Behavioral cloning project

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
- runit.mp4 contains the video of a lap around the track in autonomous mode
- Current document is 'writeup_report.pdf' summarizing the results

Model Architecture and Training Strategy:

Model Architecture Snap-Shot:

```
Samples:  30478
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
cropping2d_1 (Cropping2D)        (None, 65, 320, 3)    0           cropping2d_input_1[0][0]
____________________________________________________________________________________________________
lambda_1 (Lambda)                (None, 40, 160, 3)    0           cropping2d_1[0][0]
____________________________________________________________________________________________________
lambda_2 (Lambda)                (None, 40, 160, 3)    0           lambda_1[0][0]
____________________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 10, 40, 16)    3088        lambda_2[0][0]
____________________________________________________________________________________________________
elu_1 (ELU)                      (None, 10, 40, 16)    0           convolution2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 5, 20, 32)     12832       elu_1[0][0]
____________________________________________________________________________________________________
elu_2 (ELU)                      (None, 5, 20, 32)     0           convolution2d_2[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 3, 10, 64)     51264       elu_2[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 1920)          0           convolution2d_3[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 1920)          0           flatten_1[0][0]
____________________________________________________________________________________________________
elu_3 (ELU)                      (None, 1920)          0           dropout_1[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 512)           983552      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, 50)            25650       elu_4[0][0]
____________________________________________________________________________________________________
elu_5 (ELU)                      (None, 50)            0           dense_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 1)             51          elu_5[0][0]
====================================================================================================
Total params: 1,076,437
Trainable params: 1,076,437
Non-trainable params: 0
```

More details on the chosen model:

Layer	**Details**
Convolution Layer 1	Filters: 16 kernel: 8x 8 Stride: 4 x 4 Padding: SAME Activation: ELU
Convolution Layer 2	Filters: 32 kernel: 5 x 5 Stride: 2 x 2 Padding: SAME Activation: ELU
Convolution Layer 3	Filters: 64 kernel: 5 x 5 Stride: 2 x 2 Padding: SAME
Flatten, Drop-out and ELU Activation	Dropout: 0.2
Fully Connected Layer 1	Neurons: 512 Dropout: 0.5 Activation: ELU
Fully Connected Layer 2	Neurons: 50 Activation: ELU
Fully Connected Layer 3	Neuron(s): 1

I tried several possible combinations of multiple Convolution layers and Fully connected layers and settled on 3 convolution layers and 3 Fully connected layers as I did not have access to AWS GPU instance and was working off my laptop CPU (limited computing capacity.

- Data was **normalized** and **mean centered** by using lambdas this is so that that the data is resilient to effects of environmental conditions such as lighting.
- **Convolution layers** were chosen as part of the training data constitutes a set if **three images** from the perspective of 3 cameras positioned on the the front of the car from a left, right and center perspective.
- **ELU activation** was chosen to introduce non-linearities.
- **Final layer** with 1 neuron was chosen as the steering angle is continuous (-1 → 1) and therefore no softmax function is required.

Reduce Over-fitting:

- **Dropouts** were added to prevent over-fitting.
- **ELU** activations were chosen to prevent over-fitting as well.
- Model was trained and validated on different data sets. 10 Percent of the training data was set aside for validation.

Parameters and Optimizer:

- **Adam optimizer** with a **mean square error loss function on accuracy** was chosen.
- **Epoch size** was chosen as 2
- **Batch size** was picked as 32
- **A multiplier of 6** was used for samples per epoch as for each row in the dataset, there are 6 images (3 – (left, center, right) 3 – (flipped left, flipped center, flipped right))

A very Low **Learning rate** was initially picked as 0.00001 as a parameter to the adam optimizer but but then I picked a higher learning rate of 0.0001 as I decided to reduce the number of epochs from 20 to 5 and eventually 2. The loss was very low and therefore I decided to keep these parameters.

Choice of Training Data:

- Multiple forward laps were done on the simulator on the left track and a few laps were done in the other direction to get a total of over 22 thousand * 3 (for left,center, right) training data points.
- As the model struggled in the portion of the lap prior, on and after the bridge, I used the simulator to get several rounds of forward and backward driving data on this portion of the track.
- I also drove a few laps where the car was intentionally navigated close to the edge and over the edges but was trained to quickly recover.
- In general, wherever the car seem to drive erratically in the simulation, multiple backward and forward drives were done on that portion of the track to get additional training data to improve performance.

In the generator:

- As the car has a left bias due to the anti-clockwise nature of the track, I added the left, center and right images flipped.
- To eliminate erratic turns, I added a **correction** of 0.2 to the steering angle of the left and right images and also the flipped left and right images.
- I also tried to threshold the steering angle with low and high values for eg: (low 0.5, high 23), (low 0.01, high 22) etc to prevent the car from making sudden turns off the

track but decided that providing the model with additional training data was better suited to keep the car from going over the track markings so those were eliminated.

Other Pre-processing:

- Images were cropped 70 pixels from the top to eliminate the data above the driving horizon and 25 pixels from the bottom to eliminate the hood of the car.
- I resized the image to reduce the the input image dimensions.

Eventually with the provided drive.py runner, in autonomous mode, the trained model.h5 was able to smoothly navigate the left track without the tires ever leaving the drivable portion of the track surface.