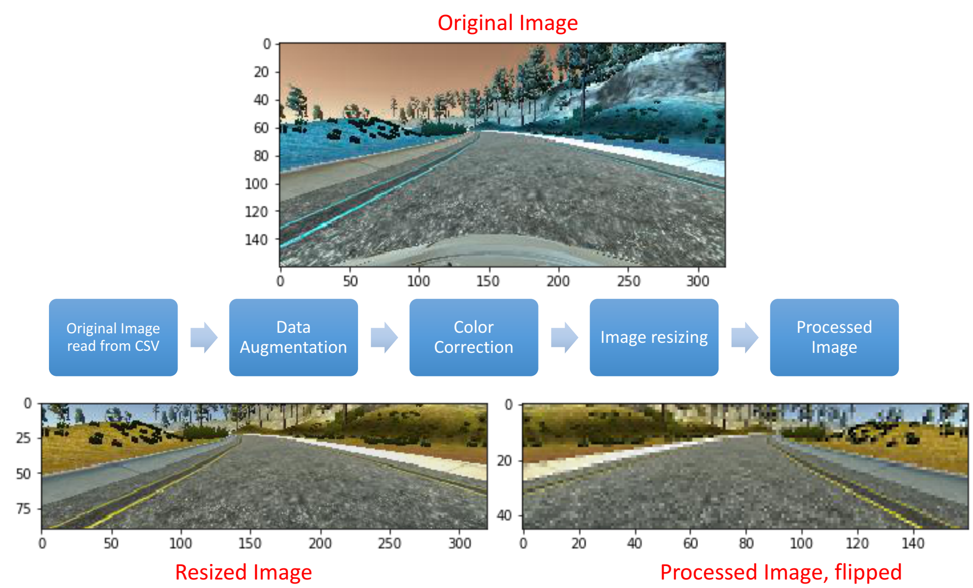
Steering Angle of processed data:



After the processing the input data, we can see the better data across the range.

Preprocessing:

Here are the various stages of the preprocessing of the simulator Image:



- **Original Image**
  The data files driving_log.csv and its image files are located in the IMG directory. The driving_log.csv is processed using the csv.Dictreader which returns the dictionary of the data.

```
Original Image dimensions: (160, 320, 3) steering 0.4194869

{'steering': ' 0', 'throttle': ' 0', 'speed': ' 7.83842E-05', 'brake
': ' 0', 'left': ' /Users/rthirumu/CarND-Behavioral-Cloning-P3/IMG/l
eft_2017_05_13_14_12_51_959.jpg', 'center': '/Users/rthirumu/CarND-B
ehavioral-Cloning-P3/IMG/center_2017_05_13_14_12_51_959.jpg', 'right
': ' /Users/rthirumu/CarND-Behavioral-Cloning-P3/IMG/right_2017_05_1
3_14_12_51_959.jpg'}
```

- **Data Augmentation**
  This implementation uses the images provided by all the 3 cameras (left, center and right). Used the augmentation technique in correcting the angle of the Left and Right Cameras. So that all the camera images can be used in the learning. Used .35 angle correction, tried using .15 to .35 delta. The

processed image is randomly flipped, in order to learn the left and right turns.

Reference: https://medium.com/@ksakmann/behavioral-cloning-make-a-car-drive-like-yourself-dc6021152713

- **Color Correction**
  Actual image is read using the cv2.imread API and required color conversation being done, to process in cv2 APIs. Also, the brightness was randomly adjusted.

- **Image Resizing**
  In order to reduce the memory usage, CPU/GPU cropped the image to half the height of the image. Cropped the height of the image to avoid the sky, tree tops and front bonnet of the cars. Still did not crop the side, the side of the image is still need for the learning. Useful for the cases, when the car goes to the edges of the lanes, bridges at different angles.

```
Original Image dimensions: (160, 320, 3) steering 0.4194869

Resized Image dimensions: (90, 320, 3) steering 0.4194869
```

Model Architecture and Solution

Model Architecture:

Used Comma AI research model **https://github.com/commaai/research**

This a popular Convolutional neural network by Comma AI, used to predict the steering angle of the simulator. This CNN takes input share 3 x 45 x 160. Each image got normalized in the range -1.0 to +1.0 . There are 3 convolutional layers conv2d_1, conv2d_2, conv2d_3 produces output of (1 x 12 x 16), (1 x 6 x 32), (1 x 3 x 64) respectively.

The layers are fully connected with ELU activations and as well dropout regularizations between the layers. The final layer provides the predicted streering angle for the image. Also, this model does not do pooling.

```
Layer (type)                 Output Shape              Param #
=================================================================
lambda_2 (Lambda)            (None, 3, 45, 160)        0
```

```
conv2d_1 (Conv2D)              (None, 1, 12, 16)         163856
_________________________________________________________________
elu_1 (ELU)                    (None, 1, 12, 16)         0
_________________________________________________________________
conv2d_2 (Conv2D)              (None, 1, 6, 32)          12832
_________________________________________________________________
elu_2 (ELU)                    (None, 1, 6, 32)          0
_________________________________________________________________
conv2d_3 (Conv2D)              (None, 1, 3, 64)          51264
_________________________________________________________________
flatten_1 (Flatten)            (None, 192)               0
_________________________________________________________________
dropout_1 (Dropout)            (None, 192)               0
_________________________________________________________________
elu_3 (ELU)                    (None, 192)               0
_________________________________________________________________
dense_1 (Dense)                (None, 512)               98816
_________________________________________________________________
dropout_2 (Dropout)            (None, 512)               0
_________________________________________________________________
elu_4 (ELU)                    (None, 512)               0
_________________________________________________________________
dense_2 (Dense)                (None, 1)                 513
=================================================================
Total params: 327,281
Trainable params: 327,281
Non-trainable params: 0
```

Model Parameters, Optimizer

Used Epoch as 20 and Batch size as 128. The loss were ranging from 2.6 % to 1.7%. It was decreasing from First Epoch to Last. But it stabilized around Epoch 7 and stayed at 1.7% . Tried Adam optimizer (.001) and .0001 during the fine tuning.

EPOCHS	20
BATCH_SIZE	128
Optimizer	Adam

The following output was collected running from a GPU machine:

2017-05-14 16:09:30.925824: I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\core\common_runtime\gpu\gpu_device.cc:977] Creating TensorFlow device (/gpu:0) -> (device: 0, name: Quadro M4000, pci bus id: 0000:02:00.0)
2649/2649 [==============================] - 223s - loss: 0.0263 - mean_squared_error: 0.0263 - val_loss: 0.1825 - val_mean_squared_error: 0.1825
Epoch 2/10

2649/2649 [==============================] - 218s - loss: 0.0215 - mean_squared_error: 0.0215 - val_loss: 0.1756 - val_mean_squared_error: 0.1756
Epoch 3/10
2649/2649 [==============================] - 218s - loss: 0.0207 - mean_squared_error: 0.0207 - val_loss: 0.1736 - val_mean_squared_error: 0.1736
Epoch 4/10
2649/2649 [==============================] - 218s - loss: 0.0195 - mean_squared_error: 0.0195 - val_loss: 0.1660 - val_mean_squared_error: 0.1660
Epoch 5/10
2649/2649 [==============================] - 217s - loss: 0.0192 - mean_squared_error: 0.0192 - val_loss: 0.1686 - val_mean_squared_error: 0.1686
Epoch 6/10
2649/2649 [==============================] - 217s - loss: 0.0184 - mean_squared_error: 0.0184 - val_loss: 0.1673 - val_mean_squared_error: 0.1673
Epoch 7/10
2649/2649 [==============================] - 217s - loss: 0.0178 - mean_squared_error: 0.0178 - val_loss: 0.1696 - val_mean_squared_error: 0.1696
Epoch 8/10
2649/2649 [==============================] - 216s - loss: 0.0179 - mean_squared_error: 0.0179 - val_loss: 0.1663 - val_mean_squared_error: 0.1663
Epoch 9/10
2649/2649 [==============================] - 215s - loss: 0.0179 - mean_squared_error: 0.0179 - val_loss: 0.1694 - val_mean_squared_error: 0.1694
Epoch 10/10
2649/2649 [==============================] - 215s - loss: 0.0178 - mean_squared_error: 0.0178 - val_loss: 0.1675 - val_mean_squared_error: 0.1675

(tensorflowgpu) C:\Users\rthirumu\udacity\CarND-Behavioral-Cloning-P3>

Conclusion, Future Work

- Was able use model.h5 in running the car in Autonomous mode. CommaAI model is used in this CNN learning.
- Able to get the Windows based CUDA/GPU setup ready after almost a week of effort, with the GPU machine able to run the maximum data I had collected around 40 minutes.
- Will try to use Nvidia network next to how efficient it fits with this project.
- Also will try different optimizations mechanism in generating different results.
- Overall, it was great experience in experimenting in this project. As a first time learner of ML/Deep learning the last couple of projects gave a great insight on the machine, how TensorFlow/Keras APIs evolved and GPU has become a ML sandbox.