Behavioral Cloning Project 3

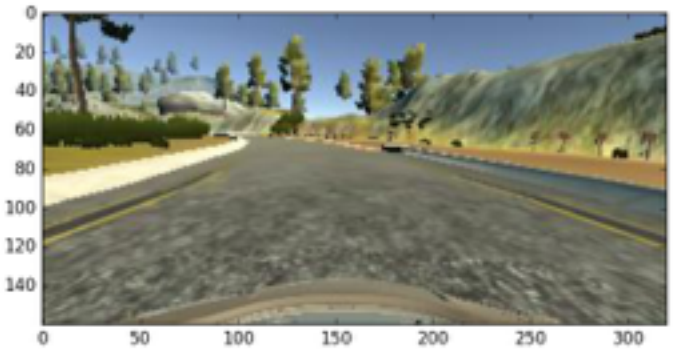
By Ahmad Chatha

Introduction: In this project, we used convolutional neural networks to clone driving behavior. We trained, validated and tested a model using Keras. The resulting model outputs a steering angle which is used to drive the car around the track.

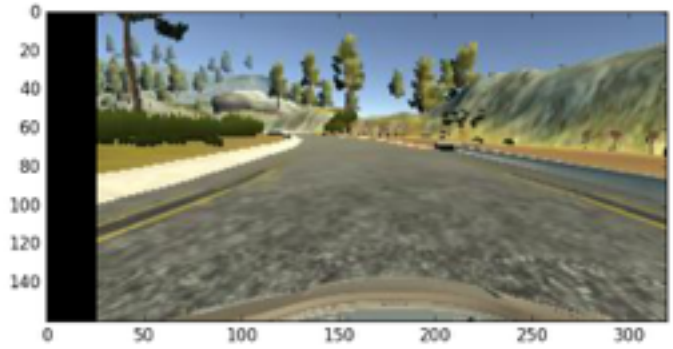
Data Preprocessing: In addition to the dataset provided with the assignment, i also generated my own data using the simulator. Before preprocessing the data, we filter it using the steering angles. Most of the images with steering angle equal to 0 are thrown away. This turned out to be extremely useful because without it i wasn't able to go around the track. This makes sense because we don't want the training data to include lots and lots of images with 0 steering angles because then the car will learn to go mostly straight which is exactly what was happening in my case.
Out of 8967 total images, only 5028 were used for training and validation.

The images were also cropped and resized to remove unnecessary details such as the sky which is mostly useless for the purposes of learning how to drive. Finally, upon the recommendation of other students, i randomly shifted, flipped and changed the brightness of the images. Following sequence of images shows the pre-processing step:

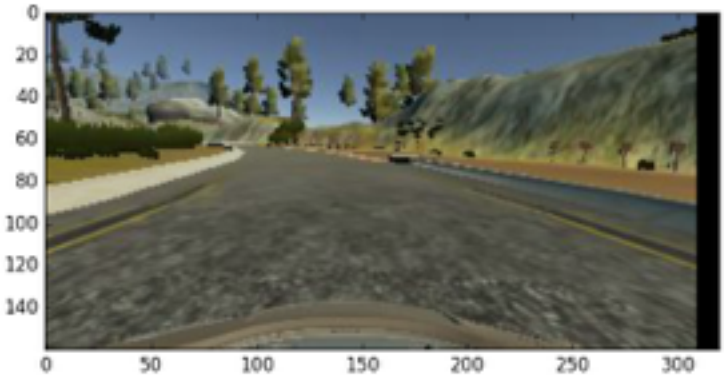
Original



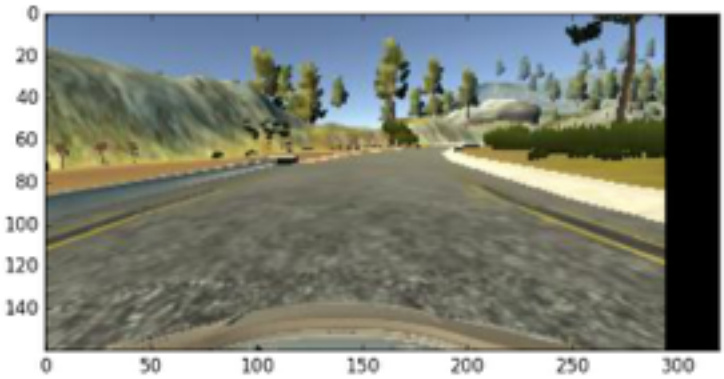
After Shift



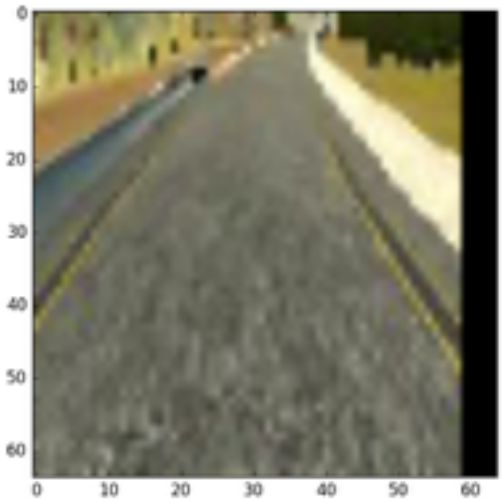
After Flip



Changing Brightness



Cropping and resizing



The biggest help was provided by reducing the number of images with 0 steering angle and the cropping and resizing steps. Also, the left, center and right images by randomly chosen and an angle offset was used for left and right images.

Network architecture:

As suggested by the assignment the model in the NVIDIA paper was used. The main difference was really the input. As opposed to previous projects, this time we are doing regression instead of classification. The network as roughly 5 million parameters. Following table (taken from the Keras summary) shows the structure:

Layer (type)	Output Shape	Param #	Connected to
lambda_1 (Lambda)	(None, 64, 64, 3)	0	lambda_input_1[0][0]
convolution2d_1 (Convolution2D)	(None, 32, 32, 24)	1824	lambda_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 31, 31, 24)	0	convolution2d_1[0][0]
convolution2d_2 (Convolution2D)	(None, 16, 16, 36)	21636	maxpooling2d_1[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 16, 16, 36)	0	convolution2d_2[0][0]
convolution2d_3 (Convolution2D)	(None, 8, 8, 48)	43248	maxpooling2d_2[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 8, 8, 48)	0	convolution2d_3[0][0]
convolution2d_4 (Convolution2D)	(None, 8, 8, 64)	27712	maxpooling2d_3[0][0]
maxpooling2d_4 (MaxPooling2D)	(None, 8, 8, 64)	0	convolution2d_4[0][0]
convolution2d_5 (Convolution2D)	(None, 8, 8, 64)	36928	maxpooling2d_4[0][0]
maxpooling2d_5 (MaxPooling2D)	(None, 8, 8, 64)	0	convolution2d_5[0][0]
flatten_1 (Flatten)	(None, 4096)	0	maxpooling2d_5[0][0]
dropout_1 (Dropout)	(None, 4096)	0	flatten_1[0][0]
dense_1 (Dense)	(None, 1164)	4768908	dropout_1[0][0]
dropout_2 (Dropout)	(None, 1164)	0	dense_1[0][0]
dense_2 (Dense)	(None, 100)	116500	dropout_2[0][0]
dropout_3 (Dropout)	(None, 100)	0	dense_2[0][0]
dense_3 (Dense)	(None, 50)	5050	dropout_3[0][0]
dropout_4 (Dropout)	(None, 50)	0	dense_3[0][0]
dense_4 (Dense)	(None, 10)	510	dropout_4[0][0]
dense_5 (Dense)	(None, 1)	11	dense_4[0][0]

Total params: 5,022,327
Trainable params: 5,022,327
Non-trainable params: 0

Training: The dataset was shuffled, preprocessed (as described earlier) and split into training and validation sets through a 80/20 split.

Total samples: 5028
Training size: 4022
Validation size: 1006

As mentioned by the assignment, python generators ('yield') was used to generate/load the image son the fly. # of epochs was set to 10. Batch size was 32. Adam optimizer was used with a learning rate of 0.0001. The loss was mean squared error. The validation loss turned out to be 0.024. To avoid overfitting, dropout was also used among dense layers. The model was trained on a CPU because the dataset was not that huge. On Mac it took around 10 minutes for 10 epochs.

Results:

The car is able to go around track 1 multiple times with out any problems. It remains mostly stable and is able to take sharp turns especially when approaching dangerous corners.

The performance on track 2 is not as good as compared to track 1. It is able to go around most of the lap but gets stuck against the boundary eventually. I think its because of the shadows. I never augmented my training data to account for shadows. Despite that, i was genuinely surprised by its performance because the track looks significantly different. I believe with a little bit of more image processing we can make it go around track 2.

Conclusion: This project took a lot of time as compared to previous projects. I spent a lot of time collecting the data and training the model to learn the track slowly. The sharp turn right after the bridge gave me the most trouble. The way i solved it was by going around that corner a lot of time at slow speeds to capture more training data so that the model can learn that corner. Another important step was removing images with 0 steering angles. That step helped improve the ability of the car to take sharp turns.

In short, this project taught me the importance of training data. The data on which you train is sometimes more important than the algorithm itself. Moreover, we have to understand how to pre-process the data and how much to use for the training process.