Behavrioal Cloning Project

Introduction

Files in project

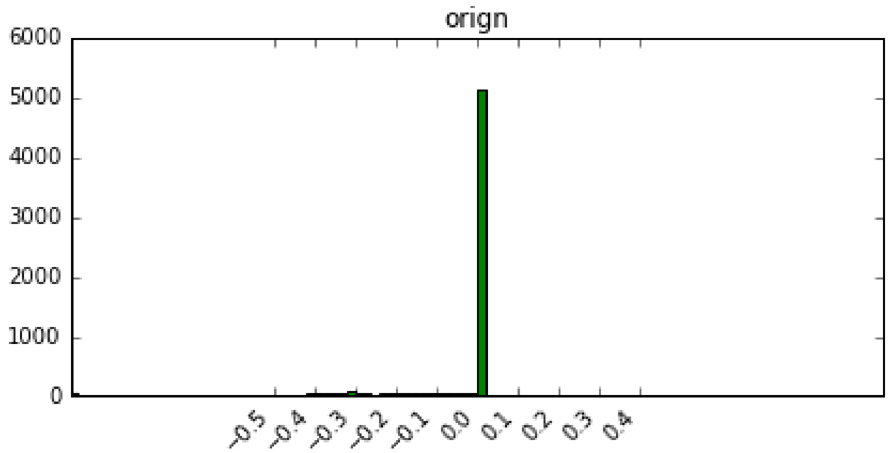
In this project I maintain following files:

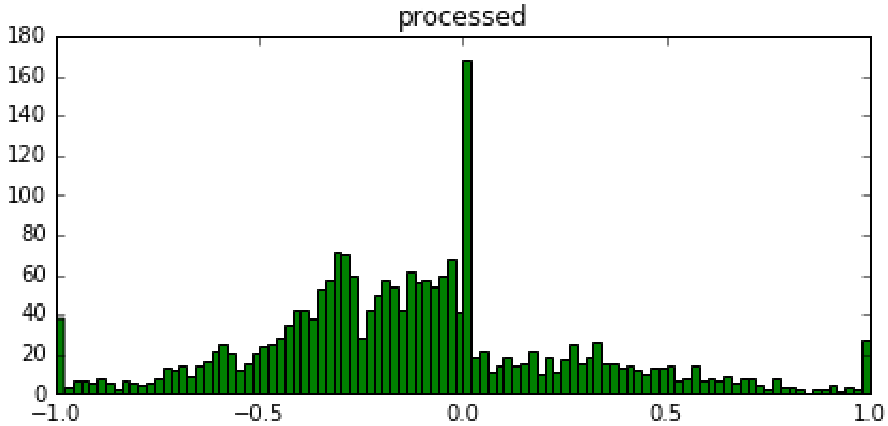
model.py containing all of the CNN model functionality
model_func.py containing many functions utilized in the model.py.
drive.py for driving the car in autonomous mode
writeup_report.md summarizing the results

I used the data generated from simulator to train my model with keyboard. Because I'm not very good at control the car by keyboard, this lead my car has some oscillation in its running, but it still can hold in the lane. I believe with better operation, the car can also run better.

Pre-Processing

I collect datas by drive the car in the simulator with 7-8 laps with 2 laps where the car recovers from the right side of the road or the left side of the road and moves back center, add up to 7244 frames. I plot histogram of images from center camera as follow:





These frames are split uniformly on the corresponding steering angles. I found there are a large no of image corresponding to nearly zero steering angle.

I use function, train_data( ) to pre-process origin data set. Because I think the color changes with the background and have not help to drive, it is actually noise. Then I convert the orign image to HSV space and then resize the S channel to 32x64 pixels as trainning sample.

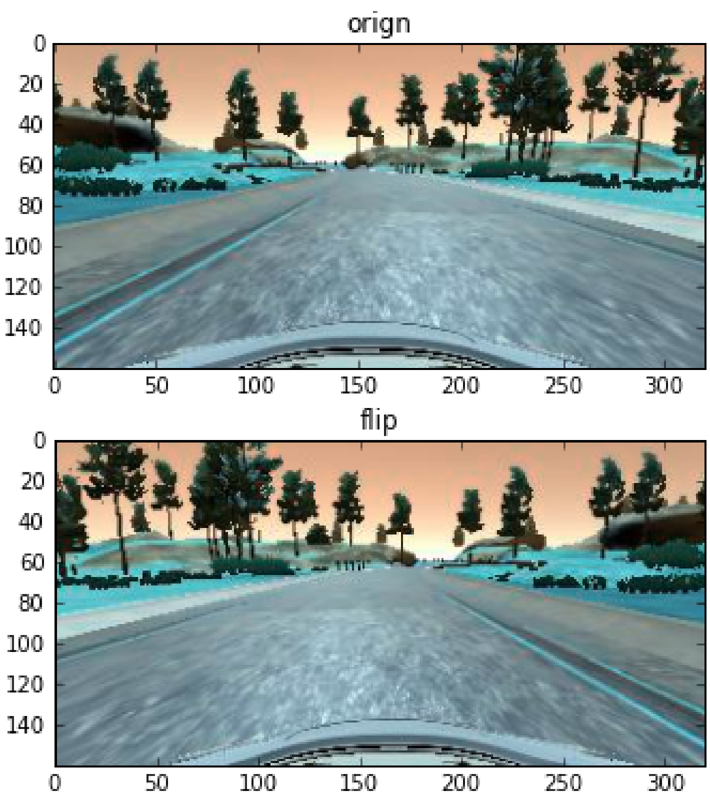
Model Architecture and Training Strategy

To train the model:
python model.py

To run the model and test the drive:
python drive.py model.json

The model.py loads the data preproeessed from the a.npy and b.npy files.

I saw the nvidia paper and make my model based on this paper. Later I read another paper , in which the author simplified the architecture of nvidia paper because the enviroment in simulator is much simple than actual enviroment. According to this analysis, I also simplied my model. So I make the following changes:
The data is normalized in the model using a Keras lambda layer first. I add some max pooling layers followed by Dropout layer after the convolution to decrease overfitting. I use ELU as activation function rather than ReLU accoding to the paper FAST AND ACCURATE DEEP NETWORK LEARNING BY EXPONENTIAL LINEAR UNITS ELUS. The model used an adam optimizer, so the learning rate was not tuned manually. To valiate my model, I have initially split the dataset in training and validation and 80% for trainning ,20% for validation. Because the network should output a continous value, so this is a regression problem and the mean squared error is used as objective function.
To get more data help the car recovery from side to center, I used image from both side camera by added a correction to the corresponding angles. To overcome the left turn bias, I flipped the image and steering angle (multiple by -1). Here is an image that has then been flipped:



At the end,12936samples(include flipped samples) were saved in npy files. In the file model.py I defined a generator function to process data with batchsize=256. Learning rate wasn’t necessary. and set epoch=10.

CNN Model Architecture

My final model looks as following:

Layer (type)	Output Shape	Param #	Connected to

reshape_3 (Reshape)	(None, 32, 64, 1)	0	reshape_input_3[0][0]
lambda_3 (Lambda)	(None, 32, 64, 1)	0	reshape_3[0][0]
convolution2d_9 (Convolution2D	(None, 30, 62, 8)	80	lambda_3[0][0]
convolution2d_10 (Convolution2D	(None, 28, 60, 12)	876	convolution2d_9[0][0]
maxpooling2d_5(MaxPooling2D)	(None, 7, 15, 12)	0	convolution2d_10[0][0]
dropout_7 (Dropout)	(None, 7, 15, 12)	0	maxpooling2d_5[0][0]
convolution2d_11 (Convolution2D	(None, 5, 13, 16)	1744	dropout_7[0][0]
convolution2d_12 (Convolution2D	(None, 3, 11, 16)	2320	convolution2d_11[0][0]
maxpooling2d_6 (MaxPooling2D)	(None, 1, 5, 16)	0	convolution2d_12[0][0]
dropout_8 (Dropout)	(None, 1, 5, 16)	0	maxpooling2d_6[0][0]
flatten_3 (Flatten)	(None, 80)	0	dropout_8[0][0]
dense_7 (Dense)	(None, 128)	10368	flatten_3[0][0]
activation_5 (Activation)	(None, 128)	0	dense_7[0][0]
dropout_9 (Dropout)	(None, 128)	0	activation_5[0][0]
dense_8 (Dense)	(None, 64)	8256	dropout_9[0][0]
activation_6 (Activation)	(None, 64)	0	dense_8[0][0]
dense_9 (Dense)	(None, 1)	65	activation_6[0][0]

Trainable params: 23,709
Trainable params: 23,709
Non-trainable params: 0

Training

In original attempts at training the model, I failed, the car seemed to drive almost straight all the time. I notice there are a large majority of the images having zero steering angle, and the model tending to overfit to this situation. So I drop 95% of the images where the steering angle is less than 0.005.
After this process, I get a trained model that would navigate whole track. I also find the model is sentive to the ratio of nearly zero data , so the one should take great care in preprocessing trainning dataset to get the balance.

Conclusion

In the trainning process, I find the quality of trainning data is more important than the model architecture, so one should make great effort to get data with good balance and enough amount.