udacity/**CarND-Behavioral-Cloning-P3**

Behavioral Cloning

Writeup Template

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

####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_report.md or writeup_report.pdf summarizing the results

####1. Solution Design Approach

The overall strategy for deriving a model architecture was to ...

My first step was to use a simple model, which only includes one Flatten layer. Because it will a good start to test the codes and compare with more sophistic neural networks. Then I applied the convolution neural network model similar to the Nvidia's architecture from their paper "End to End Learning for Self/ Driving Cars". This model proved to be effective for this problem without much . I did many works on hyperparameter tuning, adding dropout layers, and data augementation and image processing.

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 a low mean squared error on the training set but a high mean squared error on the validation set. This implied that the model was overfitting.

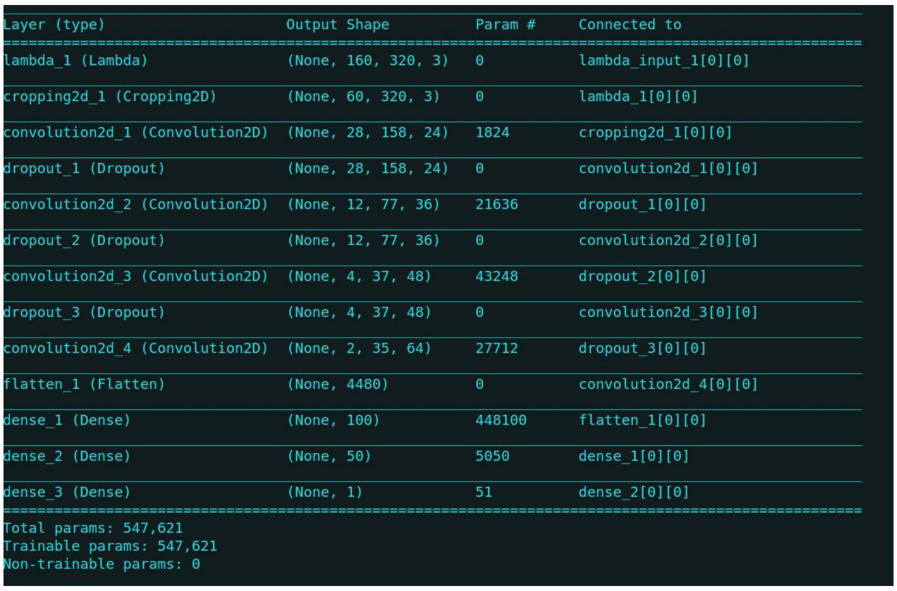
To combat the overfitting, I modified the model by adding more dropout layers. Then I process the images to improve the accuracy of the model.

The final step was to run the simulator to see how well the car was driving around track one. There were a few spots where the vehicle fell off the track. to improve the driving behavior in these cases, I crop the images and increase the number of epches from 3 to 5.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 18/ 24) consisted of 9 layers, including a normalization layer, 4 convolutional layers, and 3 fully connected layers. The first layer accepts an rgb image of size 160*320*3 and performs image normalization, resulting in features ranging from values －1 to 1. Then by a cropped layer the output is 3@60x320. The first convolutional layer accepts input of 3@60*320, has a filter of size 5x5 with stride of 2x2, resulting in output of 24@28x158. The second convolutional layer then applies a filter of size 5x5 with stride of 2x2, resulting in and output of 36@12x77, follows a dropout layer. The third convolutional layer applies a filter of size 5x5 with stride of 2x2, resulting in and output of 48@4x37, follows by a dropout layer. The fourth convolutional layer applies a filter of size 5x5 with stride of 2x2, resulting in and output of 64@2x35. The output is then flattened to produce a layer of 4480. The first fully/ connected layer then results in output of 100 neurons, followed by 50 neurons, and 1 neuron. The activation function used is the exponential linear unit(ELU), and an adaptive learning rate is used via the Adam optimizer.

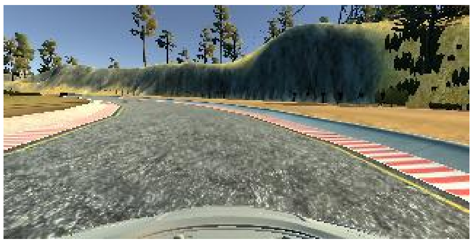
Here is a visualization of the architecture



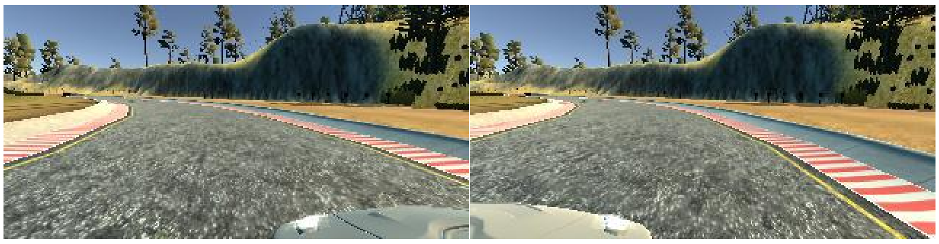
```
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
lambda_1 (Lambda)                (None, 160, 320, 3)   0           lambda_input_1[0][0]
____________________________________________________________________________________________________
cropping2d_1 (Cropping2D)        (None, 60, 320, 3)    0           lambda_1[0][0]
____________________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 28, 158, 24)   1824        cropping2d_1[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 28, 158, 24)   0           convolution2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 12, 77, 36)    21636       dropout_1[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 12, 77, 36)    0           convolution2d_2[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 4, 37, 48)     43248       dropout_2[0][0]
____________________________________________________________________________________________________
dropout_3 (Dropout)              (None, 4, 37, 48)     0           convolution2d_3[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 2, 35, 64)     27712       dropout_3[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 4480)          0           convolution2d_4[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 100)           448100      flatten_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 50)            5050        dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 1)             51          dense_2[0][0]
====================================================================================================
Total params: 547,621
Trainable params: 547,621
Non-trainable params: 0
```

####3. Creation of the Training Set & Training Process

To capture good driving behavior, I first recorded two laps on track one using center lane driving. Here is an example image of center lane driving:



I then recorded the vehicle recovering from the left side and right sides of the road back to center so that the vehicle would learn to .... These images show what a recovery looks like starting from ... :



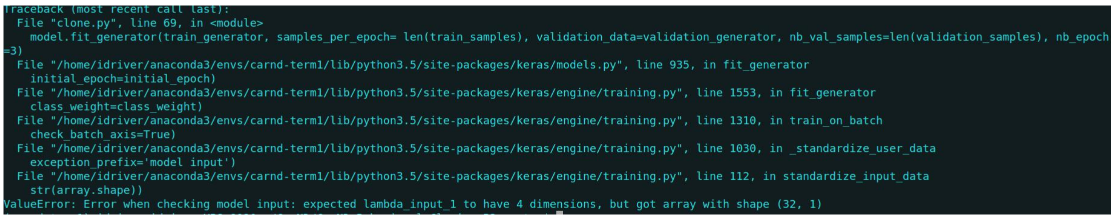
Then I repeated this process on track two in order to get more data points.

To augment the data sat, I also flipped images and angles. by augmented_images.append(cv2.flip(image,1))

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.

Training is done via a keras fit. When I applied the fit generator, there is always an error as following:



```
Traceback (most recent call last):
  File "clone.py", line 69, in <module>
    model.fit_generator(train_generator, samples_per_epoch= len(train_samples), validation_data=validation_generator, nb_val_samples=len(validation_samples), nb_epoch
=3)
  File "/home/idriver/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/keras/models.py", line 935, in fit_generator
    initial_epoch=initial_epoch)
  File "/home/idriver/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/keras/engine/training.py", line 1553, in fit_generator
    class_weight=class_weight)
  File "/home/idriver/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/keras/engine/training.py", line 1310, in train_on_batch
    check_batch_axis=True)
  File "/home/idriver/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/keras/engine/training.py", line 1030, in _standardize_user_data
    exception_prefix='model input')
  File "/home/idriver/anaconda3/envs/carnd-term1/lib/python3.5/site-packages/keras/engine/training.py", line 112, in standardize_input_data
    str(array.shape))
ValueError: Error when checking model input: expected lambda_input_1 to have 4 dimensions, but got array with shape (32, 1)
```

I include the code in the file, named clone.py. I have no idea how to solve this problem.

Evaluation:

The model is able to successfully navigate both track 1 and track 2 without much trouble.