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

Files Submitted & Code Quality

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_BehCloning_T1P3.pdf summarizing the results

* run1.mp4 & run2.mp4 videos recording of the vehicle driving autonomously around both the track for at least one full lap

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.h5
```

3. Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

My model is summarized below which is a Keras implementation of the Comma.ai’s convolutional neural network designed specifically to generate steering data for self-driving cars based on camera inputs. See "Solution Design Approach and Final model architecture" below for more details.

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 85 & 88).

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, so the learning rate was not tuned manually (model.py line 94).

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.

For details about how I created the training data, see the below section.

Model Architecture and Training Strategy

1. Solution Design Approach

All training was conducted on my laptop. I've set up TensorFlow to use my installed GPU

(Nvidia GeForce 920M).

My model is a Keras implementation of the Comma.ai’s convolutional neural network designed specifically to generate steering data for self-driving cars based on camera inputs. I referred from this link : https://github.com/commaai/research/blob/master/train_steering_model.py .

Before trying the above model, I tried using the Nvidia’s model but unfortunately and weirdly I was getting memory issue in my laptop. So, I had to try other model to check and came across Comma.ai’s implementation and worked well for me. Honestly, I was lazy enough to try what was wrong with Nvidia’s model !

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. I found that my first model had similar mean squared error on the training set as on the validation set.

To combat the overfitting, I modified the model by adding dropout layers before and between the bigger fully connected layers.

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 (model.py lines 70-94) consisted of a convolution neural network with the following layers and layer sizes ...

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
lambda_1 (Lambda)                (None, 160, 320, 3)   0           lambda_input_1[0][0]
____________________________________________________________________________________________________
cropping2d_1 (Cropping2D)        (None, 65, 320, 3)    0           lambda_1[0][0]
____________________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 17, 80, 16)    3088        cropping2d_1[0][0]
____________________________________________________________________________________________________
elu_1 (ELU)                      (None, 17, 80, 16)    0           convolution2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 9, 40, 32)     12832       elu_1[0][0]
____________________________________________________________________________________________________
elu_2 (ELU)                      (None, 9, 40, 32)     0           convolution2d_2[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 5, 20, 64)     51264       elu_2[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 6400)          0           convolution2d_3[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 6400)          0           flatten_1[0][0]
____________________________________________________________________________________________________
elu_3 (ELU)                      (None, 6400)          0           dropout_1[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 512)           3277312     elu_3[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 512)           0           dense_1[0][0]
____________________________________________________________________________________________________
elu_4 (ELU)                      (None, 512)           0           dropout_2[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 1)             513         elu_4[0][0]
====================================================================================================
Total params: 3,345,009
Trainable params: 3,345,009
Non-trainable params: 0
```

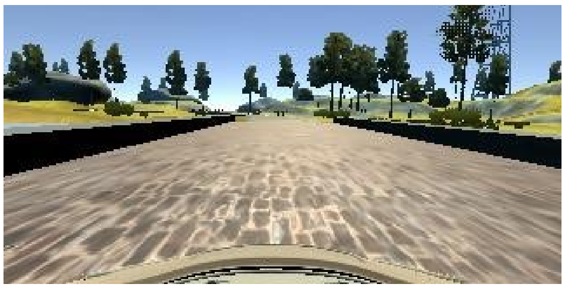
The cropping is followed by three convolutions and two fully connected layers. ELU (Exponential Linear Units) are used as activations within network which are said to speed up learning and lead to higher accuracies. Subsampling was also used in each convolutional layer. This architecture also employs aggressive dropout probabilities (0.2 and 0.5) to reduce overfitting. The model used an adam optimizer, so the learning rate was not tuned manually. The loss function used for optimization is the mean square error function.

3. Creation of the Training Set & Training Process

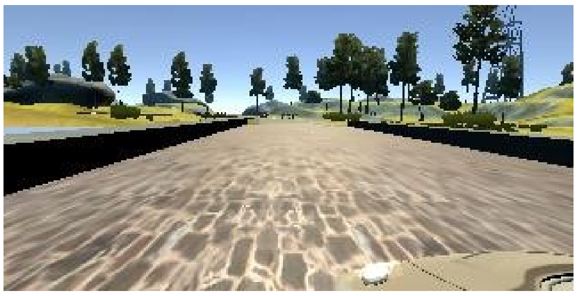
Initially, I tried capturing my own data and tried training with that to see if the behaviour is cloned ! But to my bad luck, even after several iterations, one point or the other the car was deviating from the main track. Probably it could be that the way I am capturing the training data may not be appropriate as I was not comfortable in capturing via either keyboard or mouse !

I decided to use the training data provided by the Udacity and the corresponding driving_log.csv.

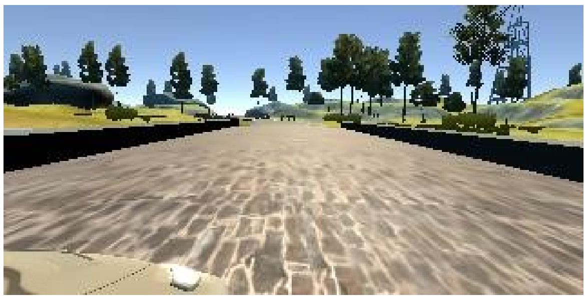
Centre:



Left:



Right:



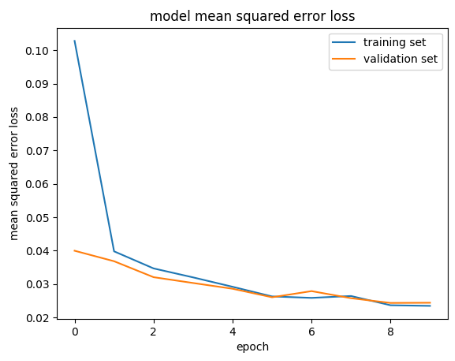
I used this training data for training the model. This consists of 8036 data samples . 85% of the samples were used to train while the remaining 15% was used for validation. Each data sample contains the steering measurement as well as three images captured from three cameras installed at three different locations in the car [left, center, right]. The data is biased towards left turns because of the track used to record this data. To address this biases, I used couple of data augmentation techniques: 1. Randomly choose among the three camera positions (left, center, right), and employ a steering correction of 0.25. The value of steering correction is adjusted based on how training the network was behaving. 2. Randomly flip the image and change the sign of the steering angle.

A python generator (model.py, line 50) was used to generate samples for each batch of data that would be fed when training and validating the network. I used a generator so that we don't need to store a lot of data.

I used an adam optimizer so that manually training the learning rate wasn't necessary.

```
Epoch 1/10

6830/6830 [==============================] - 205s - loss: 0.1028 - val_loss: 0.0400

Epoch 2/10

6830/6830 [==============================] - 112s - loss: 0.0398 - val_loss: 0.0368

Epoch 3/10

6830/6830 [==============================] - 70s - loss: 0.0347 - val_loss: 0.0321

Epoch 4/10

6830/6830 [==============================] - 55s - loss: 0.0320 - val_loss: 0.0303

Epoch 5/10

6830/6830 [==============================] - 82s - loss: 0.0292 - val_loss: 0.0286

Epoch 6/10

6830/6830 [==============================] - 35s - loss: 0.0263 - val_loss: 0.0261

Epoch 7/10

6830/6830 [==============================] - 32s - loss: 0.0259 - val_loss: 0.0279

Epoch 8/10

6830/6830 [==============================] - 31s - loss: 0.0264 - val_loss: 0.0258

Epoch 9/10

6830/6830 [==============================] - 27s - loss: 0.0237 - val_loss: 0.0244

Epoch 10/10

6830/6830 [==============================] - 25s - loss: 0.0235 - val_loss: 0.0244
```

Also, the graph to visualize the error loss:



With all above approach, the car was able to navigate correctly on track 1 (run1.mp4).

Unfortunately, model was not able to generalize for track 2 (run2.mp4) as first it was swerving left to right and eventually crashed. As a recommendation for further improvement aside from gathering more data, use the augmentation techniques like adding shadows, random brightness and contrasts, and vertical and horizontal shifting. Also, training data can be collected from track 2 for generalizing better.