Behavioral Cloning

Report on Udacity SDCND Project No. 3, Nov. cohort 2016

Albert Killer, Feb 2017

1 Initial situation and assignment

On basis of recorded driving data created with the Udacity SDCND Simulator the goal of this project is to design a model which processes the image data and uses an artificial Neural Network (NN) to generate correct steering predictions. Following this, the simulator is able to use the trained model to drive along a virtual track without any human input in an adequate manner, which is considered to be safe.
The recorded data set contains approx. 24,000 front camera images (8,036 of each center, left and right camera view), taken while driving several laps (cp. Figure 1 for an example). Each of those 8,036 situations is labeled with an respective steering angle, ranging from -1.0 (most left / -25°) to 1.0 (most right / 25°) and provided in a CSV file.

Figure 1: Original data set provides 3 image samples for each steering angle.

Hardware setup and software environment for developing and training:

- Python 3.5.2 (Anaconda 3) with TensorFlow (Keras) using PyCharm Community Ed.
- CUDA 8.0.44 and cuDNN 8 to run code on a Nvidia GeForce GTX 1070 GPU
- Linux Mint 18 Sarah
- Intel i7-6700K CPU with 16 GB RAM

2 Creating data

The first approach consisted of directly importing the entire image data to memory followed by first test drives using a simple fully connected neural net. After implementing the first augmentation techniques to improve training it was soon pointed out that the use of an image data generator is very useful, due to memory limits. Two generators, one for training and one for validation data, were set up in order to generate batches of image data with real-time augmentation, which then can be fed to the model.
By applying image processing methods randomly the augmentation of training data can be easily adjusted and multiplied for testing by a single value. As by Figure 2 the majority of the images were linked to near zero steering values, the left and right camera view images where used to reduce bias on zero steering. This was achieved best by adding default offset values of +/- 0.2 to the steering angles of every left and right view image.

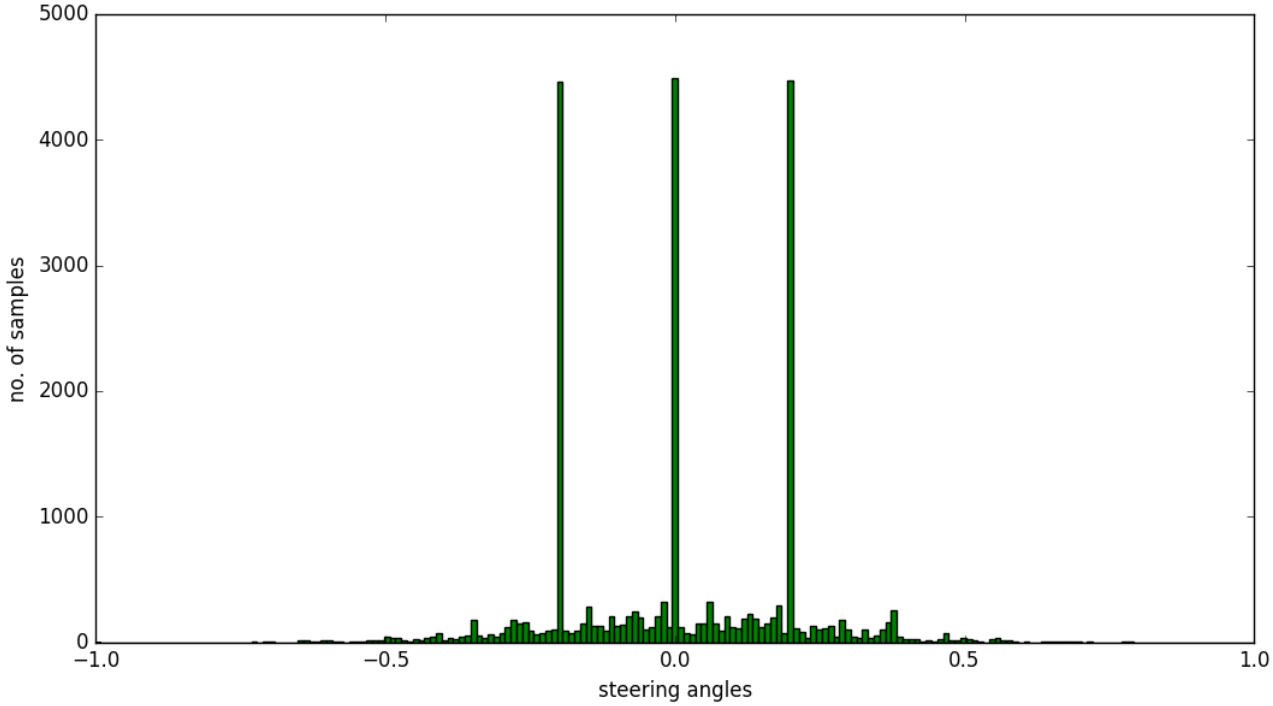

Figure 2: Compensate zero steering bias of original data set by adding steering offset.

In addition every second generated image was flipped to reduce the all over bias to one side, as the recorded track was circle-like, asking for steering mainly in one direction. The brightness of each image was manipulated randomly to create further diversity. The basic idea of preprocessing training data is to make crucial features more accessible and at the same time reduce obliviously insignificant attributes. Therefore large parts of the horizon (60 px) and the bottom of the images (25 px) were cropped to set focus on the interesting parts of the road ahead. Finally every image was downsized from the original 320x160 to 64x64 pixels (Figure 3).
The validation set was created by a second generator, unlike the training data, only applying resizing to fit the validation images to the input expected by the model. This also ensures that the model's output is validated on data as close as possible to the "real world". The image arrays were not being shuffled before training as the data stream was picked in a random fashion within the generator.

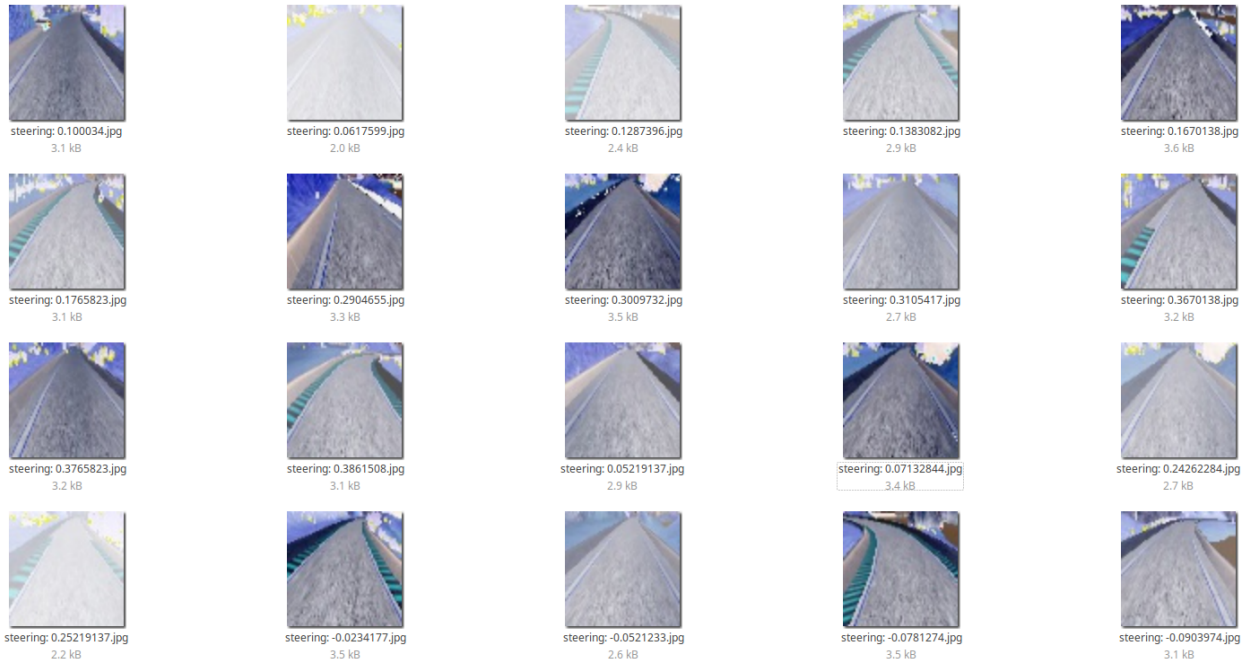

Figure 3: Input images produced by Keras generator after preprocessing.

3 Architecture

Illustrated by a research team of Nvidia, the most promising architecture type for this computer vision task are Convolutional Neural Networks (CNN): “The CNN approach is especially powerful in image recognition tasks because the convolution operation captures the 2D nature of images. Also, by using the convolution kernels to scan an entire image, relatively few parameters need to be learned compared to the total number of operations.” [1]
First progress was made by setting up a less complex model on basis of the LeNet-5 architecture [2] comprising two convolutions followed by 3 fully connected layers. With the goal of getting a more generalized output, which achieves to react more appropriately to sharper curves and varying image input i.e. change of road surface texture, a model was implemented inspired by the Nvidia CNN Architecture shown in Figure 4.
After normalizing the input, the first three convolutional layers apply a 5x5 kernel with a stride of 2 in order to half-size each output roughly by two. The following two convolutions with a 3x3 kernel increase the depth up to 64 feature maps. The output is flattened and passed to three fully-connected layer with respective outputs of 100, 50 and 10, closing with an outcome of a single steering value prediction. Adding a dropout function between the first convolutions proved to regularize the model, helping it to produce more generalized output. The common standard of using Rectified Linear Units (ReLU) after every convolutional layers to avoid saturation was replaced by a new type of neuron called Exponential Linear Unit (ELU) which is claimed to induce slightly faster learning [3]. Testing further adjustments and even different architecture prototypes did not show any significant improvements on this case.

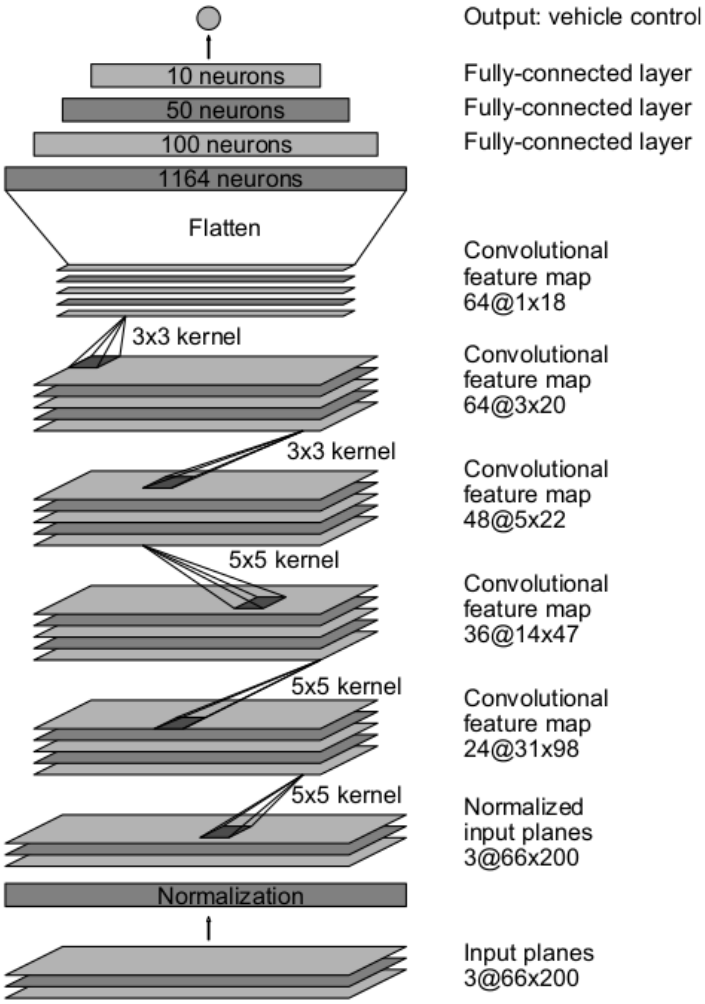

Figure 4: Nvidia CNN Architecture with approx. 27 million connections and 250 thousand parameters. [1]

4 Training and Tuning

The model was trained using Adam optimizer, which is known to be a very efficient method for stochastic optimization requiring only little tuning of hyper parameters. Not being a classification but a regression task mean squared error (MSE) was the loss function of choice. For the same reason the validation loss is the only metric of interest for analyzing training output. Around 20 % of the generated data was split from the original data set and reserved for validation.
Even though certain tuning variations revealed loss values of only 0.0149, the final, successful configuration resulted in a validation loss of 0.0228, proving that validation loss is an indication during the progression of epochs, but not entirely trustworthy for optimizing and tuning a model designed for solving regression tasks.

The final setup was trained for 8 epochs with a total input of roughly 24,000 images each epoch, including around 4,800 for validation. Augmenting the data only works to a certain extend and requires further manipulation to avoid loosing diversity and therefor causing unwanted outcomes. Neither did further lowering of the learning rate (down to 0.0001) bring any improvement, even by increasing the number of epochs.

5 Results

As a result steering values predicted by the trained model were able to safely conduct the simulated car around a virtual test track. On basis of input from the road in front of the car, the output of the model was streamed back into the simulation in real-time, proving the high performance of Deep Neural Networks combined with the processing power modern GPU. “Conclusion: The CNN is able to learn meaningful road features from a very sparse training signal (steering alone).” [1] This project also clearly wants to emphasize the significance of data processing and augmentation to produce appropriate training input and solving this task. Both techniques proved to have even more crucial effects on the output, in terms of generalization, than tuning of hyper parameters and experimenting with additional network layers.
Till now the above demonstrated methods are already being applied to various regressive computer vision tasks and very different fields. They are subject to further investigations worldwide to enhance the impact of machine learning in the ongoing development of today’s self-driving car.

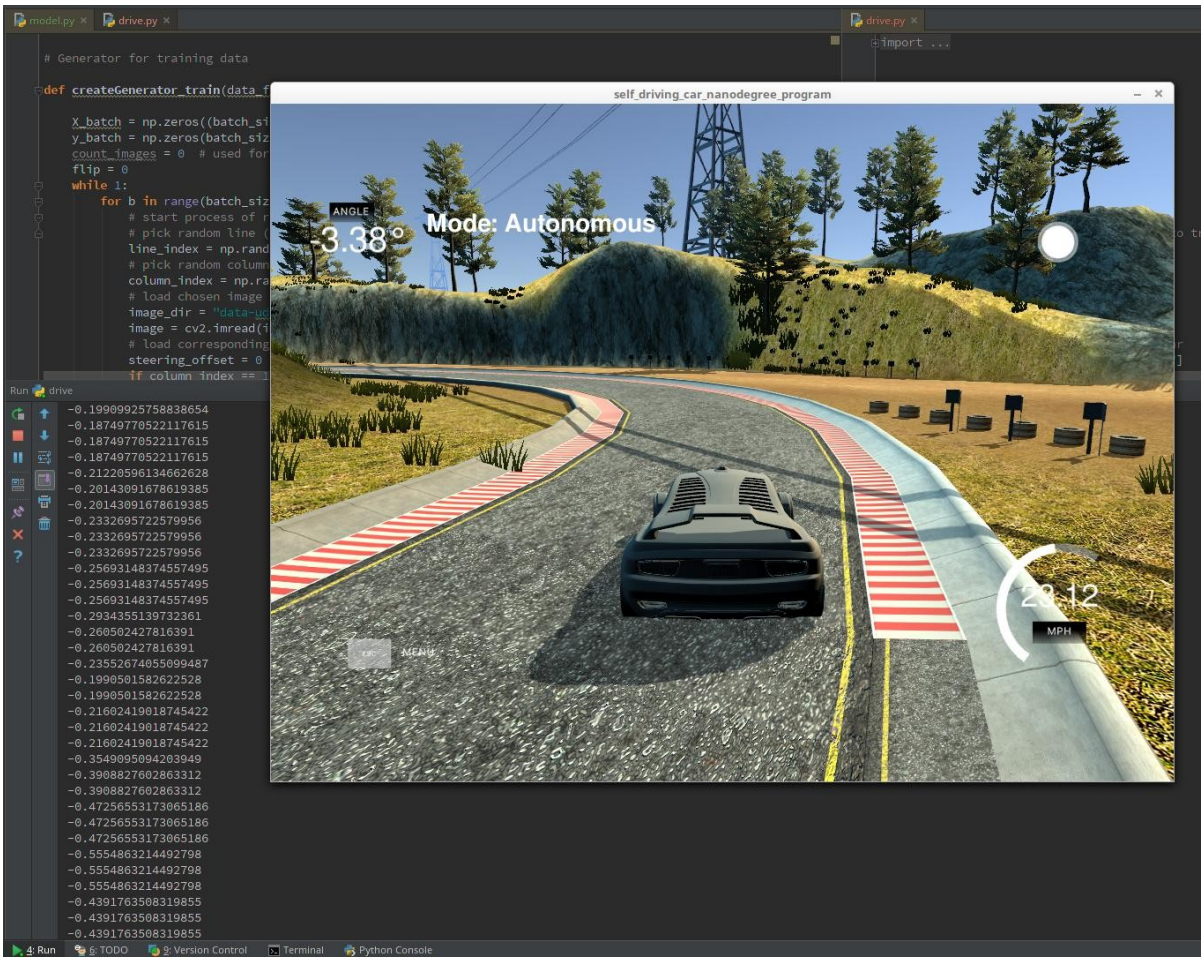

Figure 5: Car navigates correctly with steering input produced by trained model in real-time.

References

[1] M. Bojarski, D. Del 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 (Nvidia Corporation). *End to End Learning for Self-Driving Cars*. 1604.07316v1, April 2016.
URL: https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/

[2] Y. LeCun, L. Bottou, Y. Begio and P. Haffner. *Gradient-Based Learning Applied to Document Recognition*. 10.1109/5.726791, November 1998.
URL: http://ieeexplore.ieee.org/document/726791/

[3] D. Clevert, T. Unterthiner and S. Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 1511.07289v5, February 2016.
URL: https://arxiv.org/pdf/1511.07289.pdf