Behavioral Cloning

Model

For the project 3 of Self-Driving car nanodegree, I chose to start with the NVIDIA CNN used for end to end learning of self-driving cars. This model had been used to train a car for a very similar purpose, hence I found it to be a good place to start building my own model. The architecture of the model is shown in the figure below.

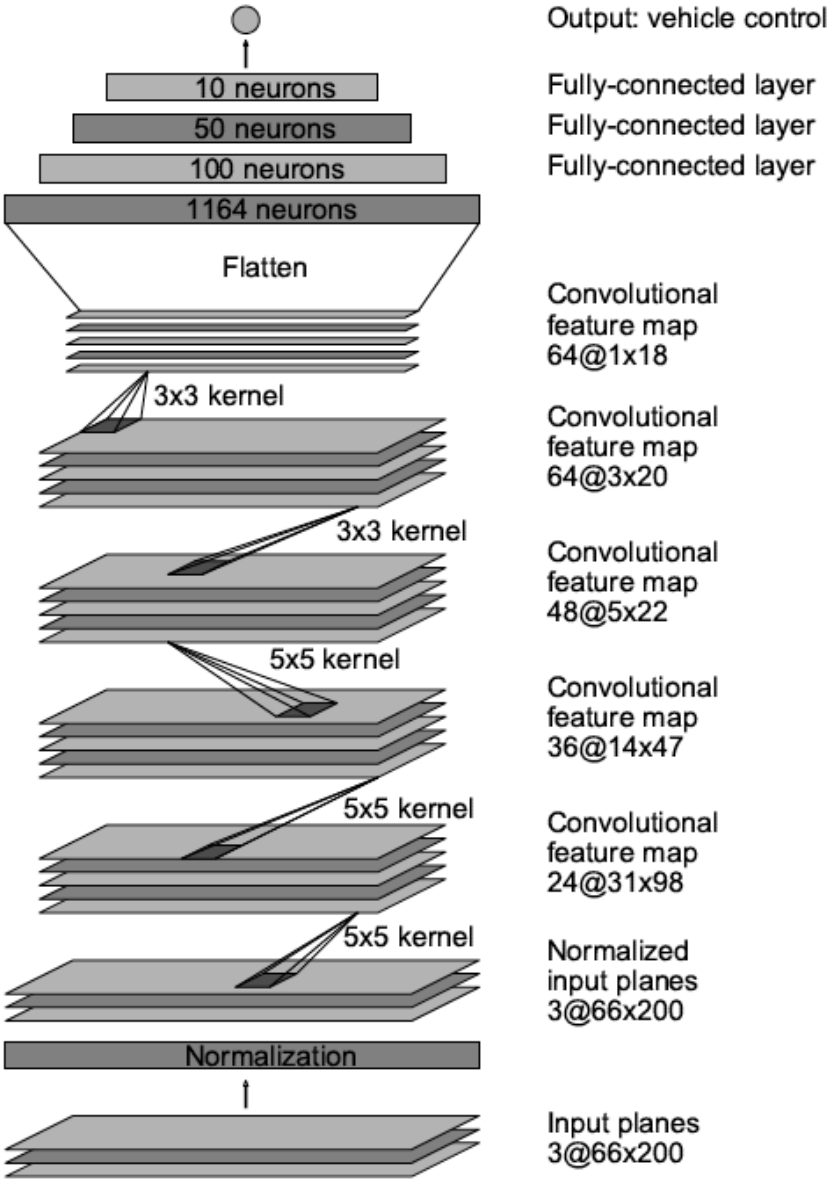


Fig1: NVIDIA CNN used for this project

(taken from the paper *'End to End Learning for Self-Driving Cars: Mariusz Bojarski et al'*)

At first, the images were resized to 100x200 pixels and then fed into the first layer of the model. There, additional 34 pixels were chopped from the top thereby making the image compatible for the NVIDIA CNN. Chopping 34 pixels from the top removed extraneous information such as the sky etc., which makes the network concentrate on the tracks and the position of the car with respect to the edges of the tracks. After chopping, the image pixels were normalized to lie fro -0.5 to 0.5 instead of 0 – 255. This made the model focus on the features instead of the colors of the images. This was followed by a Batch Normalization operation on the image. The next five layers consisted of convolutions each followed by a dropout layer having keep probability of 50%. Each convolution had an

Exponential Linear Units (ELU) as the activation function instead of conventional RELU function to speed up the learning process.; the first three convolutionlayers had a kernel size of 5x5 with a 2x2 stride and the next two layers had a 3x3 kernel with no stride. Following this I added three fully connected layers with the final output being the steering angle. The model is defined from lines 112 to 152 in the code.

Overfitting

To prevent over fitting, I added a dropout layer after every convoluted layer with a keep probability of 50% (line 124 – 150).
Also, I split the generated data into training and validations set using a split ratio of 80-20 %. The learning loss for both training and validation set were similar, hence giving some confirmation that the model was generalized. Furthermore, the model was tested on the simulator and the car was keeping in between the lanes.

Model Parameter Tuning

The model was trained using Adam optimizer (line 177) and the learning rate was kept at 0.001. Also, a batch size of 32 was used for a custom built generator (using an nuygen's code). Keeping the epoch size at 10 seemed to give best results.

Appropriate training data

Histogram was plot to check the distribution of steering angles:

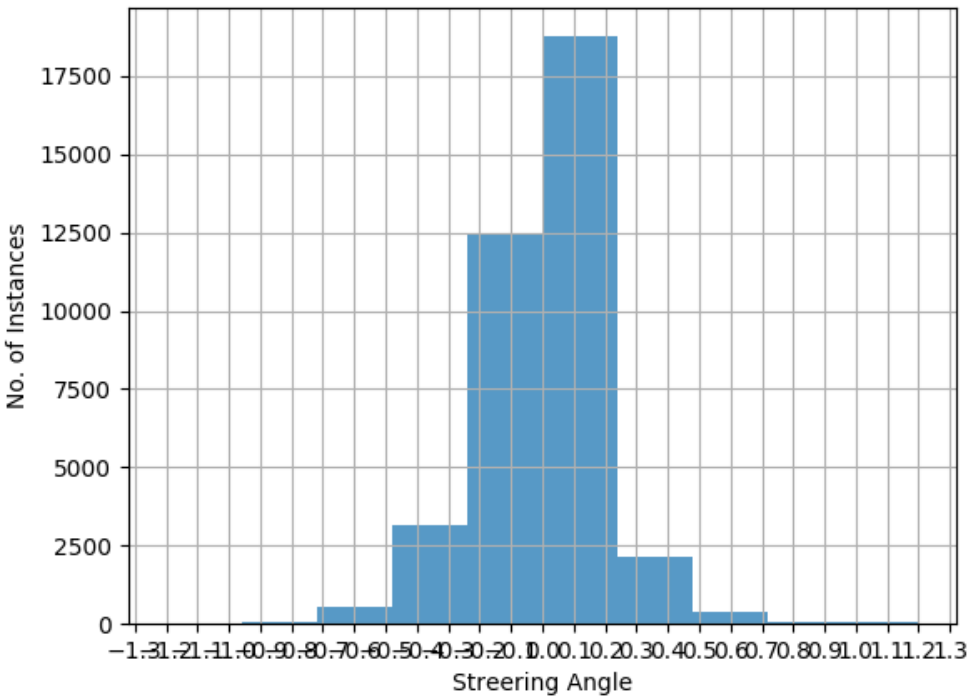


Fig 2: Original Steering angle distribution

Clearly, the graph shows that majority of the steering angles were zero or close to zero (indicating that the car was travelling straight for most of the time). To remove this bias towards zero steering angle a filter was put to keep only 26% of the zero steering angle data (line 65). This value was achieved via trial and error. Also, a steering

angle of 0.25 was added to the left camera image while the same was subtracted from the right camera image. Training with this data gave reasonable results but I felt the need to generate more data to improve performance. I recorded the data to drive around the track myself. I made sure I reduced the speed around the corners to a very low value so that I generate more data for these cases. Also, I recorded data for cases where the car begins to slide off the road. I generated a total of 11,474 images for each camera (including the original dataset) and began training my model.
The final steering angle distribution looked like this:

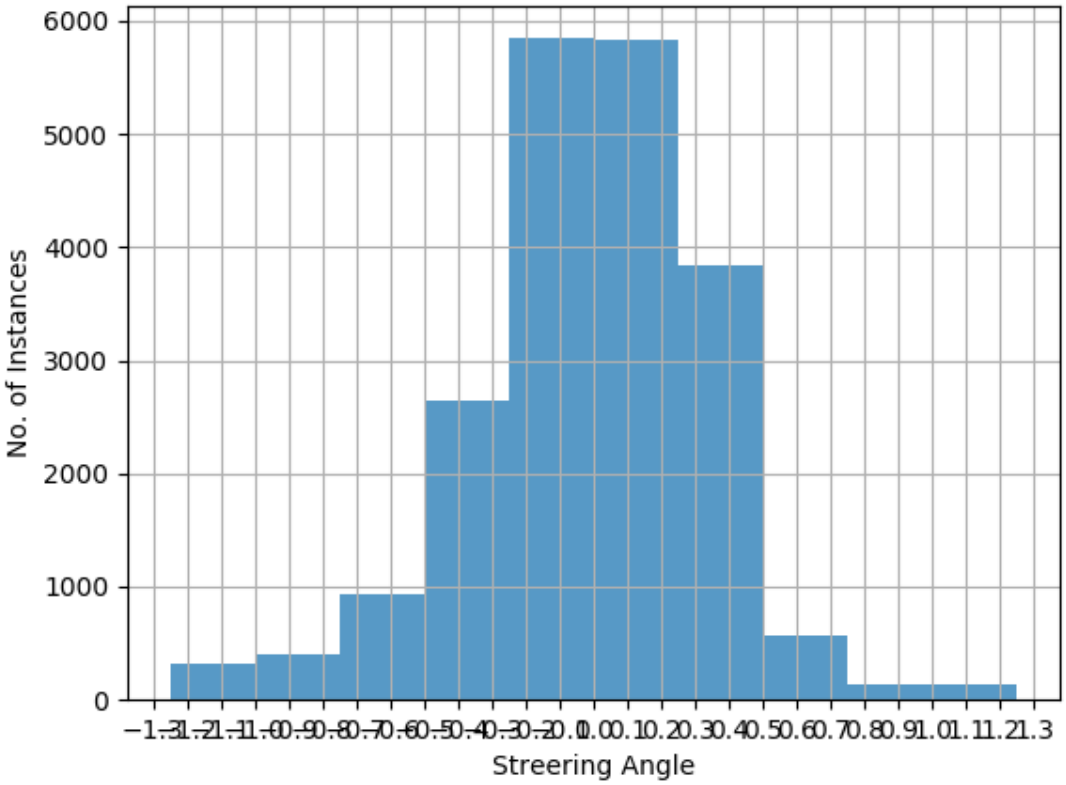
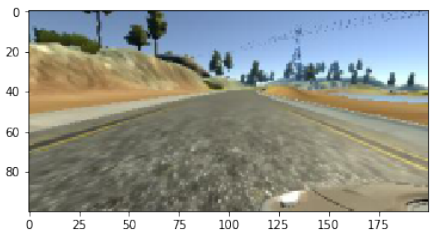


Fig 3: Final steering angle distribution

The final set of trained weights gave a decent performance and the car was able to safely drive around the track.

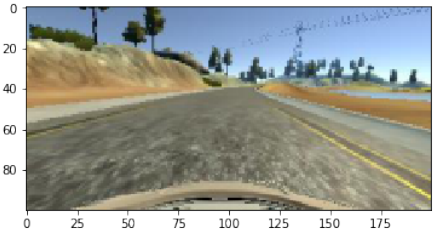
Examples of images from dataset:

Left Camera



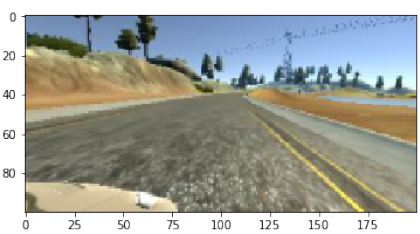
Predicted Steer : 0.15726504

Center Camera



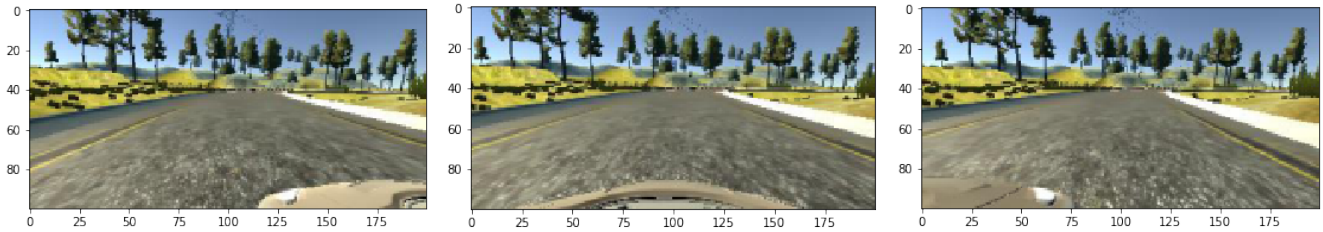
Predicted Steer : -0.07658526

Right Camera



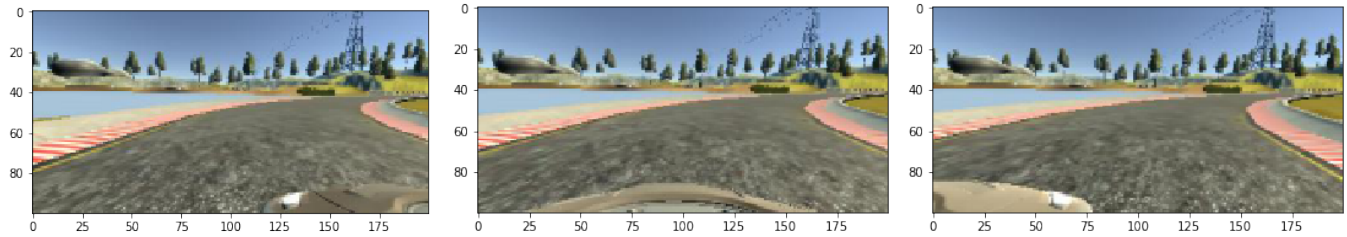
Predicted Steer : -0.36423635

Left Camera **Center Camera** **Right Camera**



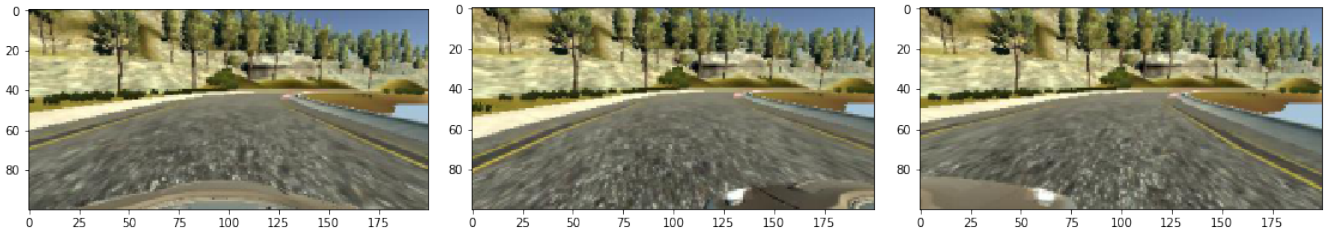
Predicted Steer : 0.18463141 **Predicted Steer : 0.01494177** **Predicted Steer : -0.18790412**

Left Camera **Center Camera** **Right Camera**



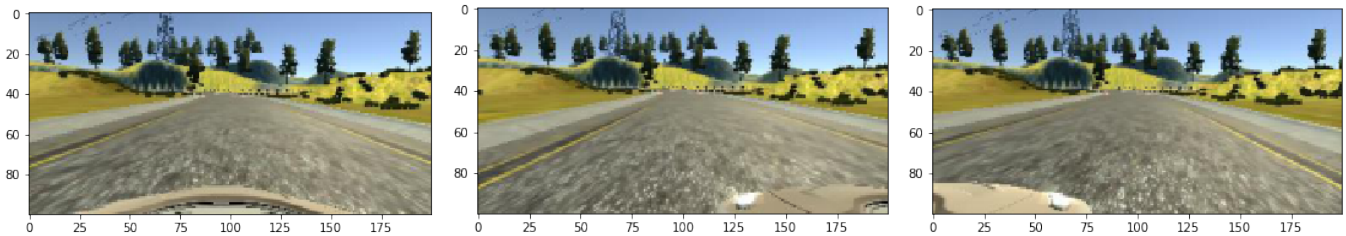
Predicted Steer : 0.3095414 **Predicted Steer : 0.17415804** **Predicted Steer : 0.00795034**

Left Camera **Center Camera** **Right Camera**



Predicted Steer : 0.18098289 **Predicted Steer : -0.03195026** **Predicted Steer : -0.25459**

Left Camera **Center Camera** **Right Camera**



Predicted Steer : 0.06273354 **Predicted Steer : 0.19997102** **Predicted Steer : -0.19256812**