Behavioral Cloning Writeup

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

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
* BehavioralCloning.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.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

I used NVIDIA model to train on both track 1 and track 2 for a total of 40 epochs.

The model includes ELU layers to introduce nonlinearity(model.py line 152 onwards), and the data is normalized in the model using a Keras lambda layer(model.py line 145).

2. Attempts to reduce overfitting in the model

Dropout layers are also used after every convolutional and fully connected layers to avoid over-fitting and give the model the ability to generalize(model.py line 153 onwards).

The model was trained and validated on different data sets using **train_test_split()** to use 80% for training and 20% for validation to ensure that the model was not overfitting(model.py line 28).

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, and also a learning rate of **1e-04** was set.

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, driving correctly on the turns and driving around the track in the opposite direction.

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

Model Architecture and Training Strategy

1. Solution Design Approach

My first step was to use a convolution neural network model similar to the the one used by comma.ai https://github.com/commaai/research/blob/master/train_steering_model.py, whilst it did pretty well, it was always stumped by a corner on track 1 by the water.
I tried in vain to collect more data assuming it would fix the issue but to no avail.
And on track 2 it was really abysmal, so I decided to chuck this model and try a new one.

The NVIDIA model worked really well on track 1 and on track 2 with a reduced speed of 15 kmph.
Didn't modify the model much except add the final layer to give out the single output.

Since it did pretty well for both the tracks, I now was curious if 1 single model would be able to drive perfectly on both the tracks.
So I modified the code so that it takes the data for both the tracks and had to increase epochs from the previous 20 to 40.
The only caveat is that after training on both tracks, the model seems to swerve from right to left on track 1, when before it was quite stable.
Even though it swerves, it is still very good at staying within the lanes of the road, including within the dashed lines around the corners.

At the end of the process, the vehicle is able to drive autonomously around **both** tracks without leaving the road.

2. Final Model Architecture

The final model architecture (model.py lines 142-181) consisted of 5 Convolutional layers and 3 fully connected layers.

Here is a visualization of the architecture (note: visualizing the architecture is optional according to the project rubric)

Model Summary =>

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
==================
lambda_1 (Lambda)                    (None, 66, 200, 3)        0
       lambda_input_1[0][0]
____________________________________________________________________________
__________________
convolution2d_1 (Convolution2D)      (None, 31, 98, 24)        1824
       lambda_1[0][0]
____________________________________________________________________________
__________________
dropout_1 (Dropout)                  (None, 31, 98, 24)        0
       convolution2d_1[0][0]
____________________________________________________________________________
__________________
convolution2d_2 (Convolution2D)      (None, 14, 47, 36)        21636
       dropout_1[0][0]
____________________________________________________________________________
__________________
dropout_2 (Dropout)                  (None, 14, 47, 36)        0
       convolution2d_2[0][0]
____________________________________________________________________________
__________________
convolution2d_3 (Convolution2D)      (None, 5, 22, 48)         43248
       dropout_2[0][0]
____________________________________________________________________________
__________________
dropout_3 (Dropout)                  (None, 5, 22, 48)         0
       convolution2d_3[0][0]
____________________________________________________________________________
__________________
convolution2d_4 (Convolution2D)      (None, 3, 20, 64)         27712
       dropout_3[0][0]
____________________________________________________________________________
__________________
dropout_4 (Dropout)                  (None, 3, 20, 64)         0
       convolution2d_4[0][0]
____________________________________________________________________________
__________________
convolution2d_5 (Convolution2D)      (None, 1, 18, 64)         36928
       dropout_4[0][0]
____________________________________________________________________________
__________________
dropout_5 (Dropout)                  (None, 1, 18, 64)         0
       convolution2d_5[0][0]
```

```
____________________________________________________________________________
__________________
flatten_1 (Flatten)                 (None, 1152)                0
        dropout_5[0][0]
____________________________________________________________________________
__________________
dense_1 (Dense)                     (None, 100)                 115300
        flatten_1[0][0]
____________________________________________________________________________
__________________
dropout_6 (Dropout)                 (None, 100)                 0
        dense_1[0][0]
____________________________________________________________________________
__________________
dense_2 (Dense)                     (None, 50)                  5050
        dropout_6[0][0]
____________________________________________________________________________
__________________
dropout_7 (Dropout)                 (None, 50)                  0
        dense_2[0][0]
____________________________________________________________________________
__________________
dense_3 (Dense)                     (None, 10)                  510
        dropout_7[0][0]
____________________________________________________________________________
__________________
dense_4 (Dense)                     (None, 1)                   11
        dense_3[0][0]
============================================================================
==================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
______________________________________________________________________
```

3. Creation of the Training Set & Training Process

To capture good driving behavior I did the following =>

- Drive 3 laps with centre lane driving on both tracks.
- Drive 3 laps with centre lane driving, in the opposite direction, on both tracks.
- Drive 1 lap were the car straightens from left and right side of the road on both tracks in both directions.

- Drive 1 lap exclusively around the corners of the track, on both tracks and in both directions.

Which were stored separately in **center**, **curves**, **straighten**, **center_jungle**, **curves_jungle**, **straighten_jungle** as well as **default** which includes the default data given by Udacity.

I also had 3 different data augmentation procedures =>
- Flip: Where I flip the image from left to right.
- Shift: Where I random include a translation along X axis in the range of -50 to 50 which corresponded to an angle shift of 0.4% of the X translation, and a translation along the Y axis in the range of -20 to 20.
- Bright: Increase or decrease the brightness of the image by a certain value in the range of 0.5 to 1.5
- None: Where no augmentation was done to the image.

Each time I used to randomly pick from which camera(left, center, right) as well as randomly pick the augmentation method(flip, shift, brightness, none) so as to give the model the ability to generalize and not be biased to certain cameras or conditions of the picture.