Self Driving Car Nano Degree

Behavioral Cloning Project

Bastian Dittmar

June 2, 2017

1 Introduction

In this project, an end-to-end network is designed using the keras framework and trained to output a steering angle based on an imput image. The input is what a car sees in frot of it using a virtual camera while driving down a given track. The steering angle output will be transmitted to the car and effects its direction directly. (Bojarski et al. 2016; Krizhevsky, Sutskever, and Hinton 2012; LeCun et al. 1998; Yadav 2016)

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road
- Summarize the results with a written report

2 Files Submitted & Code Quality

2.1 Submission includes all required files and can be used to run the simulator in autonomous mode

My project includes the following files:

- model.py containing the script to create and train the model
- drive.py (unmodified) for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- test_drive.mp4 showing the car driving around the track successfully
- writeup_report.pdf summarizing the results

2.2 Submission includes functional code

Using the Udacity provided simulator and my drive.py file, the car can be driven autonomously around the track by executing

```
$ sh python drive.py model.h5
```

2.3 Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works. The program executes different actions based on command line parameters. The basic modes are training a network, evaluating a set of images, plotting the accumulated training history and testing the data augmentation.

3 Model Architecture and Training Strategy

3.1 An appropriate model architecture has been employed

My model is the based on the one used by Bojarski et al. (2016) from Nvidia, except for the input dimensions and a preprocessing layer. The layers are depicted in figure 1 (implementation in model.py lines 253-305).

The first two layers are for preprocessing. At first a lambda layer scales and shifts the data from an interval of [0, 255] to [-1, +1]. The second layer crops the image remove the bottom 25 and the top 70 pixels. That basically cuts hoff the hood and the sky down to the horizon. At first I applied those two layers the other way around as it should be more efficient to crop first but the network converged worse on my first initial tests. That is wh I went back to this order and never tried again. In principal it should not make a difference regarding the accuracy.

The 5 layers are convolutional layers. Three of them with a kernel size of $5x5$ and a stride of $2x2$ with a growing depth. The last two of the convolutional layers have a kernel size of $3x3$ and a stride of $1x1$. Each convolutional layer is followed by a relu activation to introduce non-linearity.

Next the network is flattened and the data passed through four fully connected layers reducing the output to 100, 50, 10 and 1 where the last one is the output of the network and thus the steering angle to be learned.

3.2 Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 21).

The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 10-16). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

3.3 Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually (model.py line 25).

3.4 Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left and right sides of the road ...

For details about how I created the training data, see the next section.

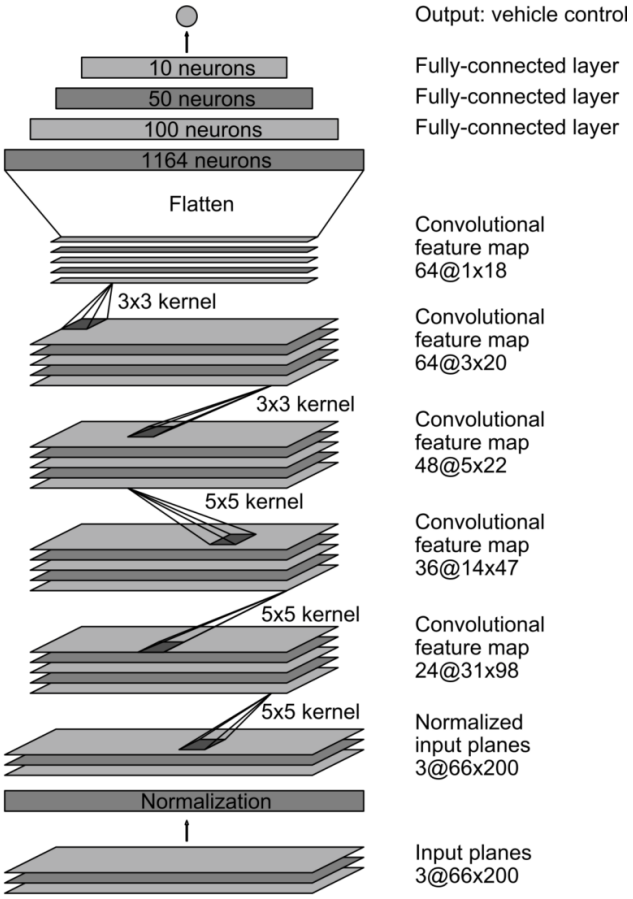

Figure 1: The network architecture used by (Bojarski et al. 2016)

4 Model Architecture and Training Strategy

4.1 Solution Design Approach

The overall strategy for deriving a model architecture was to ...

My first step was to use a convolution neural network model similar to the ... I thought this model might be appropriate because ...

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. I found that my first model had a low mean squared error on the training set but a high mean squared error on the validation set. This implied that the model was overfitting.

To combat the overfitting, I modified the model so that ...

Then I ...

The final step was to run the simulator to see how well the car was driving around track one. There were a few spots where the vehicle fell off the track... to improve the driving behavior in these cases, I ....

At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

4.2 Final Model Architecture

The final model architecture (model.py lines 18-24) consisted of a convolution neural network with the following layers and layer sizes ...

Here is a visualization of the architecture (note: visualizing the architecture is optional according to the project rubric)

4.3 Creation of the Training Set & Training Process

To capture good driving behavior, I first recorded two laps on track one using center lane driving. Here is an example image of center lane driving:

![alt text][image2]

I then recorded the vehicle recovering from the left side and right sides of the road back to center so that the vehicle would learn to .... These images show what a recovery looks like starting from ... :

![alt text][image3] ![alt text][image4] ![alt text][image5]

Then I repeated this process on track two in order to get more data points.

To augment the data sat, I also flipped images and angles thinking that this would ... For example, here is an image that has then been flipped:

![alt text][image6] ![alt text][image7]

Etc ....

After the collection process, I had X number of data points. I then preprocessed this data by ...

I finally randomly shuffled the data set and put Y% of the data into a validation set.

I used this training data for training the model. The validation set helped determine if the model was over or under fitting. The ideal number of epochs was Z as evidenced by ... I used an adam optimizer so that manually training the learning rate wasn't necessary.

References

Bojarski, M., D. D. Testa, D. Dworakowski, et al. (2016). "End to End Learning for Self-Driving Cars". In: *CoRR* abs/1604.07316.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, et al. Curran Associates, Inc., pp. 1097–1105.

LeCun, Y., L. Bottou, Y. Bengio, et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Yadav, V. (2016). *An augmentation based deep neural network approach to learn human driving behavior*. https://chatbotslife.com/using-augmentation-to-mimic-human-driving-496b569760a9. [Online; accessed 27-May-2017].