Project 3 Behavior Cloning

1 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
- model.json
- BehaviorCloning_Project3-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.json

Please use above as I could not drive the car executing model.h(may besome local system problem):

python drive.py model.h

Model weights are generated by executing

python model.py

3. Code Summary

The model.py file contains the code for preprcessing , training and saving the convolution neural network. The file shows the pipeline

used for training and validating the model, and it contains comments to explain how the code works.

The code makes an array of the training images, crops them and feed to the Keras model. I have used comma.ai model archtecture.

I tried modifications like additional layer, differnet filter size, but could not got the Image get resized and croped and normalised within the Keras model. Python generator is used to take away the burden from the memory and fasten the calculation. After executing the model, it generates model.h5 and model.json files. Then saved weights can be used for the autonomous driving using the driver,py. Dues to some local system issue , I could not run model.h5 file. Instead it worked with the model.json file.

4.Training Data Generation: Udacity's Car-Driving Simulator

The model was trained by driving a car in the Udacity's car-driving simulator. The simulator recorded 9-10 datapoints per second. Each datapoint comprised 7 attributes:

- Image taken from the center camera on the car (320 x 160 pixel image with 3 colour channels)
- Image taken from the left camera on the car
- Image taken from the right camera on the car
- Throttle
- Speed
- Brake
- Steering angle (variable to predict, a float ranging from -1.0 to 1.0 inclusive)

Data Preprocessing

I started using Udacity's data and some recovery data.

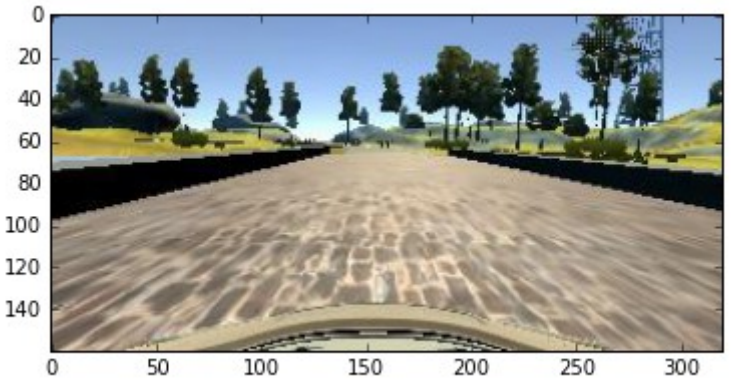
When I have not got the required results, I have generated more data with additional laps. Later I used some additional data by flipping the images.

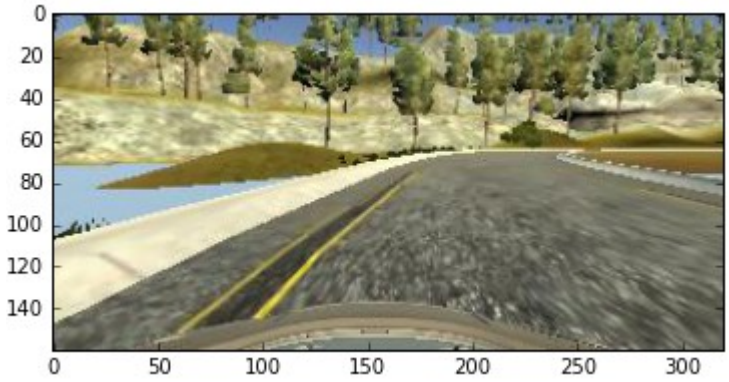
I recorded myself driving around Track 1 first using the using the mouse.

1. Recovery: I then recorded driving the car only from the side of the road to the center, both along straight roads and along curves. This was so the model could learn to drive back to the center of the road if it swerved to the side. I did not record driving off to the side of the road because that's not something I want the model to do. But this is the toughest part in the project.

2. Whole laps: I recorded driving the car around the training track for two laps. I have not used track 2 at all for the training.

The simulator captures data into a csv log file which references left, centre and right captured images within a sub directory. Telemetry data for steering, throttle, brake and speed is also contained in the log. Only steering was used as label in this project. Below are the example image at center and at the left.





The image output from the simulator is of size 160*620 with three color channels.

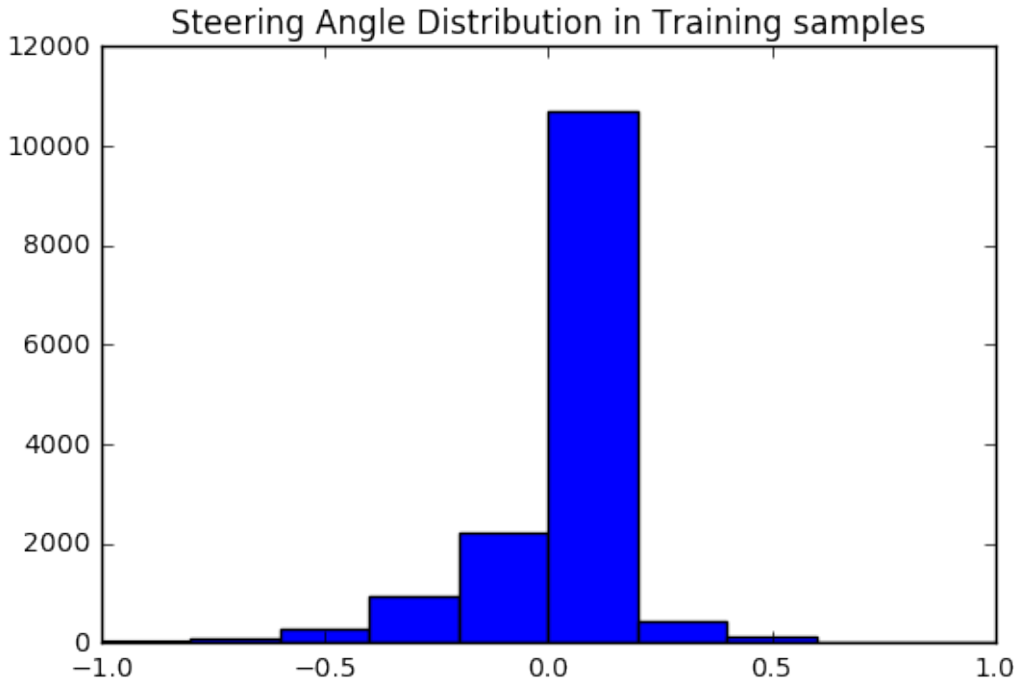
The first step I did was to reduce the size, which helps to hold more images in the memory at one time. I choose to resize the image to 40*160 and then cropped the image 70 pixels from top and 25 from bottom. Please look at the line no. 83 and 93 in the model.py.

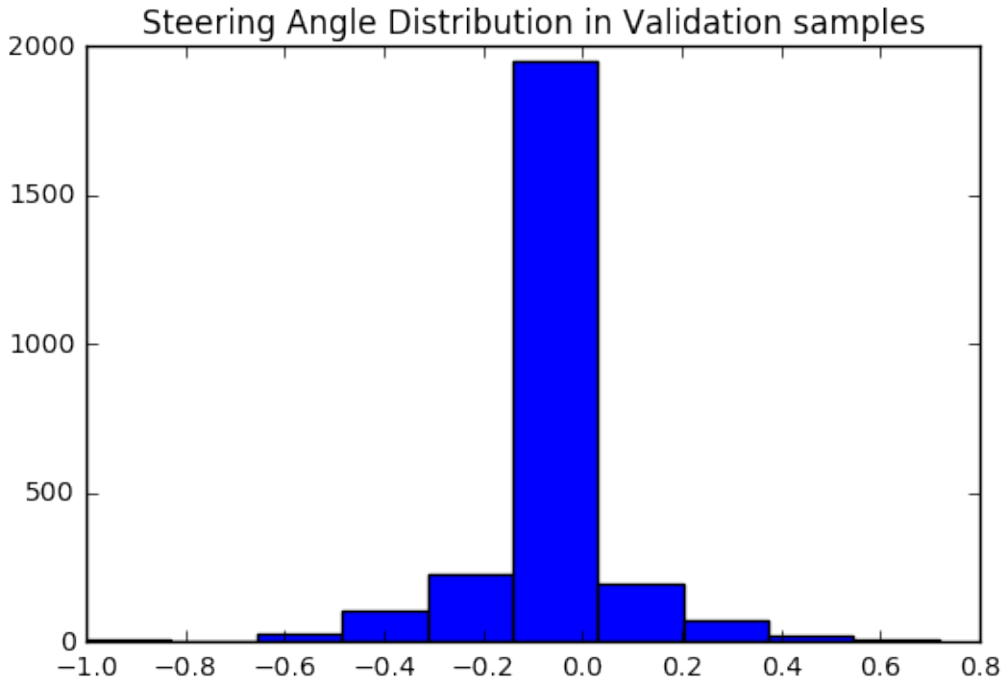
Cropping and normalisation done on the image inside Keras model. But I dont unerstand how to plot the images within the Keras model.

Total size of the data used is 17584. Out of that 15% ie 2638 images are used for validation and 14946 images are used to train the model.Below are distribution of training and validation samples:

Training samples are biased towards the negative steering angle.

The reason for that is initial failure to turn left and running of the car out of the track. I have created extra images by moving the car on little right side and taking the left turn. But it helped to get good results.





Model Architecture and Training Strategy

I have used a python generator to feed training and validation data to the model while training using the Keras fit_generator() function. I have added a callback to the model so that it saves the best weights of each training session to file model.h5 I tried with different batch sizes of 64 , 128 and 256. But decided on the 128 which gives miminum validation loss. Beyond 30 epochs, model results are not improving. Even similar results for 20 epochs. But decided on 30 as I decided to overtrain the model so that car can move for few minutes at least.(as my most of the results were very poor. The model has been saved as model.json at the end of each training session.

1. I started with comma.ai. Tried changing the different filtersizes and adding flatten layer, but performance was too poor on my all experiments that I decided to finalise on commai.ai model itself. Beleing that this is the best model, I tweaked rest of the

parameters like batch size, epoch , learning rate. Then I worked on different imput trials as discussed in training data generation.

My model consists of a convolution neural network with 2*2 and 4*4 filter sizes and depths between 16 to 64 (model.py lines 107-115)

The model includes ELU layers to introduce nonlinearity (code line 20), and the data is normalized in the model using a Keras lambda layer (code line 104).

20 percent of the total deta was used for the validatiion. Data was shuffled before feeding to the model to avoid biased results.

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 21).

The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 10-16). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

3. Model parameter tuning

The model used an adam optimizer, I have not experimented much with it. But got some improved results with the learning rate change from 0.0005 to 0.0001. So I adhered to it till last.

Final Model architecture:

Layer (type)	Output Shape	Param #	Connected to
cropping2d_9 (Cropping2D)	(None, 65, 320, 3)	0	cropping2d_input_9[0][0]
lambda_12 (Lambda)	(None, 40, 160, 3)	0	cropping2d_9[0][0]
lambda_13 (Lambda)	(None, 40, 160, 3)	0	lambda_12[0][0]
convolution2d_16 (Convolution2D)	(None, 10, 40, 16)	3088	lambda_13[0][0]
elu_26 (ELU)	(None, 10, 40, 16)	0	convolution2d_16[0][0]
convolution2d_17 (Convolution2D)	(None, 5, 20, 32)	12832	elu_26[0][0]
elu_27 (ELU)	(None, 5, 20, 32)	0	convolution2d_17[0][0]
convolution2d_18 (Convolution2D)	(None, 3, 10, 64)	51264	elu_27[0][0]
flatten_6 (Flatten)	(None, 1920)	0	convolution2d_18[0][0]
dropout_1 (Dropout)	(None, 1920)	0	flatten_6[0][0]
elu_28 (ELU)	(None, 1920)	0	dropout_1[0][0]
dense_16 (Dense)	(None, 512)	983552	elu_28[0][0]
dropout_2 (Dropout)	(None, 512)	0	dense_16[0][0]
elu_29 (ELU)	(None, 512)	0	dropout_2[0][0]
dense_17 (Dense)	(None, 1)	513	elu_29[0][0]

```
Total params: 1,051,249
Trainable params: 1,051,249
Non-trainable params: 0
```

The model was compiled with an adam optimizer (learning rate =0 . 0001), Mean Squared Error (mse) as a loss metric, and was set to train for 20 epochs with an early stopping callback which discontinues training when validation loss fails to improve for consecutive epochs. I chose to lower the learning rate in the adam optimizer because I found that the lower learning rate led to increased performance both in terms of mse and autonomous driving.

Testing model in the sumulator:

The script drive.py takes in a constant stream of images, resizes and crops them to the input shape for the model, and passes the transformed image array to the model which predicts an appropriate steering angle based on the image. The steering angle is then passed to the car as a control and the car steers itself accordingly. The autonomous vehicle travels like this through the

course, constantly putting out images and receiving steering angles and hopefully the model has been trained well enough that the steering angles it receives keep the vehicle driving safely in the middle of the lane and not drifting off the road or doing anything else which would be considered dangerous.

Discussion:

I started with the sample data provided with the course and the comma.ai model. To get the feel of the model, I started with 500 images only. Once I understood the step by step process I started working on the preprocessing of the images. Images were croped to

xxxxxx and normalised within the Keras code. Initial result was so poor that I could drive only around 5-10 seconds.

Then I had started with the incremental approach to train the car. I have used recovery images so that car can center itself whenever goes off track, which certainly helped to go little further, but not satisfactory.Then I added 2 extra laps and some flipped data. I could not understand, how to fo flip images within the generator.

So generated flipped images seperately and appended to the data.So after so much of data I can see my car showing little results.

Simultaneously I worked on the model , experimented with relu activation and dropouts to get the optimum results.

Though results was not good , at least I could tried to get it moving for around a mute . I have not tried at all on the second track as I could not get over the first.

Second Submission:

Due to poor performance in autonomous driving, reviwer suggested me few changes like changing the throttle calculation as below:

*throttle = max(0.1, -0.15/0.05 * abs(steering_angle) + 0.38)*

And surprisingly I can see car moving perfecly on the track. Thanks to the specific insight into the code. This proves that my training data and the modeling strategy was right.

Now I can see the car moving steadily most of the time through the center of the track. Whenever it tries to move away it align itself to the center position by calculating appropriate steering angle