SELF-DRIVING CAR NANODEGREE

PROJECT 3
Behavioural cloning

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 obtained during training.
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road

Details about the model

The initial models were simple regression models. Some of the techniques used in CNN models with small number of layers were pooling and normalization on data. Training images were cropped at the top and bottom to get rid of unhelpful parts of the image. Other models that were used for the purpose was LeNet. This did not provide useful predictions as I could see the car was driving off the road even in straight lines. The probable cause could be due to the architecture of the model where pooling is an important part. I observed similar behaviour with the NVIDIA model where when I added pooling layers the performance got worse. I also tried using only just the center images, as expected it did not perform very well. But with more data it got better but never perfect.

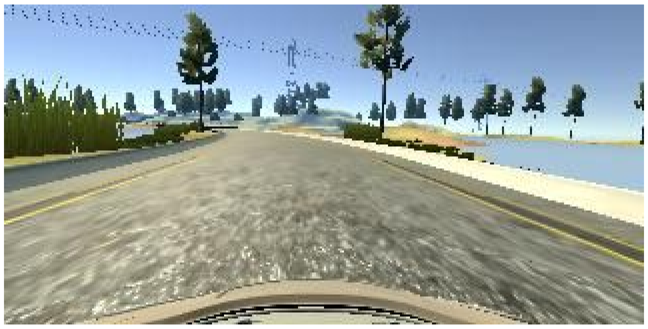
The final model is based on the NVIDIA’s deep learning model for self-driving cars. This model has a **normalization** layer at the beginning. This is followed by **6 convolutional** layers and then **4 fully connected** layers. The additional convolutional layer in my model made all the difference to my results because before this addition even with large amounts of data my model did not perform very well. The car use to go off track at certain points. The main reason was the model overfitting most of the time. This occurred even when I tried with different sets of data. Even after adding dropout layers the model improved considerably but not completely. So with this additional convolutional layer I was able to get rid of even the dropout layers. Dropout layers improved the performance only considerably. With the additional convolutional layer I was able to achieve better results and without any dropout layers. Adding a dropout layers made the model a little weaker, not considerably but a little. So I refrained from adding the dropout layer.

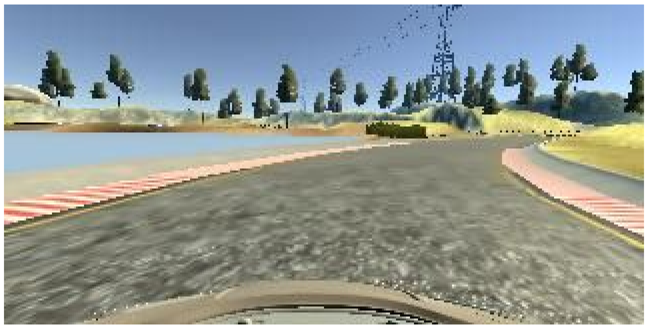
The addition of the convolutional layer was mainly based on the last project intuitions where adding a new layer at the end improved the model. So I employed the same intuition and applied that here and it worked, making the model much better.

The model is convolutional neural network with depths ranging from 24 to 128. Filter sizes were 3x3 for few and 5x5 for remaining layers.
The model includes RELU layers to introduce nonlinearity, and the data is normalized in the model using a Keras lambda layer. Adam optimizer was used in the network.

Training data

Training data included 3 laps of driving around track 1. It also included few recovery scenarios as well. The total size of the training set was about 29000 images, including left , right and center. Some of the images from the training set are as below.





Preprocessing: These images were cropped at the top and bottom. 50 pixels from the top and 15 pixels from the bottom. Normalization of the images using the Keras lambda layer. For some unknown reason flipping the images and augmenting the dataset did not work very well for my model. So I have not employed that technique.

Used the preprocessed data to build the model. Train and validation data split were 0.9 and 0.1. Model was trained for 5 epochs.