Behavioral Cloning

Writeup

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

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

Basic Steps of the Project:

Download https://github.com/UDACITY/CarND-Behavioral-Cloning-P3

Download https://github.com/udacity/self-driving-car-sim

Get your record data from simulator on your local pc.

Adjust your record csv file columns

Write your model.py sw code

Enabled GPU

Generate and Save model.h5

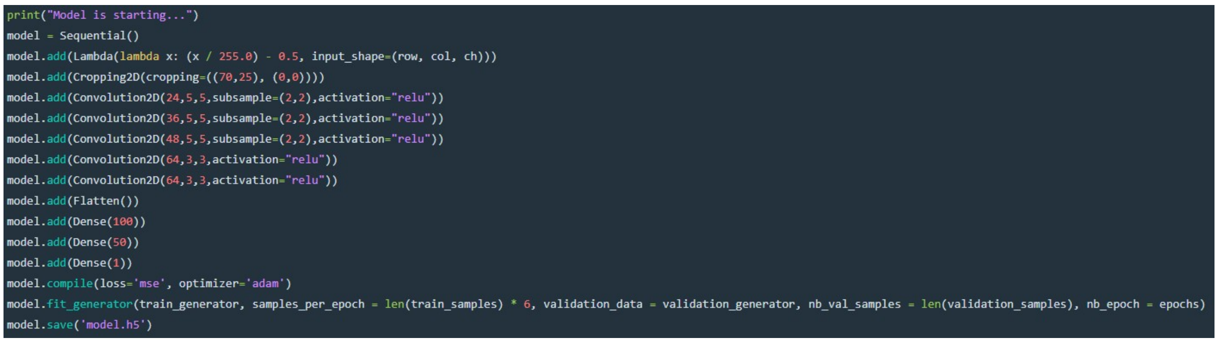
Run drive.py model.h5 run1

Run video.py run1

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

My model consists of a convolution neural network with 2x2 filter sizes and depths between 24 and 64.



```
print("Model is starting...")
model = Sequential()
model.add(Lambda(lambda x: (x / 255.0) - 0.5, input_shape=(row, col, ch)))
model.add(Cropping2D(cropping=((70,25), (0,0))))
model.add(Convolution2D(24,5,5,subsample=(2,2),activation="relu"))
model.add(Convolution2D(36,5,5,subsample=(2,2),activation="relu"))
model.add(Convolution2D(48,5,5,subsample=(2,2),activation="relu"))
model.add(Convolution2D(64,3,3,activation="relu"))
model.add(Convolution2D(64,3,3,activation="relu"))
model.add(Flatten())
model.add(Dense(100))
model.add(Dense(50))
model.add(Dense(1))
model.compile(loss='mse', optimizer='adam')
model.fit_generator(train_generator, samples_per_epoch = len(train_samples) * 6, validation_data = validation_generator, nb_val_samples = len(validation_samples), nb_epoch = epochs)
model.save('model.h5')
```

The model includes RELU layers to introduce nonlinearity and the data is normalized in the model using a Keras lambda layer.

2. Attempts to reduce overfitting in the model

The model was trained and validated on different data sets to ensure that the model was not overfitting.

3. Model parameter tuning

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

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 and repeated driving over more complex parts of track.

For details , I will explain it in the next parts.

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to start with the simple model, see how the data improves the behavior and then move to a more complex network.

My first step was to use a very simple network with one connected layer to test if the whole pipeline is working.

It was barely working with the small training data that I've collected. I then addded the augmentation of images by flipping horizontally which improved a bit. Adding left and right views helped a bit as well.

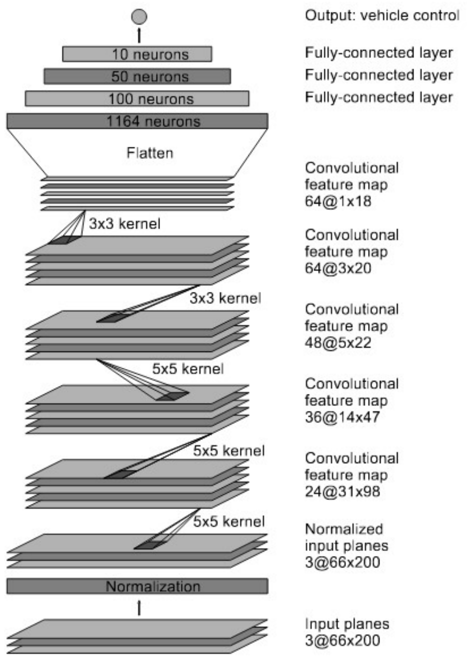
I've played around with number of epochs but this was not helping so clearly the next step was to move to a more complex network.

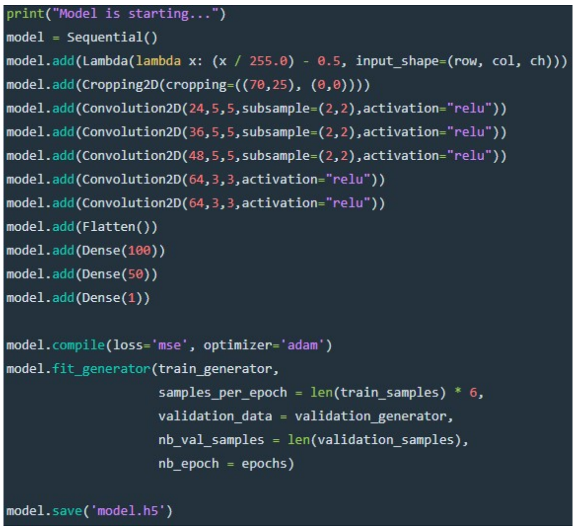
The NVIDIA network worked wonders and almost got the car to cover the whole track. The final step that made things click was finding the right parameter for steering offset for images from the left and right cameras.(0,4)

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

2. Final Model Architecture

The final model architecture consisted of a convolution neural network as found in the NVIDIA paper.



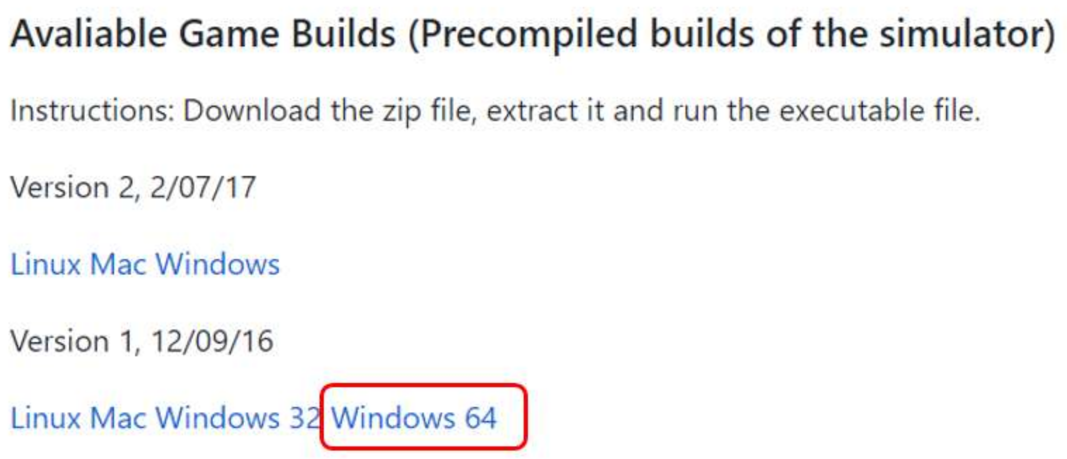


```
print("Model is starting...")
model = Sequential()
model.add(Lambda(lambda x: (x / 255.0) - 0.5, input_shape=(row, col, ch)))
model.add(Cropping2D(cropping=((70,25), (0,0))))
model.add(Convolution2D(24,5,5,subsample=(2,2),activation="relu"))
model.add(Convolution2D(36,5,5,subsample=(2,2),activation="relu"))
model.add(Convolution2D(48,5,5,subsample=(2,2),activation="relu"))
model.add(Convolution2D(64,3,3,activation="relu"))
model.add(Convolution2D(64,3,3,activation="relu"))
model.add(Flatten())
model.add(Dense(100))
model.add(Dense(50))
model.add(Dense(1))

model.compile(loss='mse', optimizer='adam')
model.fit_generator(train_generator,
                    samples_per_epoch = len(train_samples) * 6,
                    validation_data = validation_generator,
                    nb_val_samples = len(validation_samples),
                    nb_epoch = epochs)

model.save('model.h5')
```

3. 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.

I used Udacity simulator on my **local pc.** I got simulator exe from https://github.com/udacity/self-driving-car-sim



Avaliable Game Builds (Precompiled builds of the simulator)

Instructions: Download the zip file, extract it and run the executable file.

Version 2, 2/07/17

Linux Mac Windows

Version 1, 12/09/16

Linux Mac Windows 32 Windows 64

I used Windows 64 to simulate.

After Simulation finished I got these pictures from 3 cameras inside the car.

There are three cameras: center, left and right.

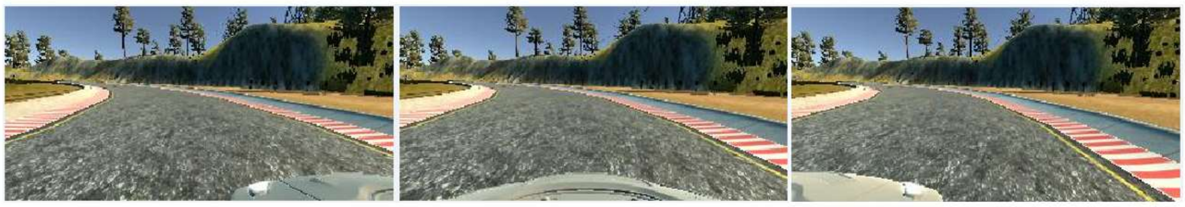
right_2019_01_07_17_19_44_667.jpg

left_2019_01_07_17_19_44_667.jpg

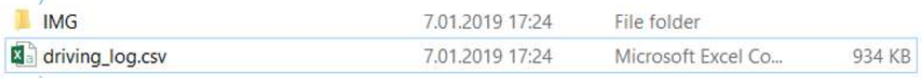
center_2019_01_07_17_19_44_667.jpg

Camera views

Left Center Right

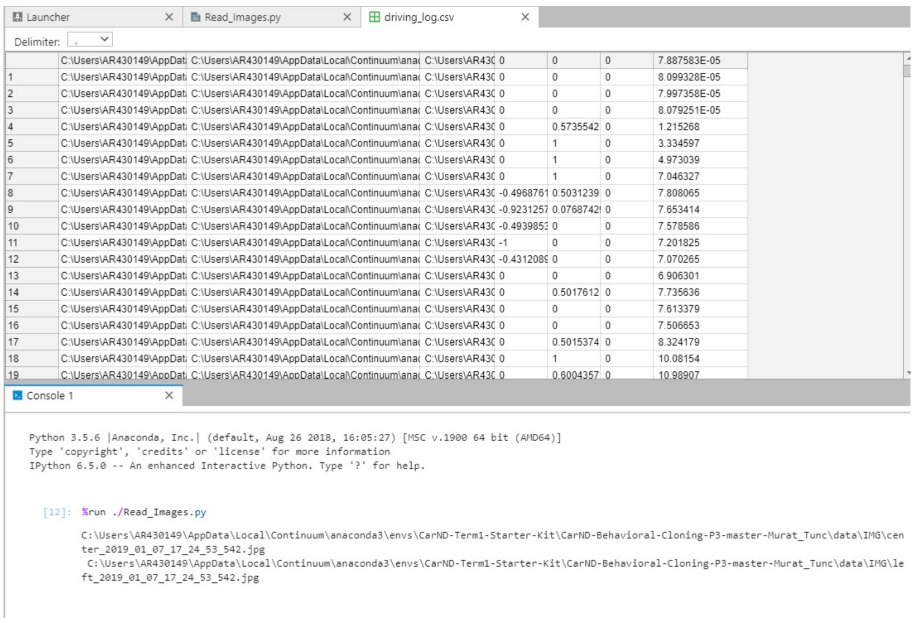


Center, Left, Right → images



IMG	7.01.2019 17:24	File folder	
driving_log.csv	7.01.2019 17:24	Microsoft Excel Co...	934 KB

Steering angle, Throttle, Break, Speed



Launcher | Read_Images.py | driving_log.csv

Delimiter: ,

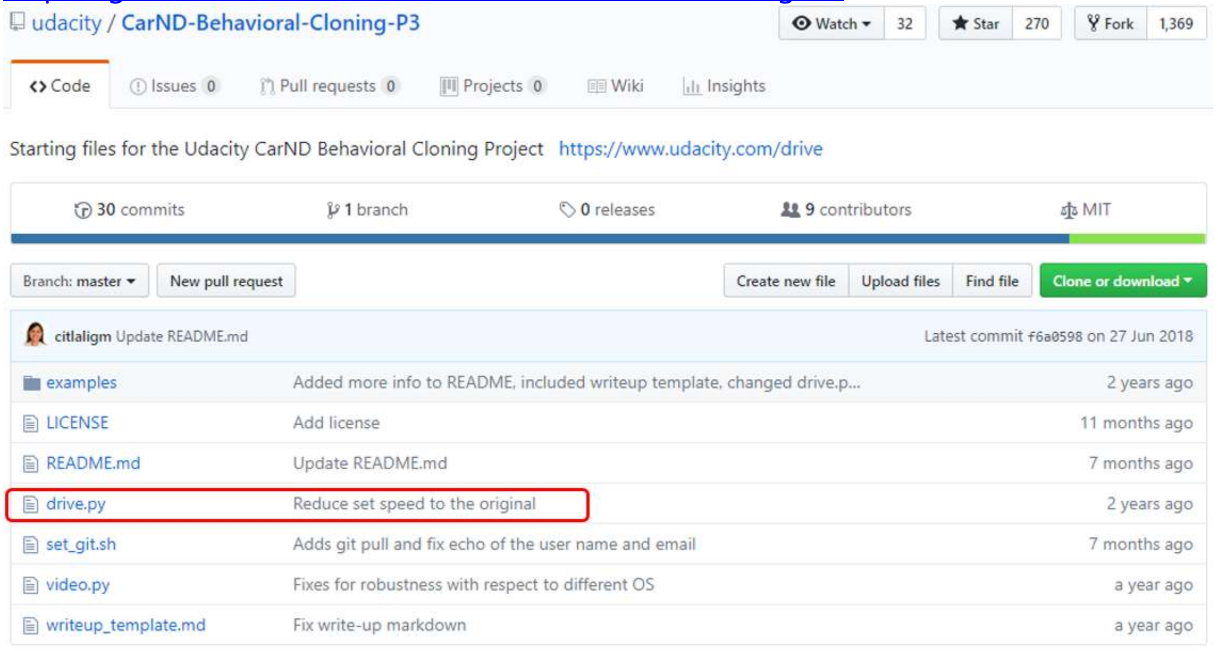
	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0	0	7.887583E-05
1	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0	0	8.099328E-05
2	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0	0	7.997358E-05
3	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0	0	8.079251E-05
4	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0.5735542	0	1.215268
5	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	1	0	3.334597
6	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	1	0	4.973039
7	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	1	0	7.046327
8	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	-0.4968761	0.5031239	0	7.808065
9	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	-0.9231257	0.0768742	0	7.653414
10	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	-0.4939853	0	0	7.578586
11	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	-1	0	0	7.201825
12	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	-0.4312089	0	0	7.070265
13	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0	0	6.906301
14	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0.5017612	0	7.735636
15	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0	0	7.613379
16	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0	0	7.506653
17	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0.5015374	0	8.324179
18	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	1	0	10.08154
19	C:\Users\AR430149\AppDat	C:\Users\AR430149\AppData\Local\Continuum\ana	C:\Users\AR430	0	0.6004357	0	10.98907

Console 1

```
Python 3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:05:27) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.5.0 -- An enhanced Interactive Python. Type '?' for help.


  [12]: %run ./Read_Images.py

        C:\Users\AR430149\AppData\Local\Continuum\anaconda3\envs\CarND-Term1-Starter-Kit\CarND-Behavioral-Cloning-P3-master-Murat_Tunc\data\IMG\cen
        ter_2019_01_07_17_24_53_542.jpg
         C:\Users\AR430149\AppData\Local\Continuum\anaconda3\envs\CarND-Term1-Starter-Kit\CarND-Behavioral-Cloning-P3-master-Murat_Tunc\data\IMG\le
        ft_2019_01_07_17_24_53_542.jpg
```

Result is greate. I used my laptop which has powerful GPU.
Now we have model.h5 (but be cool may be it is not enough power to auto mode)

Download
https://github.com/UDACITY/CarND-Behavioral-Cloning-P3


udacity / CarND-Behavioral-Cloning-P3

Watch 32 | Star 270 | Fork 1,369

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights

Starting files for the Udacity CarND Behavioral Cloning Project https://www.udacity.com/drive

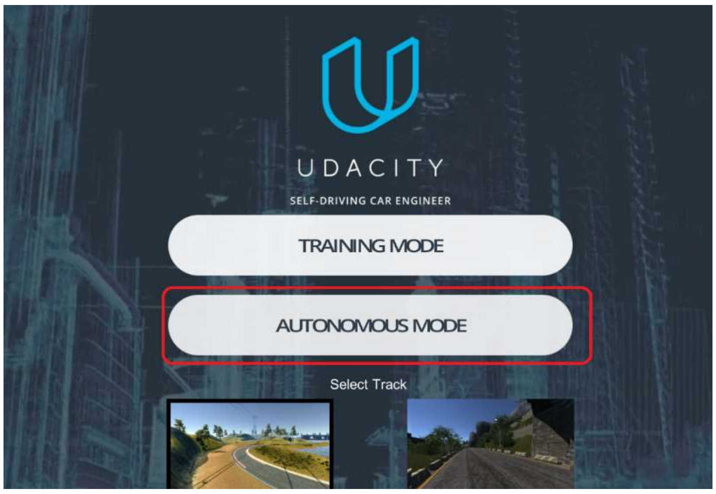
30 commits | 1 branch | 0 releases | 9 contributors | MIT

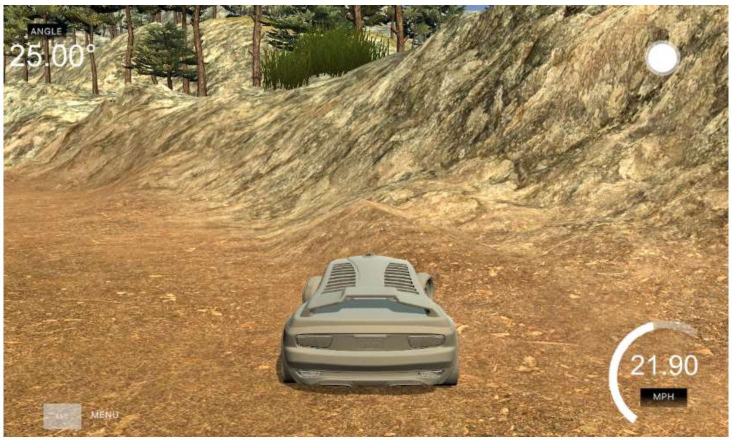
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

citlaligm Update README.md — Latest commit f6a0598 on 27 Jun 2018

examples	Added more info to README, included writeup template, changed drive.p...	2 years ago
LICENSE	Add license	11 months ago
README.md	Update README.md	7 months ago
drive.py	Reduce set speed to the original	2 years ago
set_git.sh	Adds git pull and fix echo of the user name and email	7 months ago
video.py	Fixes for robustness with respect to different OS	a year ago
writeup_template.md	Fix write-up markdown	a year ago

And then press autonomus mode to see the result.





The first result is very bad. The car goes to hill ☺

We need some image preprocessing and adjusting steering angles.

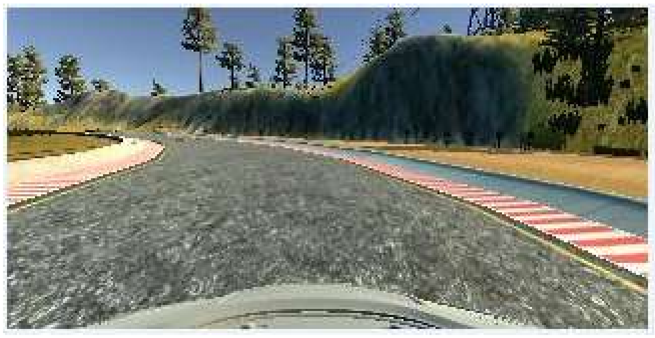
3. Creation of the Training Set & Training Process

The data provided by Udacity has 8036 images from each camera, resulting in 24108 images. On the other hand, I generated my own data.I have two typs of recording.

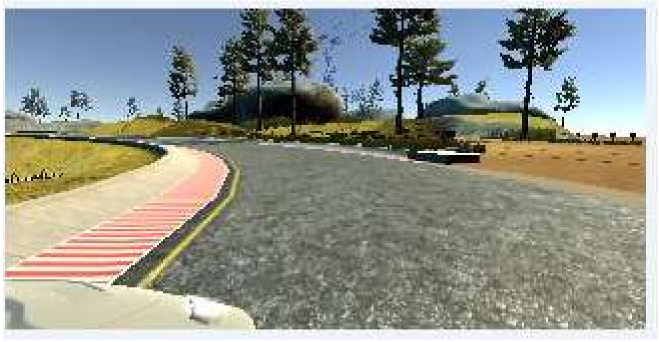
1→On the center→1791 images.

2→Recovering from left to center and Recovering from right to center→1477 images
Total Image Numbers=3268
Why we need recovering?
Becaouse you have learn to drive from center. If any trouble on the left and right side , you have to recover the car to center.

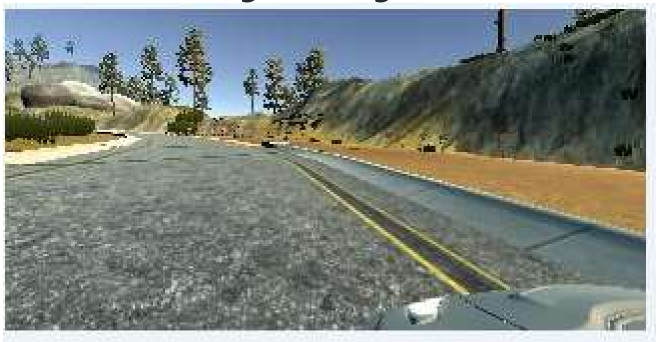
From Center



Recovering from left to center



Recovering from right to center

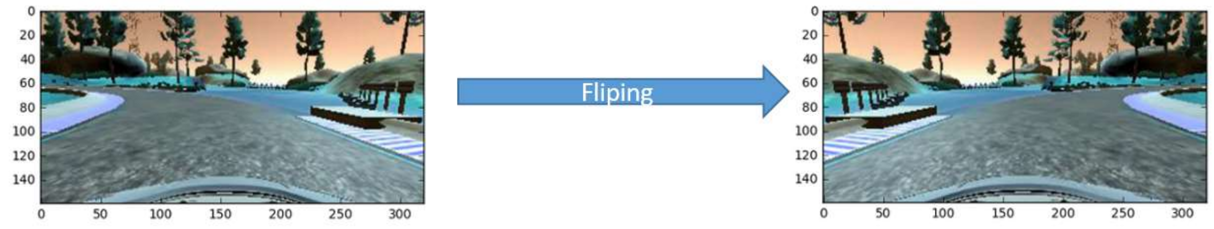


It is very important!!! Do not forget this recovering behaviour.
I also flipped images and angles thus generating 2x more training data

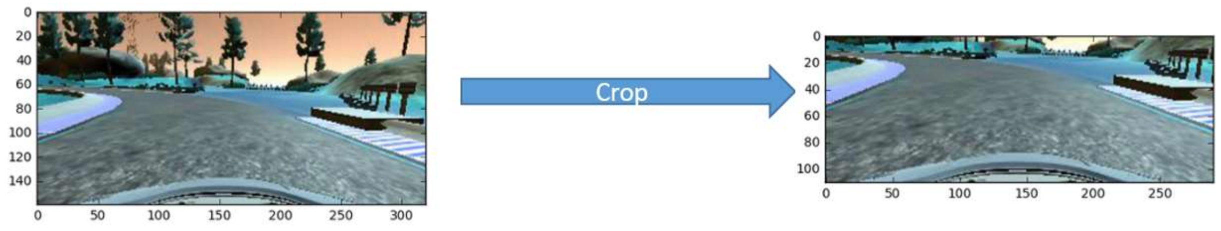
The network will not converge to a good solution. We have to preprocess the data: to do what is called "image augmentation".

Image augmentation & Preprocessing steps

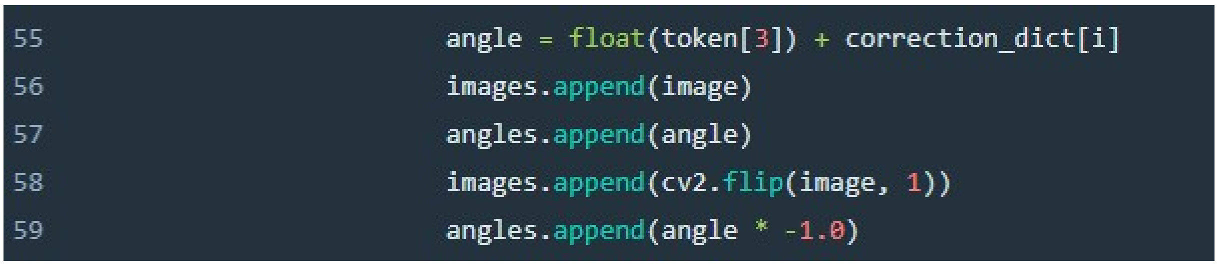
Flip the image→
On the other hand,we can randomly to flip the image, and invert the steering angle. This way, we can neutralize some tendency of the human driver that drove a bit more to the left or to the right of the lane. We only care about the section of the road that has lane lines, so I cropped out the sky and the steering wheel.



Crop→
The image was cropped to remove irrelevant portions of the image, like the sky, or the trees. Doing this, we're assuming the camera is fixed in a stable position.



Add Steering angle with correction coefficient→



```
                        angle = float(token[3]) + correction_dict[i]
                        images.append(image)
                        angles.append(angle)
                        images.append(cv2.flip(image, 1))
                        angles.append(angle * -1.0)
```

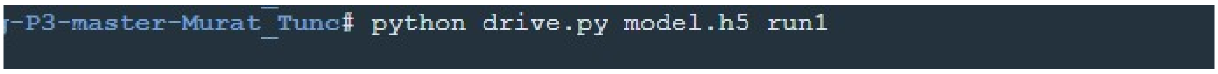
I then preprocessed this data by cropping off the top 70 pixels and bottom 25 pixels that don't show the road.

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

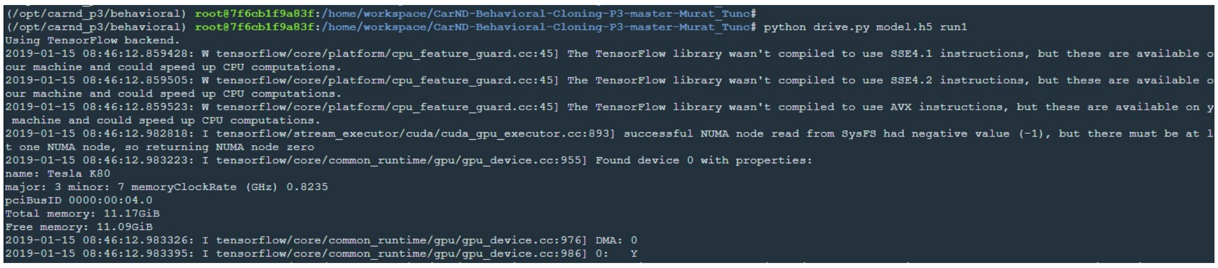
I used this training data for training the model. The validation set helped determine if the model was over or under fitting. The ideal number of epochs was 1 as evidenced by loss: 0.078 and val_loss: 0.071. I used an adam optimizer so that manually training the learning rate wasn't necessary.

At the end of the work we have these results:

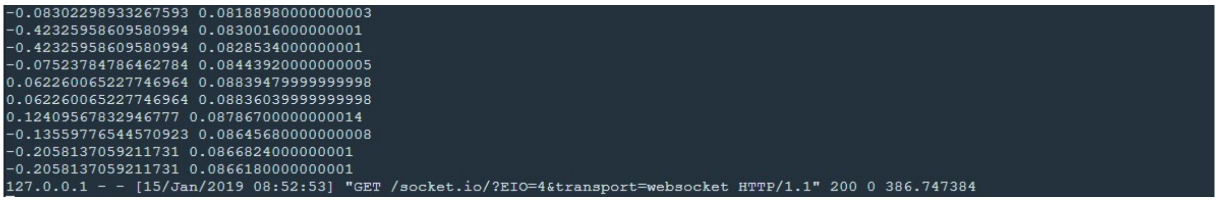
GPU Enable, Try model.h5 with saving images from auto mode



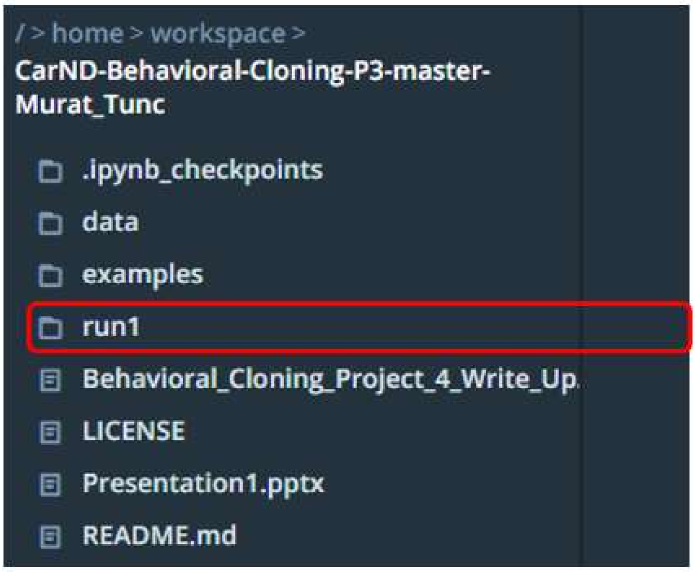
```
-P3-master-Murat_Tunc# python drive.py model.h5 run1
```



```
(/opt/carnd_p3/behavioral) root@7f6cb1f9a83f:/home/workspace/CarND-Behavioral-Cloning-P3-master-Murat_Tunc#
(/opt/carnd_p3/behavioral) root@7f6cb1f9a83f:/home/workspace/CarND-Behavioral-Cloning-P3-master-Murat_Tunc# python drive.py model.h5 run1
Using TensorFlow backend.
2019-01-15 08:46:12.859428: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available o
our machine and could speed up CPU computations.
2019-01-15 08:46:12.859505: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available o
our machine and could speed up CPU computations.
2019-01-15 08:46:12.859523: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are available on y
 machine and could speed up CPU computations.
2019-01-15 08:46:12.982818: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:893] successful NUMA node read from SysFS had negative value (-1), but there must be at l
t one NUMA node, so returning NUMA node zero
2019-01-15 08:46:12.983223: I tensorflow/core/common_runtime/gpu/gpu_device.cc:955] Found device 0 with properties:
name: Tesla K80
major: 3 minor: 7 memoryClockRate (GHz) 0.8235
pciBusID 0000:00:04.0
Total memory: 11.17GiB
Free memory: 11.09GiB
2019-01-15 08:46:12.983326: I tensorflow/core/common_runtime/gpu/gpu_device.cc:976] DMA: 0
2019-01-15 08:46:12.983395: I tensorflow/core/common_runtime/gpu/gpu_device.cc:986] 0:   Y
```



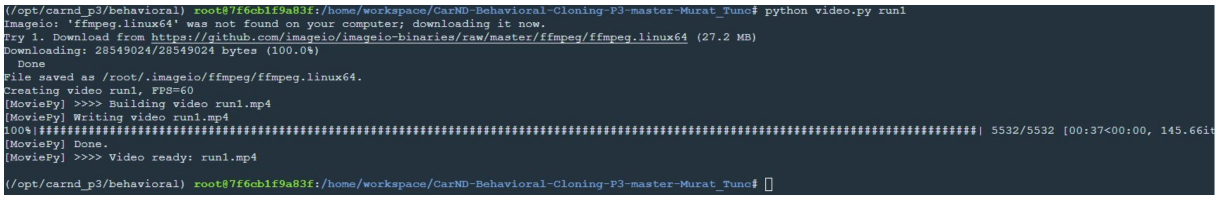
```
-0.08302298933267593 0.08188980000000003
-0.42325958609580994 0.0830016000000001
-0.42325958609580994 0.0828534000000001
-0.07523784786462784 0.08443920000000005
0.062260065227746964 0.08839479999999998
0.062260065227746964 0.08836039999999998
0.12409567832946777 0.08786700000000014
-0.13559776544570923 0.08645680000000008
-0.2058137059211731 0.0866824000000001
-0.2058137059211731 0.0866180000000001
127.0.0.1 - - [15/Jan/2019 08:52:53] "GET /socket.io/?EIO=4&transport=websocket HTTP/1.1" 200 0 386.747384
```



/ > home > workspace >
CarND-Behavioral-Cloning-P3-master-Murat_Tunc

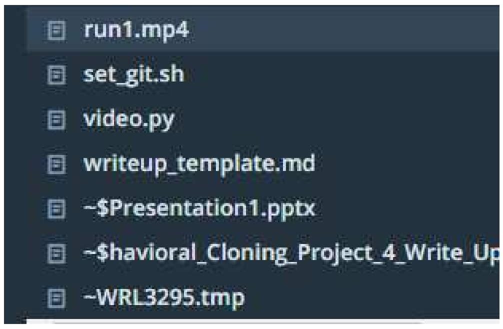
- .ipynb_checkpoints
- data
- examples
- run1
- Behavioral_Cloning_Project_4_Write_Up
- LICENSE
- Presentation1.pptx
- README.md

Create video with “videp.py”



```
(/opt/carnd_p3/behavioral) root@7f6cb1f9a83f:/home/workspace/CarND-Behavioral-Cloning-P3-master-Murat_Tunc# python video.py run1
Imageio: 'ffmpeg.linux64' was not found on your computer; downloading it now.
Try 1. Download from https://github.com/imageio/imageio-binaries/raw/master/ffmpeg/ffmpeg.linux64 (27.2 MB)
Downloading: 28549024/28549024 bytes (100.0%)
  Done
File saved as /root/.imageio/ffmpeg/ffmpeg.linux64.
Creating video run1, FPS=60
[MoviePy] >>>> Building video run1.mp4
[MoviePy] Writing video run1.mp4
100%|##########################################################################################################| 5532/5532 [00:37<00:00, 145.66it
[MoviePy] Done.
[MoviePy] >>>> Video ready: run1.mp4

(/opt/carnd_p3/behavioral) root@7f6cb1f9a83f:/home/workspace/CarND-Behavioral-Cloning-P3-master-Murat_Tunc# 
```

run1.mp4 is created



- run1.mp4
- set_git.sh
- video.py
- writeup_template.md
- ~$Presentation1.pptx
- ~$havioral_Cloning_Project_4_Write_Up
- ~WRL3295.tmp

run1.mp4→

Many Thanks Udacity Team fort his great course and Project.

Best Regards.
Murat Tunç
muratbekonet.mt@gmail.com