Project 4: Behavioral Cloning

Student: Guido Rezende de Alencastro Graça

Goals of the Project:

This project uses a driving simulator developed by Udacity on the Unity software. The focus of the project is to develop an algorithm that calculates the steering angle at any time during an autonomous driving simulation, in order to create an autonomous driving vehicle on the simulator. This algorithm uses Convolutional Neural Networks (CNN) on images recorded by the center camera of the vehicle. The images used for training the network were recorded during manual driving on the simulator.

Rubric points:

1. Files and Code Quality
2. Model Architecture and Training
3. Simulation

1. Files and Code Quality

The files are on the CarND-Behavioral-Cloning-P3 file. The code file is the **model.py**, the trained model on **model.h5** and the video on **video.mp4**. The **drive.py** file is also on the file. The code is well commented, so it is easy to understand.

2. Model Architecture and Training

2.1 Training data

The strategy for collecting data for the training consisted of two laps on the circuit, both at around 15 mph, as the autonomous simulation speed was 9 mph. The first lap was focused on smoothly driving on the center of the lane, whereas the second lap was more tortuous, performing some zigzag throughout the course. This was made in order that the first lap creates data on how the vehicle shall drive normally, whereas the second lap focus on creating data on what should the CNN do in case the vehicle is not aligned with the lane, and so be able to return to the center. The simulator captures three images at each frame, one for the center camera and one for each side of the vehicle, as seem in Figure 1. These images are 320px x160px each.

Figure 1: The three cameras of the vehicle

Previous attempts showed issues on the bridge, due to its different road and lane limits (Figure 2a), and on curves without the outer lane line (Figure 2b), so more data was recoded focused on these areas.

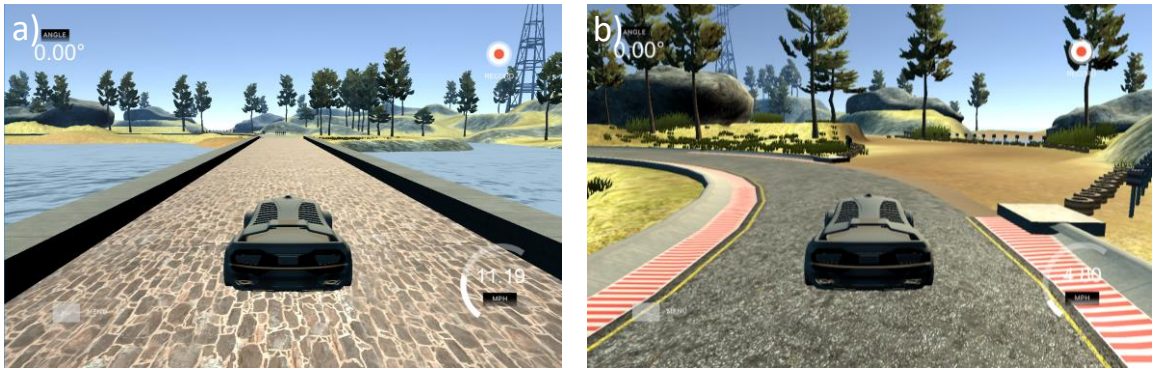

Figure 2: Different road conditions. a) Bridge. b) Without outer lane

The preprocessing consisted of cropping the images on the y-axis using 70 pixels from the top and 25 from the bottom. There was no cropping on the x-axis, as the lane extends to the image limits (Figure 3b). The data was normalized by adjusting the values of the pixels to the range [-0.5, 0.5]. The data was augmented by flipping the images on the vertical axis, resulting on a mirroring effect (Figure 3c). In total, it was used 21996 images, divided using an 80:20 ratio, resulting in 17596 training samples and 4400 validation samples.

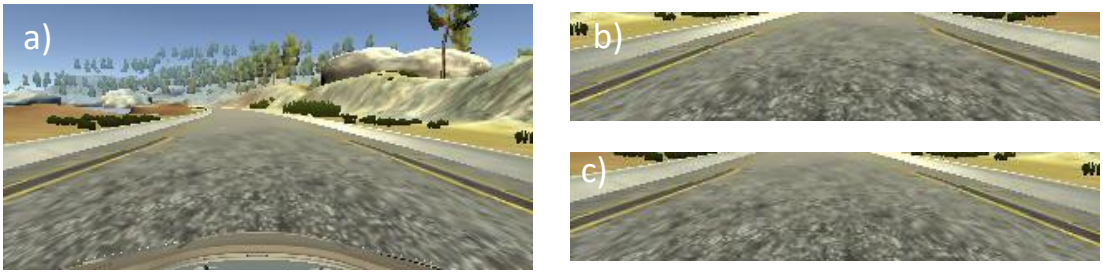

Figure 3: Editing and augmenting the data. a) Original image. b) Cropped. c) Flipped image.

2.2 Model Architecture

The CNN architecture used was proposed by *NVIDIA* at *End to End Learning for Self-Driving Cars*[1]. This model was chosen as it was intentionally developed for self-driving vehicles. This network consists of five convolutional layers, followed by flattening and four fully connected layers. The network structure is shown in Figure 4, with the number of filters and image dimensions shown on the right. This implies on around 250 thousand parameters of the network. The model was trained using one Epoch using a default batch size of 32. Resources for avoiding overfitting the model weren't used, as the loss continually decreased during training and the validation loss was also relatively small (training loss: 0.0421 and validation loss: 0.0497). However, the lack of these approaches implied on using only one Epoch, as increasing the number of Epochs leaded to overfitting and therefore loss increased during the second Epoch. As for the optimization, the optimization algorithm used as *adam*, thus not needing for manually setting the Learning Rate, and the loss calculation function was the mean squared error (*mse*).

[1] Link: https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

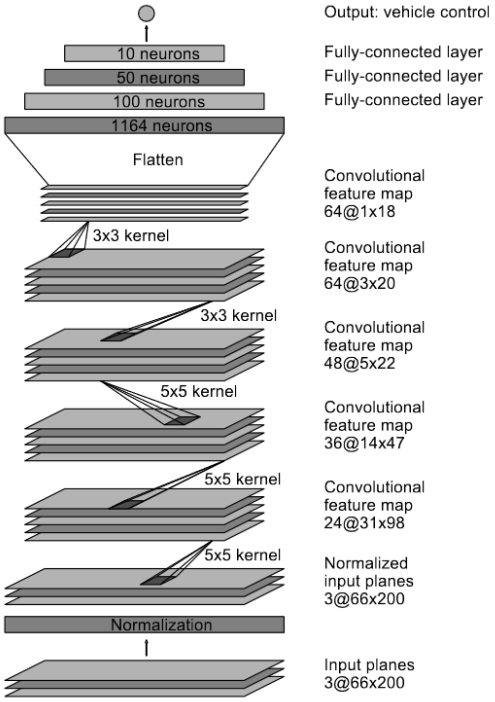

Figure 4: CNN proposed by NVIDIA. The description of each layer is on the right. . Source: https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

3. Simulation and results

The autonomous simulation was successful. The vehicle managed to drive on the circuit without crashing or drive over the lane lines. The video.mp4 file shows a lap on the circuit. The simulation went on over twenty minutes without performing any of these dangerous maneuvers. The car drives relatively smoothly, with some small zigzag, although in a real life case it would be better a more steady driving. The first circuit is a relatively simple one, without many shadows, ramps or other artifacts that hinder the drivability. On more challenging circuits as the second, for instance, more data should be necessary, and so, the need to prevent overfitting, using techniques such as Dropout, early termination or others.