1. CarND-Behavioral-Clone-P3 Project Report

1. Submissions

1. clone.py: Contains the script to create and train the model.

2. drive.py: The script for driving car in autonomous mode.

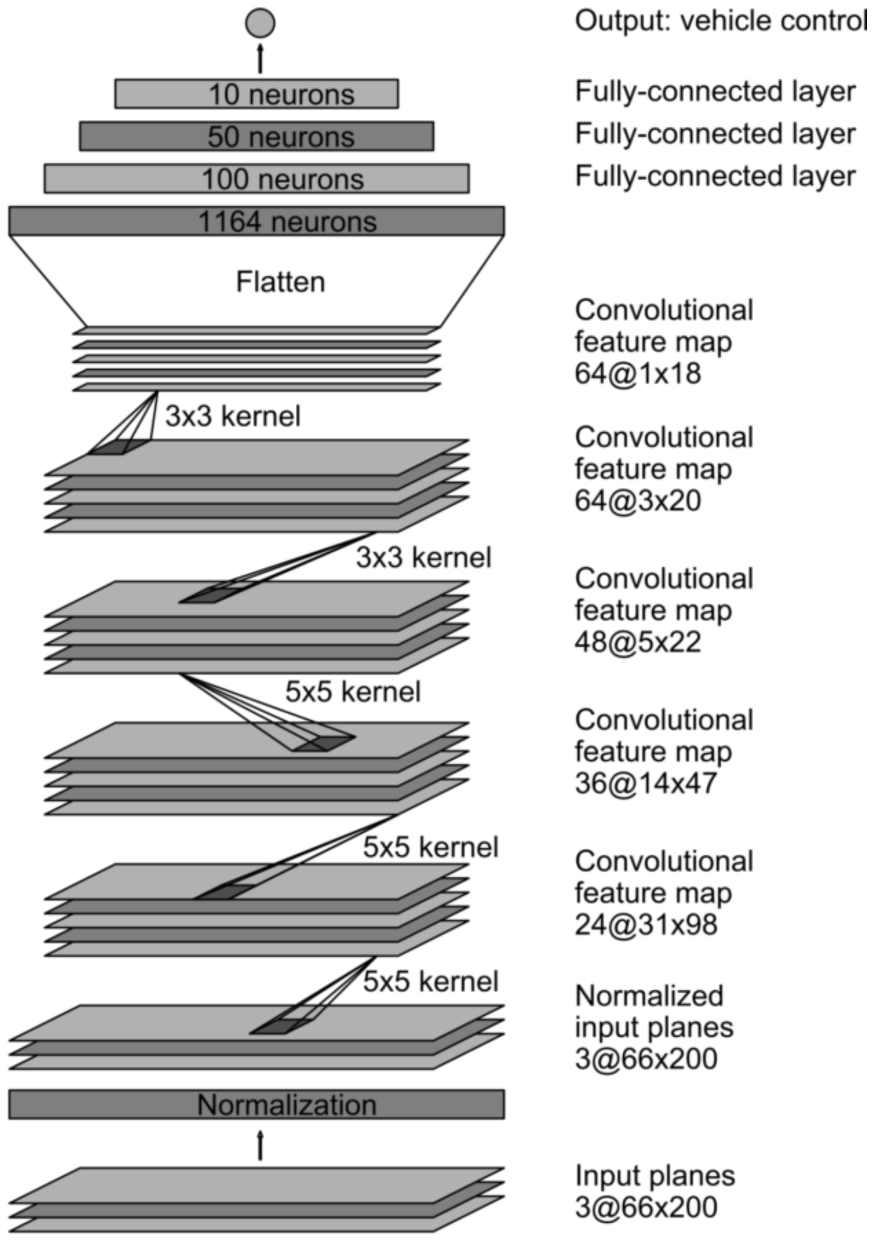
3. model.h5: The trained CNN model.

4. writeup_report.pdf: This project report file.

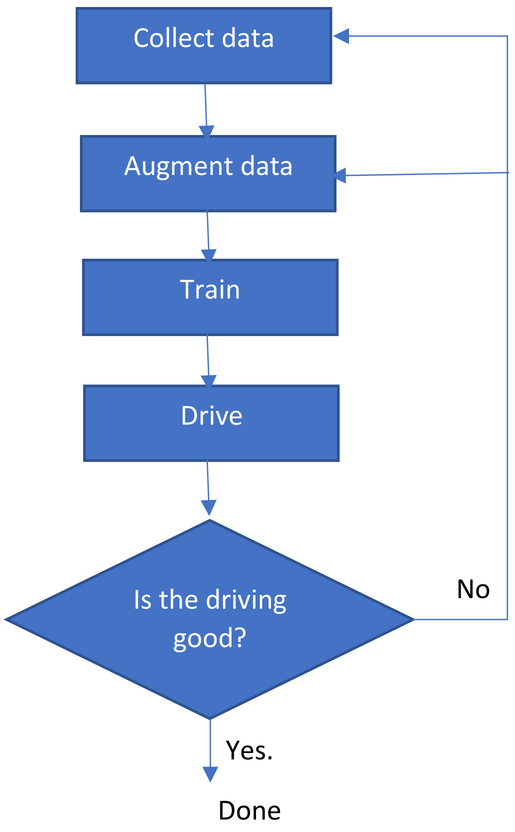
5. run1.mp4: Recorded video of the autonomous driving of one complete round of the track.

2. Model Architecture and Training Strategy

I tried various models of CNN. I created my own. I took multiple variations of the NVIDIA architecture. After days of experimenting, I found out that the NVIDIA architecture gives me best results. Also, all need was only 3 epochs. Hence, after that I fixed my CNN to that one and worked on other aspects of the project. The following image shows the NVDIA CNN architecture, which is described in: https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf.


Output: vehicle control
10 neurons
Fully-connected layer
50 neurons
Fully-connected layer
100 neurons
Fully-connected layer
1164 neurons
Flatten
Convolutional feature map 64@1x18
3x3 kernel
Convolutional feature map 64@3x20
3x3 kernel
Convolutional feature map 48@5x22
5x5 kernel
Convolutional feature map 36@14x47
5x5 kernel
Convolutional feature map 24@31x98
5x5 kernel
Normalized input planes 3@66x200
Normalization
Input planes 3@66x200

The following diagram shows how I created a good model which could run the track without going off road.



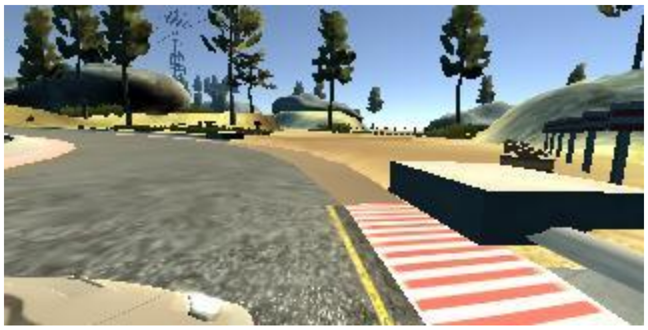
My strategy involved the following steps.

1. Collect data

This step involves the real driving and collecting data. I used the following driving data.

i) The sample training data provided in the course. I used this data as this data comes free of cost and I don't have to drive myself to get this one.

ii) Short drives near one of the curves. What I found was my car was drifting off road on the right side in one of the curves. I used more training at this place. I drove on the right-hand side of the track most of the time. And in the angle measurement, I added -0.5 to force a left turn. This data was very helpful.



The above image shows the point, where the car was drifting to the right-side shoulder. The markings at the end of the red-and-white shoulder is not clear. Hence, more data was needed for this case.

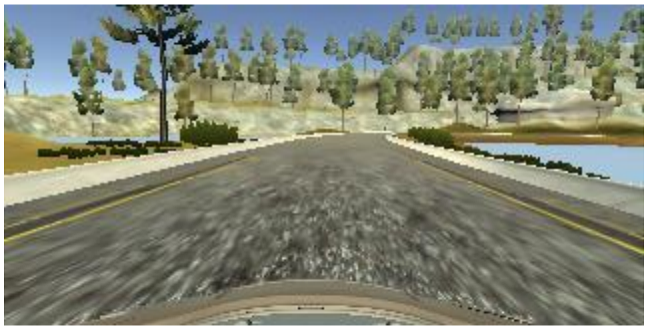
The following data was not very useful. The following data were collected. But, during my training and testing, they didn't give any sufficient advantage. Only the above mentioned two set of data were needed.

i) Driving the complete track. As I already used the sample track data, this was not very important.
ii) Driving backward to collect mirror-like data. Again, this data didn't give me any sufficient improvement.
iii) Driving in the second track. The second track was extremely hard for me to drive. I was feeling nauseous after driving it. I couldn't ever finish that track. And whatever portion, I drove was not very smooth. Hence, I didn't include this data.
iv) Recovering Laps: All the recovery laps, I needed was what I mentioned previously. I drove on the right edge of a specific part of the road and during reading of the angle measurement, I added -0.5. There was no other recovery lap needed.

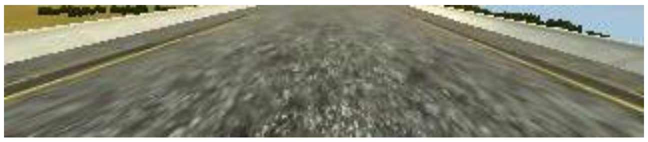
2. Preprocessing data

The only preprocessing done on the data was to clip top 70 rows and bottom 25 rows. The top rows had the landscape which was not adding to the road understanding of the car. Removing the top 70 rows removed extra noise from the data.

And the two 25 rows were mainly the road and the front portion of the car. Removed it helped the reduce the size of the image. The below image shows before and after of the cropped image.



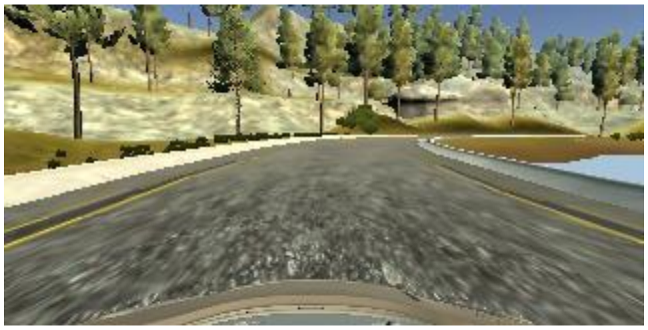
After cropping, it looks like this:



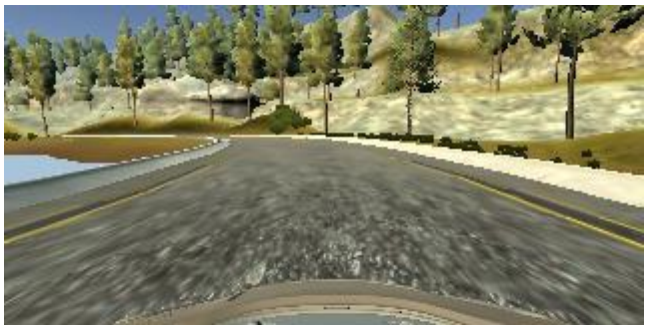
3. Augment data

The following different types of augmentation of data was done:

i) Use mirror image: By using a mirror image of the data and the opposite turn angle, we get double the data. It effectively cancels any kind of right or left turn bias we have introduced. An example is the following:

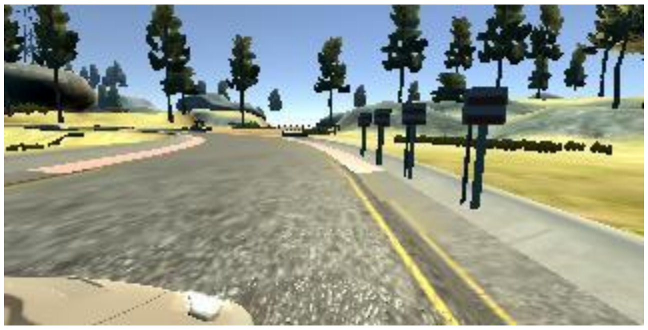


The mirror image is:



ii) Use left and right camera image: The image is captured using left, right and center cameras. But we train only for center image. If we use the right and left images and appropriately, adjust the turn-angle, then we get three times the data. By experiment, I found that a value of 0.17 is appropriate for tuning the left and right images. Hence, for the right image, I subtract 0.17 (to effectively turn it slightly left). Similarly, for left image, I add 0.17.

iii) Add extra turn: As described earlier, I drove on the right side in curved roads to capture data. Then during processing, I added -0.5 to force a big left turn on those data. A typical image from that dataset would look like this:



In this image, the car is on the right side of the track. In this situation, I force the data to turn further left that the value captured by the tool.

iv) Ignoring straight drives: To reduce the data size, I removed all the zero-angle (straight drive) from the training data. This, I did, only on the extra drive I did. For the regular drive data, I kept the zero-angle values.

4. Train

This is the time, when I train the data with the above mentioned CNN model.

i) RELU: To introduce non-linearity, I added RELU layers.

ii) Dropouts: To avoid over-fitting, I added dropout equal to 0.25 after each Convolution and flat layers.

At the end of the training, I capture the model.h5.

5. Autonomous Driving

This is where I use the model.h5 and run the car in autonomous mode.

3. Conclusion

My car successfully completed on lap of the track. I didn't try it on the 2nd track.

I used NVIDIA CNN architecture with 0.25 dropout after each layer.

I had a total of 48126 images from the sample training data.

And I had 2526 more images from my custom driving after removing zero-angle data.

Used 80/20 training vs validation split.

4. Usage of fit_generator

I used generator in between. If one looks at my git history, the usage can be seen. Since, I am working on an Azure VM with 56GB memory and since, the data set I had was reasonably small, I was not getting any advantage of using generator. Hence, I abandoned it.