The technique of Behavioral Cloning : Self-Driving Car Simulation

Kolli Venkata Madhukar
Indian Institute of Technology, Bhilai
Raipur, India
kollim@iitbhilai.ac.in

Abstract

Self-driving cars are set to revolutionize transport systems the world over. Self-driving cars are technologically a reality and in the next decade they are expected to reach the highest level of automation. The paper includes a description of a project which is a step in the field of Computer Vision [2] and Self-Driving Cars. The project makes use of technique of Behavioral Cloning to train the model to drive a car on it's own in a simulation environment. It includes the steps for generation of data, pre-processing of data and training the Convolutional Neural Networks. The investigation focuses on the detailed analysis of the project.

Keywords : Self-driving Cars, Simulations, Behavioural Cloning, Machine Learning.

1. Introduction

Self-driving cars are technologically a reality and in the next decade they are expected to reach the highest level of automation (Level 5 according to SAE standard J3016) in which the *"automated driving system [takes care] of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver".* Although there has been a lot of advancement but the truth is that even today the most advanced of the self-driving systems work only in an extremely limited set of environments. There are many conditions and cases that are still not handled and that might lead to accidents.

1.1. Simulation Environment

In the latest years, with computer technology advancing fast, simulation began to be used more regularly for this kind of projects. Simulations are safer, more efficient, and cheaper than live testing on real vehicles. They also al- low testing more scenarios than those that would be pos- sible with real world testing, as well as testing dangerous situations to involve humans. Therefore, testing complex systems in virtual worlds is the ideal solution to validate code quickly, with more possibilities, cheaply and with minimum risk. The simulation environment that is used for the project is "Udacity Self-Driving Car Simulator" which includes both training and autonomous mode. The simulator is available for major platforms (Linux / Windows / MacOS). The track is a single lane track with multiple turns, confusing three-ways, sharp corners, exits, entries, bridges, partially missing lane lines and changing light conditions. The environment is still not as that of real conditions as it does not have the traffic and surprise elements, like, child coming in front of car suddenly, etc.

1.2. Behavioral Cloning

Controlling complex dynamic systems requires skills that operators often cannot completely describe but can demonstrate. Behavioural cloning is the process of reconstructing a skill from an operators behavioural traces by means of Machine Learning techniques. In this project, the behaviour of driver driving the car in the training mode is cloned and thus self driving model is made that can drive a car.

2. Data Generation and Pre-Processing

The data is collected in training mode in the simulator. The simulated car is equipped with three cameras, one to the left, one in the center and one to the right of the driver that provide images from these different view points. The training track has sharp corners, exits, entries, bridges, partially missing lane lines and changing light conditions. An additional test track exists with changing elevations, even sharper turns and bumps. It is thus crucial that the Convolutional Neural Network (CNN) [3] does not merely memorize the first track, but generalizes to unseen data in order to perform well on the test track. The model developed here was trained exclusively on the training track and completes the test track.

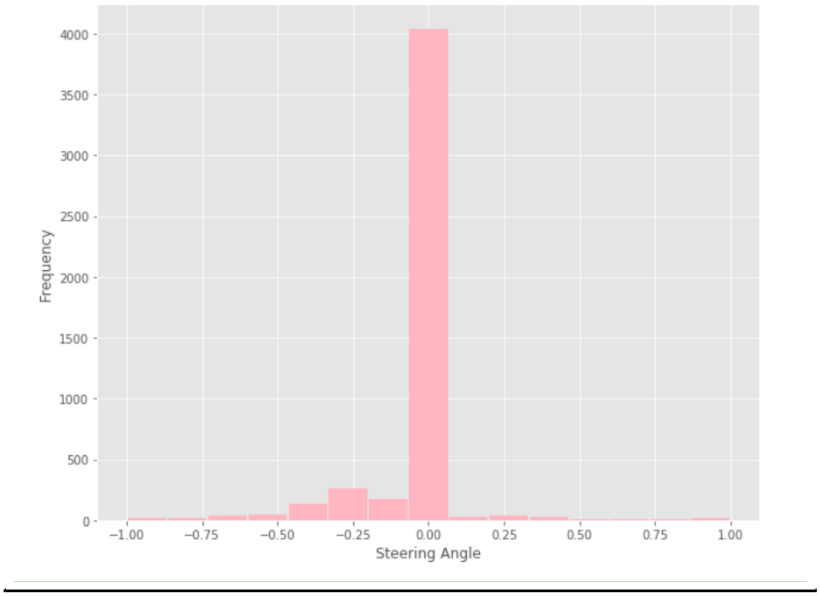

Figure 1. Frequency of different Steering angles in training data

2.1. Skew and Bias of Dataset

The main problem lies in the skew and bias of the dataset. The histogram (Figure 1) shows the steering angles recorded while driving in the middle of the road for a few laps. This is also the data used for training. The left-right skew is less problematic and can be eliminated by flipping images and steering angles simultaneously. However, even after balancing left and right angles most of the time the steering angle during normal driving is small or zero and thus introduces a bias towards driving straight. The most important events however are those when the car needs to turn sharply.

Without accounting for this bias towards zero, the car leaves the track quickly. This problem was counteracted by purposely letting the car drift towards the side of the road and by starting recovery in the very last moment. However, the correct large steering angles are not easy to generate this way too, because even then most of the time the car drives straight, with the exception of the short moment when the driver avoids a crash or the car going off the road. But this way gives a satisfactorily good and generalized data for training. The main solution to the problem was generation of artificial data which is explained in detail in next subsection.

2.2. Artificial Data Generation

After selecting the final dataset of images we augment the data by adding artificial shifts and rotations to teach the network how to recover from a poor position or orienta- tion [1]. Artificial data was generated by randomly selecting some of the left and right camera views and treating it as if it were the center camera view and the corresponding steering angle correction was made this ensured that the training data contains the datapoints for the situations in which the car is not straight and tries to align itself to the road and then drives straight. The correction constant can be treated as a hyper-parameter for the training. The value 0.2 units gave the best results for the project environment which can be tuned according to the corresponding project environment.

2.3. Data Correction and pre-processing

The image data obtained from the three cameras on the simulated car was very raw and contained the information that is not needed and can be removed like the images included the sky and the bonnet of the car which can be removed safely as we do not require that for prediction of the steering, throttle, brake and speed of car. Every image is flipped and the corresponding steering angle is changed accordingly so that we can remove the left-right skew. All the images were brought down to size (76, 280) that was expected by the model. The pre-processing steps included the normalization of the image data which made the computation faster and results a bit more accurate.

Layer (type)	Output Shape	# of Parameters	Activation
Convolution 2D	(None, 37, 139, 24)	672	ReLU
Convolution 2D	(None, 18, 69, 36)	7812	ReLU
Convolution 2D	(None, 8, 34, 48)	15600	ReLU
Convolution 2D	(None, 6, 32, 64)	27712	ReLU
Convolution 2D	(None, 4, 30, 64)	36928	ReLU
Dropout	(None, 4, 30, 64)	0	-
Flatten	(None, 7680)	0	-
Dense	(None, 100)	768100	ReLU
Dense	(None, 50)	5050	ReLU
Dense	(None, 10)	510	ReLU
Dense	(None, 1)	11	-

Table 1. Layers of the model

3. Model Architecture

The project includes Deep Learning model which is made by training multiple Convolutional Layers, Dropout Layer, and fully-connected layers. The model predicts the steering angle for given input image and simulator receives it through a Flask server. There is a pre-defined protocol of communication between the model and the simulator. The simulator sends the images at every instant to the model and the model returns the values of steering angle at every instant that is evaluated on the basis of the received image.

3.1. Model Insight - Details of Layers

The layers contain four convolutional layers, one dropout layer, flatten layer, and four dense layers. The model is used for regression hence the last layer does not contain any activation function like softmax, etc. and only contains one neuron. The details of the layers are shown in [Table 1]. Adam optimizer was used for training.

4. Conclusions and Results

At first, without the data augmentation and generalization(i.e. the flipping of images from left to right and vice-versa ,and adjusting the corresponding angles) the car passed the training track but failed miserably for the test track. The model was a overfit to the training data. After the data augmentation and generalization model success- fully passed the training track and test track. Further tuning the parameters of the angle correction in the augmentation section of the code and retraining the network for about 30 epochs fixed the issue and the car mastered the test track. The visual demonstrations of the simulator running in autonomous mode are shown below.

5. Acknowledgement

Author gratefully thank Udacity for providing their self-driving car simulator with MIT License that allowed them to use it in their work.

References

[1]M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[2]A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[3]Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989.

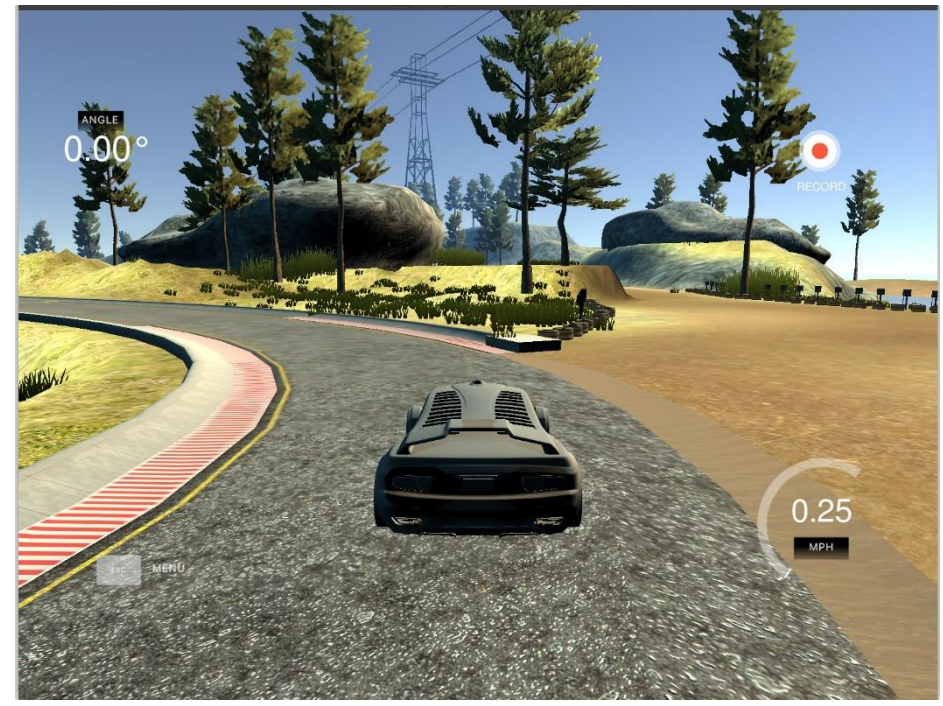

Figure 2. Image of the simulator and the car running in autonomous mode

[4] Kaspar Sakmann. Behavioral Cloning — make a car drive like yourself , *Convolutional Neural Network that clones human driving behavior in a simulator*, December 6, 2016. url : https://medium.com/@ksakmann/behavioral-cloning-make-a-car-drive-like-yourself-dc6021152713

[5] MIT Technology Review. The Open-Source Driving Simulator That Trains Autonomous Vehicles *CARLA*, November 16, 2017. url : https://www.technologyreview.com/s/609503/the-open-source-driving-simulator-that-trains-autonomous-vehicles/

[6] Udacity. self-driving-car-sim *A self-driving car simulator built with Unity*, February 10, 2018.
url : https://github.com/udacity/self-driving-car-sim