Behavioral Cloning Project

Goals

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

Rubric Points can be found here: https://review.udacity.com/#!/rubrics/432/view

The following sections are categorised based on the rubric structure.

Files Submitted & Code Quality

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

My project includes the following files:
1. model.py containing the script to create and train the model
2. drive.py for driving the car in autonomous mode
3. model.h5 containing a trained convolution neural network
4. writeup_report.pdf summarising the results
5. video.mp4

B. 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

```
python drive.py model.h5
```

C. 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.

Model Architecture and Training Strategy

Model Architecture

I have taken over the model as suggested in the introduction video. In detail my model is based on the following architecture:

1. Layer: Convolutional (24 filters, size: 5x5 stride 2)
2. Layer: Convolutional (38 filters, size: 5x5 stride 2)
3. Layer: Convolutional (48 filters, size: 5x5 stride 2)
4. Layer: Convolutional (64 filters, size: 3x3 stride 2)
5. Layer: Convolutional (64 filters, size: 3x3 stride 2)
6. Layer: Fully Connected (100 neurons)
7. Layer: Fully Connected (50 neurons)
8. Layer: Fully Connected (10 neurons)

As mentioned by the reviewer of my last submission: the fully connected layers tend to overfit. Therefore, I have added a dropout layer in between the fully connected layers

My selected activation function was a classical ReLu. I have experimented with more advanced activation functions as the ELU but their usage led to limited results.

I have also experimented with different architectures as the LeNet and variations of AlexNet. Unfortunately, these led to very poor results. I assume that the reason for the bad performance of architectures that are focused on image recognition are the used pooling layers that lead to a generalisation of the entire image and since the road boundaries as well as the surface are important for the resulting correct movement of the vehicle, these pooling layers are unnecessary. In fact, while trying different architectures I have noticed that many convolutional layers with simple fully connected layers are much more effective. Keeping this in mind, the introduced NVIDIA architecture seemed a proper fit for the task.

Since I have used the ADAM optimiser, there was simply no need to focus on particular parameters. My basic assumption was: ADAM with default settings and 10 epochs in combination with the architecture provided would do just fine as the major focus for enabling an autonomous behaviour relies in the preprocessing of the data. More details in the following.

Training Strategy

Data Preprocessing

As suggested, I have normalised the input data. This let to a tremendous increase in performance. In addition to that, I have also cropped the inputs which implied that only the useful lower parts of each input has been considered during training. Last but not least, I have used the insights from the first project on lanes to preprocess the data. My idea was based on the following question: what features in the images have most probably the highest impact on the resulting steering angle? - The road boundaries and the surface, obviously. Keeping this in mind, I have applied a gaussian blur filter over each image with kernel size of 3; which led to the second major improvement in terms of behaviour after normalisation.

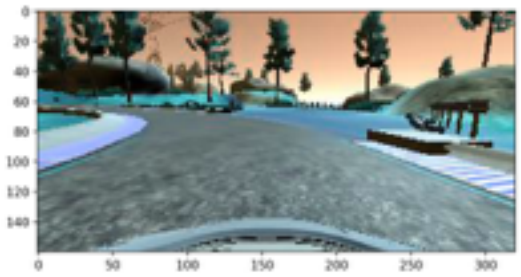

Figure 1. Original image

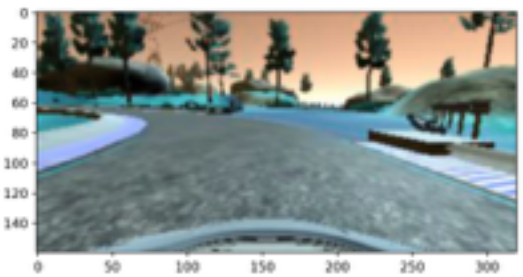

Figure 2. Applied gaussian blur

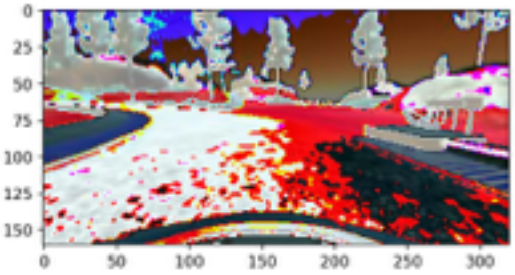

Figure 3. Normalised image

Results: The vehicle was able to drive (very shaky) until the bridge within the road boundaries solely based on these preprocessing approaches. At the bridge however, my model seemed to neglect these boundaries.

Based on the idea that the NN will be focused on edges in each image I have experimented with the following (unsuccessful) approaches:

1. Prepocessing all images with the Canny algorithm
2. Projecting lanes (using Hough) onto each input image

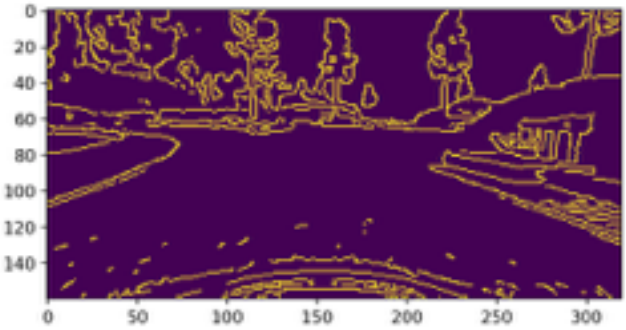

Figure 5. Applied Canny

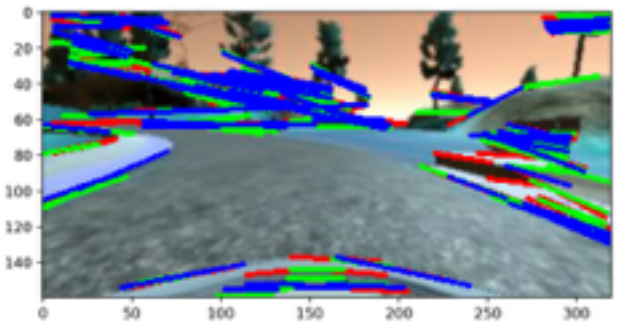

Figure 6. Applied Hough Lines

My assumption for the failure of these approaches is the lack of features available in an image after the application of Canny and the overreaction to lines in curves after the application of the Hough algorithm.
I have dropped these approaches and focused more on training data interpretation (refer below).

Training Set Characteristics

Taking the intro on the project into account it seemed that the major fact for implementing a reliable behaviour relied in the lack of available data. Therefore, flipping images as well as considering the left and right hand images are suggested as a solution.
My assumption, however, was that the center-images provided in the example training set, were enough to complete the task. In fact, I was actually quite certain that the provided data was a bit too much. Therefore, I have used the Udacity data for the training of my model.

Considering the distribution of the target values (refer below), one can clearly identify that the training data set includes many targets of value 0; in fact, almost three times as much compared to the remaining values. When using these values, its almost inevitable that the vehicle would not be able to cover left and right curves.
This let me to the idea that I should drop a few training inputs rather than creating new ones. Therefore, I have read in every second feature/target pair that had a steering angle of 0.

Figure 7. Training set target value distribution

Using this approach, the resulting model was able to drive smoothly straight as well as around the first corner. It was overreacting on the bridge and was not able to cover curvy roads, however.

Since the results were promising in anyhow, I was sure that I was on the right track and that my approaches are fine as I have reduced the issues to one single problem: covering curves.

In this case I was sure that a tiny bit of training data that covers a curve would result in the desired behaviour of the model (i.e. fine tuning of the training set).

In order to solve this remaining issue, I have recorded an additional driving manoeuvre that solely covered the first part of the track (first curve -> bridge -> second curve). The data was used to fine tune the weights after the actual training was finished.

Figure 8. Target value distribution of additional training data

Figure 9 - 11. Example images of additional training data covering the first curve, the bridge and the second curve.

This resulted in a smooth driving of the vehicle almost on every part of the lap. No boundaries are touched and steering angle corrections imply minimal jerk.

Discussion

I feel that my approach is simple and yet very functional and performant due to the reduced data set. There is some room for improvement of course. Keeping in mind that I was only focused on the major track, the next step of “evolution” is a generalisation that enables to use the second track.

Regarding the data preprocessing: one idea that I have not tried is the usage of the three dimensions in the training data. Since the colour information might not be too relevant for the training of the model, I would replace the 3 dimensions with three identical, grayscale images representing the same scene. In combination with dropout layers that avoid overfitting this could be an interesting approach.