Behavioral Cloning Project

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.

---

1. Files Submitted & Code Quality

1.1 Content of the submission

This project submission consists of the following files:

- **model.py** - Python script used to create and train the model.
- **drive.py** - script used for driving the car in autonomous mode.
- **model.h5** - containing a trained convolution neural network.
- **writeup_report.pdf** - This is the present document and it summarizes the results the key points of the project.

1.2 Submission includes functional code.

The python scripts included in this submission are functional and can be used to construct, train and use the model for autonomous driving. The model "model.h5" is already trained to successfully drive the car-simulator autonomously. To do so, please execute the following command:

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

1.3 Readability of the code.

The file "model.py" contains the code for creating, training and saving the convolution neural network used in this project. The comments in the file highlight the pipeline I used

for training and validating the model.

2. Model Architecture and Training Strategy

2.1 Model architecture

The model architecture is depicted in Fig.1.

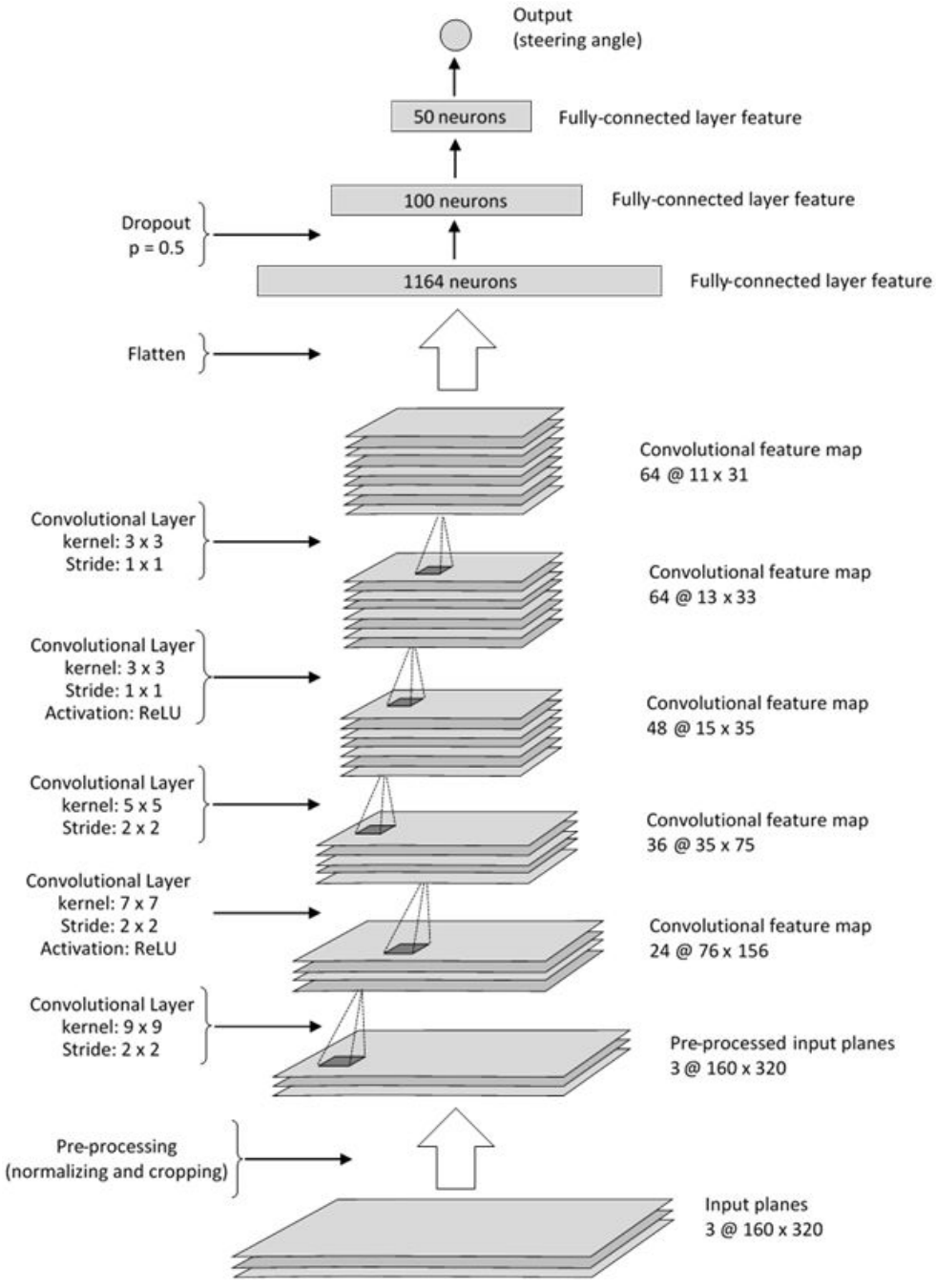

Figure 1 - Model architecture used in the "Behavioural Cloning Project".

The model consists of a convolution neural network with five filters with decreasing kernel size. The output of two of these filters is activated through ReLU to introduce nonlinearity. A dropout layer is also introduced between the first two fully connected layers to reduce overfitting.

The model also includes two layers for pre-processing the input images. In particular, the raw images are first cropped (to allow the model to focus on the road rather than the whole surrounding) and then normalized.

2.2 Attempts to reduce overfitting in the model

As mentioned before the model contains a dropout layer in an attempt to reduce overfitting. Also, to assess the performance of the model, training and validation were done on different datasets. In particular the dataset obtained by manually driving the car in the simulator was split into two parts, one (containing 80% of the samples) used for training the model and one (containing the 20% of the samples) for validating its performance.

The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

2.3 Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually.

2.4 Appropriate training data

The training data was chosen to make the model learn how to keep the vehicle on the road during autonomous driving. I used a combination of center lane driving, and recovering from the left and right sides of the road. This was meant to show to the model good driving practice (driving in the middle of the road) and how to recover from too sided or unsafe positions on the road. More details are provided in the following sections.

3. Model Architecture and Training Strategy

3.1. Solution Design Approach

My model is based on the convolutional neural network published by the "autonomous vehicles team" at Nvidia, and presented at the Udacity Self-Driving Cars Nanodegree, Term1, Course "Behavioural Cloning", section 14.

Nvidia's original model was modified to include elements that were not visibly represented in the schematic drawing provided in the course.

In particular, ReLU activation layers were added to increase the capability of the network to capture non-linearities, and a dropout layer was introduced to decrease overfitting.

The size of the kernel in the convolutional layers was also modified. In particular the size was set to a bigger value on the first filter and decreased progressively at every subsequent filter (with the exception of the last one) to end up with the same size on the last filter as in the original model.

The reason for increasing the kernel size on the first filter was to allow basic shapes to be

more easily detected given the size of the input image.

3.2 Creation of the Training Set & Training Process

I found that the model improved considerably with the number of inputs that were fed into it during the learning process. The number of inputs needed to reach acceptable results was not known at the beginning, so different manual driving sessions were recorded at different times and fed into the model.

For the model to improve upon previous parameter fittings, I used a transfer learning approach, where the fitted parameters of the previous learning session where used as starting point for the new learning session.

It should be mentioned that the images captured by all the three cameras (centre, left and right) were used to train the model. The steering angle was adjusted by 0.2° for the left camera images and by -0.2° for the right camera image. The rationale of this approach is that an image captured by the left camera resembles more to a situation in which the car is too much on the left side of the road (this is true in average, if the car is driven in the centre of the lane most of the time). To help the model recenter the car, we add 0.2 to the corresponding steering angle. A similar reasoning applies for the images captured by the right camera.

Fig. 2 shows the images captured by the center, left and right camera for center lane driving.

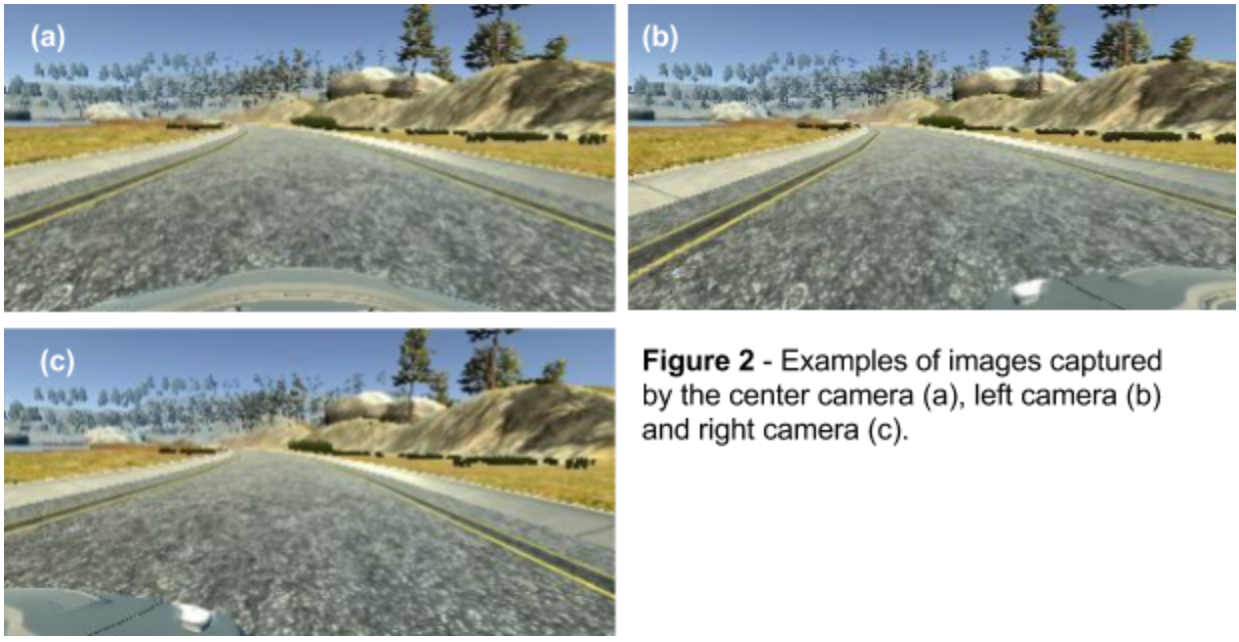

Figure 2 - Examples of images captured by the center camera (a), left camera (b) and right camera (c).

Of particular importance was also to capture recovery situations, in which the car was "pushed back" to the centre of the lane starting from the left or the right side. This allowed the model to learn how to recover from unsafe or suboptimal positions. Fig.3 shows how the recovery from the right side of the road look like.

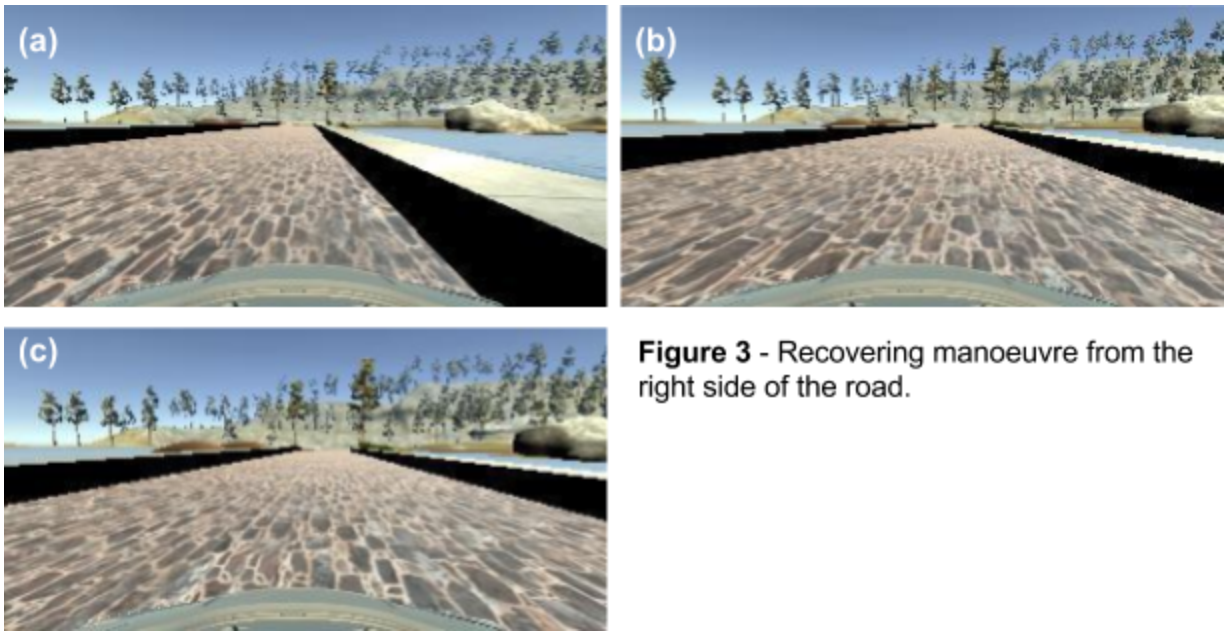

Figure 3 - Recovering manoeuvre from the right side of the road.

Finally, at each learning process I made sure that the number of epochs was not too high, leading to potential overfitting. I found that 3 to 5 epochs were enough for this purpose.