#Thanks to the suggestions from last submission, I finished the
#project finally. This is the last project of my term 1. Thank you
#so much for your suggestions. Cheers!

Behavrioal 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

My project includes the following files:

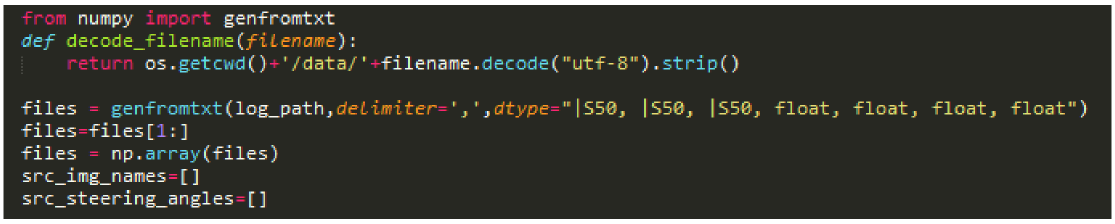
- model.py containing the script to create and train the model
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- report.pdf: summarizing the results (you are reading it!)
- video.mp4: the result video.
- Data.ipynb: notebook file to visualize the data

Appropriate training data

In my first submission, I recorded some files by using the simulator. However, I found their quality were not good. Then I use the sample data to train my model.

And this time, I still used those data.

Before training, I fetched the data from driving_log.csv and the IMG folder. After randomly shuffled the data, I did a normalization operation to those training samples. [model.py line 80 to 86, line 96]



```
from numpy import genfromtxt
def decode_filename(filename):
    return os.getcwd()+'/data/'+filename.decode("utf-8").strip()

files = genfromtxt(log_path,delimiter=',',dtype="|S50, |S50, |S50, float, float, float, float")
files=files[1:]
files = np.array(files)
src_img_names=[]
src_steering_angles=[]
```

```
for line in files:
    steering_angle = line[3]
    center_name= decode_filename(line[0])
    left= decode_filename(line[1])
    right= decode_filename(line[2])
    src_img_names.append(center_name)
    src_steering_angles.append(steering_angle)
    src_img_names.append(left)
    src_steering_angles.append(steering_angle+CAMERA_OFFSET)
    src_img_names.append(right)
    src_steering_angles.append(steering_angle-CAMERA_OFFSET)

#shuffle data to avoid overfitting
from sklearn.utils import shuffle
src_img_names,src_steering_angles= shuffle(src_img_names,src_steering_angles)
#reformat list as np array
src_img_names=np.array(src_img_names)
src_steering_angles=np.array(src_steering_angles)
```

At first ,I choose 80% of the img data as my training data and 20% as my validation data. However, this would course an memoryError on AWS when reading the the validation file. So I changed the the ratio to 9:1 (model.py line 124)

```
124        for index in range(int(len(src_img_names)*0.9)):
```

A small but an important point is that the driving_lot.csv has a header row, which would cause an error if we did not noticed that. Before load the data to the model, this line should be deleted.[model.py line 41]

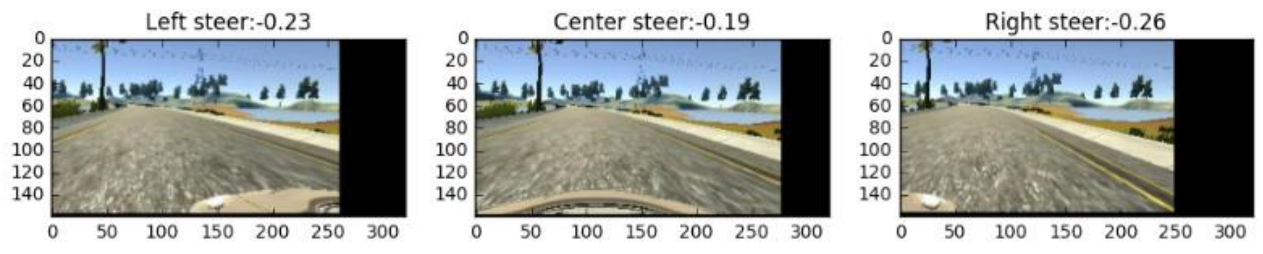
```
41    files=files[1:]
```

Visualize the data:

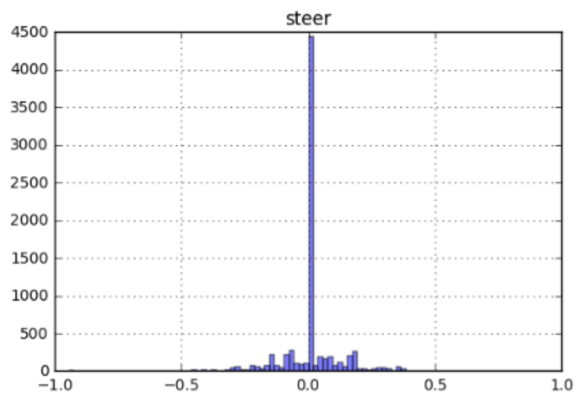
Most of this part were finished in Data.ipynb

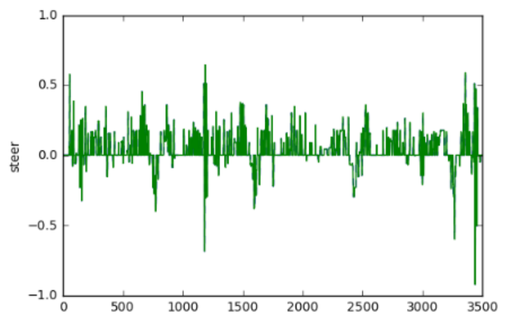
I loaded the source data(image and steer data) and visualized them:

Here are three samples(examples of images from the dataset)



And I looked into one sample and visualized its steering angel distribution :





As we can see from these two plots, most of the time, the vehicle's steering angles were close to 0.0. That is because vehicles always try to drive in a straight line.

Followed by your review suggestions:

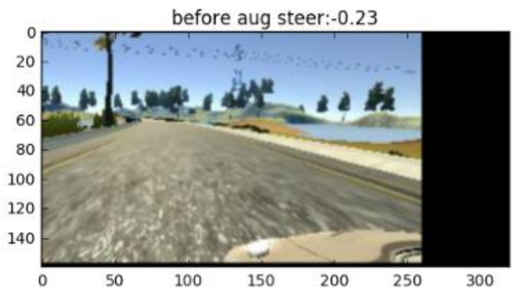
Other useful techniques can include:

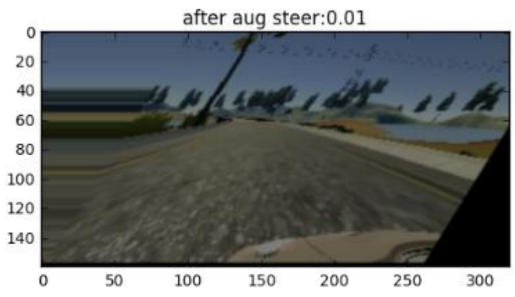
- flipping the images and the angles to augment the data
- filtering out the data for driving that isn't useful. For example, if the data captured the vehicle veering off the road, then those images and angles should be filtered out of the training set
  Behavioral Cloning Classroom Lesson 5: Data Collection Strategies

I chose to flip the images and the angles to augment the data. And then I build a pipline to finish this step: (data.ipynb, cell 36)

```
def augment_image(image, steering_angle):
    image = augment_brightness_camera_images(image)
    image, steering_angle = random_shear(image, steering_angle, 100)
    return image, steering_angle
```

The result I got was good:

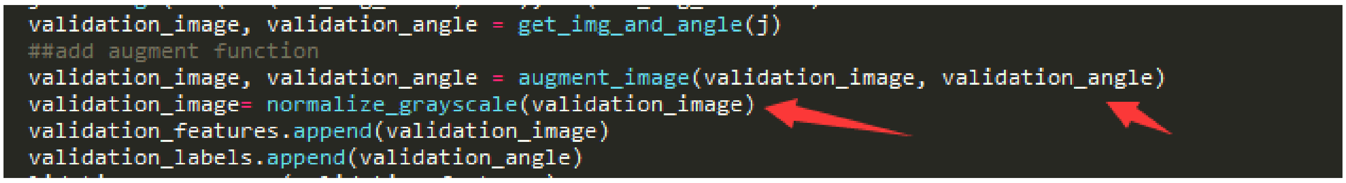




Since the augment step contribute a lot to the final accuracy, I implement this function in model.py.

The model.h5 is my stored model.

In my previous submission, I did not applied this step to my model and it might cause some errors when driving the vehicle in autonomous mode.

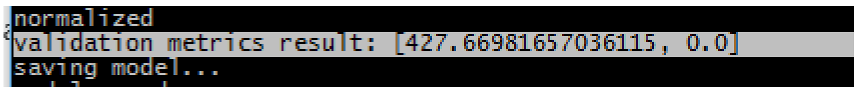


```
validation_image, validation_angle = get_img_and_angle(j)
##add augment function
validation_image, validation_angle = augment_image(validation_image, validation_angle)
validation_image= normalize_grayscale(validation_image)
validation_features.append(validation_image)
validation_labels.append(validation_angle)
```

And this time, I augment the data before I normalize them.

I further preprocessed them before feed them into the net. I also followed your suggestions and changed the architecture of the model.

The final validation metrics result is :



```
normalized
validation metrics result: [427.66981657036115, 0.0]
saving model...
```

The result is reasonable.

And the result is quiet well. You can found the output video in the video.mp4.

Using the Udacity provided simulator and my drive.py file, the car can be driven autonomously around the track. But last time I did not change my `throttle  and this time , I followed your suggestions and set the throttle as you suggested. It worked well.`

MANAGING THROTTLE PEDAL

```
throttle = max(0.1, -0.15/0.05 * abs(steering_angle) + 0.35)
```

Model Architecture and Training Strategy

An appropriate model arcthiecture has been employed

Layer1: convolutional neural network, input_shape=(160,320,3)

Layer2: ELU activation

Layer3: convolutional neural network

Layer4: ELU activation

Layer5: convolutional neural network

Layer6: ELU activation

Layer7: convolutional neural network

Layer8: ELU activation

Layer9: Flatten layer, flatten the network

Layer10: Dropout layer, prob=0.5, to prevent overfitting

Layer11: Fully connected layer, dense the net to 100

Layer12: ELU activation

Layer13: Dropout layer, prob=0.5, to prevent overfitting

Layer14: Fully connected layer, dense the net to 50

Layer15: ELU activation

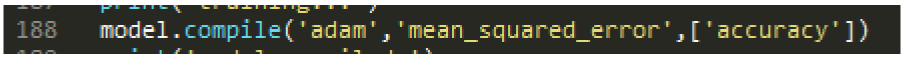
Layer16: Fully connected layer, dense the net to 10

Layer17: relu activation

Layer18: Fully connected layer, dense the net to 1

Model parameter tuning

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



```
model.compile('adam','mean_squared_error',['accuracy'])
```

The reason why I choose adam optimizer is that I used adam optimizer before and it performs well. It also appears in the lecture.

Solution Design Approach

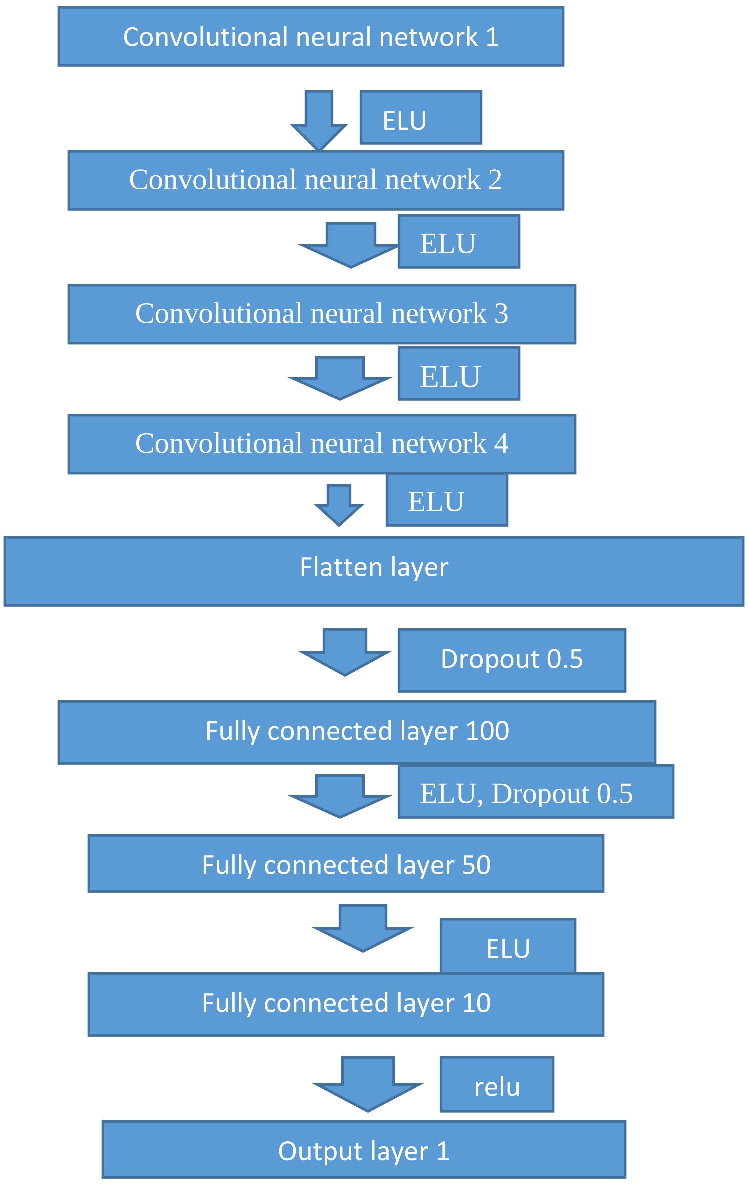
1. Based on the recommendation from other cohorts, I choose to train 5 epochs. And my batch size is 120, which is a reasonable value when training images. Since the training is just like the traffic sign classifier project, I did not change so much on those parameters. As a result, my code successfully operates on the simulation samples.

2. The filters I used in the convolutional neural net are 5*5(conv 1,2,3) and 3*3(conv4)

3. The model uses dropout layers or other methods to reduce overfitting.[line 140, line 144]

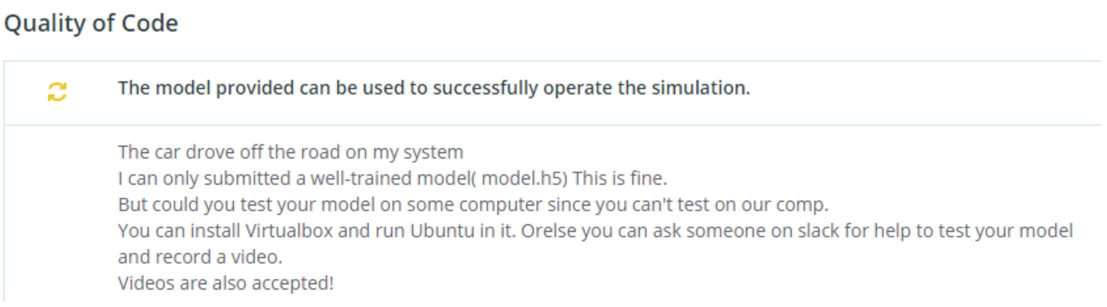
4. The generator. This is an essential part for the whole project. Firstly, I have no idea what to do about this. After going through those questions on the forum, I have some deeper understanding upon this. The mechanism of the my generator is actually continuously providing data to the training pipline. Each time the amount it provide is a batch size number. It just like feeding data to GPU, which process data at the same time as much as possible.

Final Model Architecture


Convolutional neural network 1
ELU
Convolutional neural network 2
ELU
Convolutional neural network 3
ELU
Convolutional neural network 4
ELU
Flatten layer
Dropout 0.5
Fully connected layer 100
ELU, Dropout 0.5
Fully connected layer 50
ELU
Fully connected layer 10
relu
Output layer 1

Discussions:

1. My computer's os is Windows. I can run the simulator on my computer but I cannot run my code because my Windows version is not able to run a docker. All my code were test and compile on AWS, which only has command line. In the last submission, you suggested that:


Quality of Code

The model provided can be used to successfully operate the simulation.

The car drove off the road on my system
I can only submitted a well-trained model(model.h5) This is fine.
But could you test your model on some computer since you can't test on our comp.
You can install Virtualbox and run Ubuntu in it. Orelse you can ask someone on slack for help to test your model and record a video.
Videos are also accepted!

The method I used to solve this problem:

This time, I asked my friends, Chunyuanm Jiang, in Canada to help me to test my model and get the final output video.mp4

The reason why I cannot test on my own linux is because of the Chinese Great Fire Wall. I tried several weeks to solve this problem. But the environment files (CuDNN, CUDA, tensorflow, etc.) werer too large to download due the the slow network(VPN) connection. I got “time-out error” every time.

2. One of most challenge part in this project is actually the generator part I think. At first, I did not realized the importance of generator. My first generator went wrong but the I get an error in other part, which confused me for a while.

3. Before I continue to work on this project, I finished the vehicle detection project. I was wondering the image preprocess methodsI used in project 5, such as the undistortion method, etc., can be used in this project to generate better training images, Which might led to better result. The whole project can be improved on the architecture and data preprocessing part. I have seen some other cohorts posted their result on YouTube and their results are accurate.

4. The softmax activation is not suitable for this project. When I was trying to apply softmax activation in this project. It would caused error, which is about the tensor dimension. In my first submission, I used relu. And you suggested that ELU would be a better choice. I changed my activation layer to ELU and get a better result. Since I changed so many parameters this time, there might be a better combination again. In order to improve this pipline. I think I can try more different parameters

combinations to get a better result. In other words, there might be some better activation functions for my pipline. I would try to go deeper.

5. Thanks to forum. Classmates provided me a lot of help. When I was stucking on some points, they always provided me lots of hints and even teach me how to solve the challenge.

6. I want to express my gratefulness to my friend Chunyuanm Jiang again. Without his generous help, I might not able to test my model on the simulator and obtain the video.mp4. The Chinese Great Fire Wall is really an obstacle for Chinese to learn innovative things.

7. At last, thank you so much for your reviews. I really get lots of help from your suggestions.