Car-ND-Behavioral-Cloning-P3

1. Submission

My project includes the following files:

- *model.py* containing the script to create the model
- *train.py* containing generators and training
- *drive.py* for driving the car in autonomous mode
- *model.h5* containing a trained convolution neural network
- *writeup.pdf* summarizing the results
- *responses.pdf* summarizing the required changes
- *run.mp4* video containing (approximately) one lap of autonomous driving

NOTE: For the sake of clarity, I have split the training process into two separate files: one containing the model architecture (*model.py*) and one containing the training process (*train.py*).

Using the simulator provided by Udacity, the car can be driven autonomously around the track by executing `python drive.py model.h5`.

Since currently I am having problems with my PC (fried power supply :-( ), for this project I had to work on docker images. The *keras* and *tensorflow* modules, contained in the provided docker, are somehow outdated (v.1.2.1) so it is possible (almost guaranteed) that running *drive.py* would not work when launched using (local) up to date modules (v.2.x.x). Therefore, I recommend to launch *drive.py* using the Car-ND docker image. For the same reason, the whole training process has been carried out on the docker image as well since loading network models that have been trained using a different version of *keras* usually doesn't work (at least not straightforwardly).

The car, driving on the autonomous mode, can successfully complete a lap (any number of laps, actually) on track 1 without getting off the road surface.

2. Model architecture

In one of the classroom videos it was suggested that, when facing a new problem, it is always better to look for an already existing network architecture and adapting it to the problem being tackled rather than trying to build a new model from scratch. Following this suggestion, we use the architecture proposed by Bojarski et al. "End to End Learning for Self-Driving Cars" (Nvidia, 2016) which is summarised in Fig. 1. A small modification to the original design was introduced by adding a 20% dropout after the first fully connected layer in order to combat a possible overfitting.

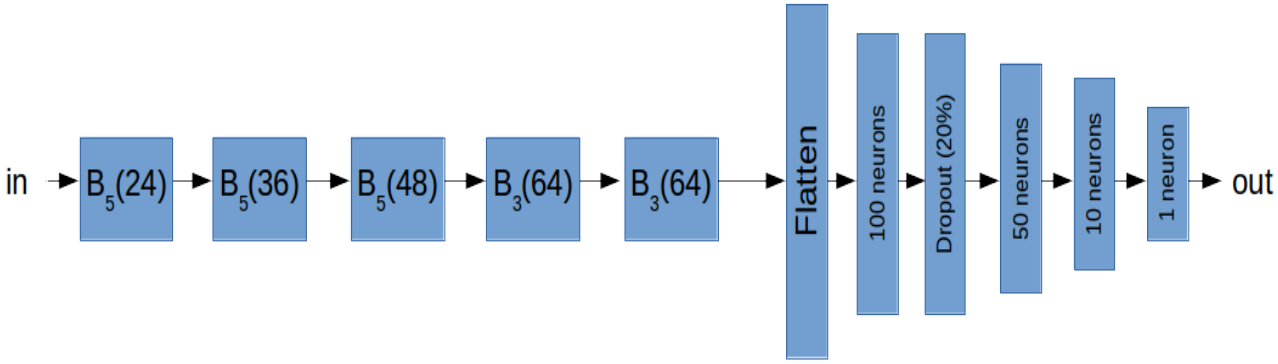


Figure 1: Network architecture used in this project.

In this scheme, $B_5(n)$ is a 2D convolution block including n 5x5 filters with strides equal to 2 and followed by a ReLU and $B_3(n)$ is a 2D convolution block including n 3x3 filters with strides equal to 1 and followed by a ReLU.

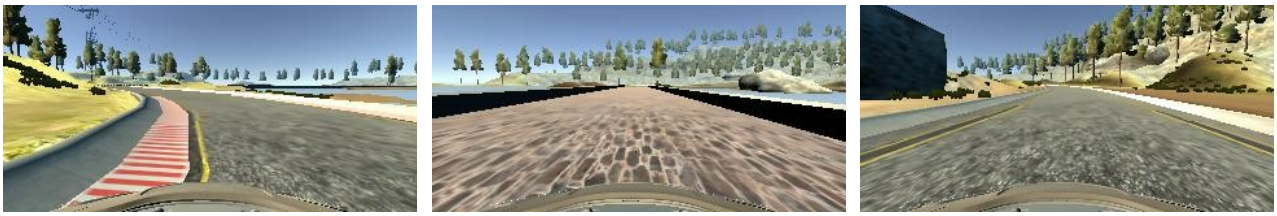
Furthermore, the input images are cropped so their upper part (which mostly contains data that is irrelevant for learning) is cut out (see *model.py*, line 16). The model uses Adam optimizer so the learning rate was not tuned manually.

3. Training data

The model was trained using both the example samples provided as a resource for this project and the samples obtained by running the simulator by ourselves. In order to gather our own data we recorded two additional laps on track 1:

- One additional lap with very slight zigzagging.
- One normally driven lap in the opposite direction.

Overall, we obtain 12197 images (8036 from the provided dataset and 4161 collected by ourselves). These images are taken from the central camera and side views are not used during the training. We use 80% of data for training and the remaining 20% for validation. The model is finally tested using the autonomous mode of the Udacity simulator. Three examples of different training images are shown in Fig. 3.



(a) (b) (c)

Figure 3: Examples of training images not included in the project resources. (a) snapshot of zig-zag driving, (b) lap in the opposite direction, (c) normally driven lap.

3.1. Data augmentation

Since the model will be run on a simulated environment, it does not make much sense to play with noise, gamma correction, contrasts, etc. Instead, the only data augmentation we incorporated during the training is horizontal flipping with the corresponding sign change in the steering angle. This augmentation is being done online using a data

generator (see *train.py*). The learning process is based on batches of 32 images and each image within a batch is flipped with a probability of 0.5.

4. Training

We have trained the model for 25 epochs and the training and validation losses are shown in Fig. 2.

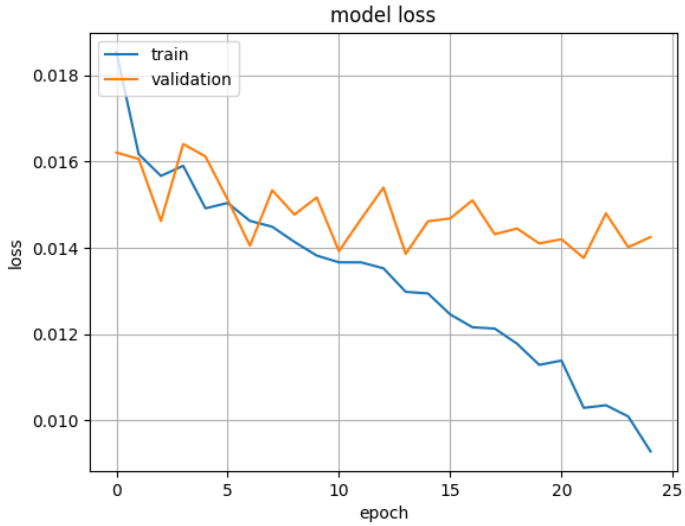


Figure 2: Training and validation loss for each epoch.

It can be observed that the validation loss (its overall behaviour) is slightly decreasing throughout the whole training so the overfitting doesn't seem to be that severe an issue. As the final model, we have chosen the one that yields the lowest loss, i.e. the one obtained at epoch 21 (starting from 0). Nevertheless, the loss differences among different epochs are (relatively) minimal and any other model would (and it actually does) drive the car successfully around the lap without getting off the road.

However, we are dealing with an implicit overfitting due to our training data. We actually overfit the network to the lap 1 and, in addition, to the situation when everything is going relatively well (no off track driving).

5. Discussion

Here we would like to provide some observations about the project and the achieved solution.

- Given the simulation scenario, normalising the data by the full dynamic range and centering it by half of the dybamic range is a reasonable approach. However, for real world scenarios the normalisation should be dynamic (i.e. taking into account the actual dynamic range and mean value).
- The training data for track recovery is very deficient. The car is able to stay on track but it won't be able to recover while off track. The more (and various) data is used for training the better.
- It has not be trained for the second track (unfortunately there have been several issues, personal and work related, that didn't allow me to dedicate as much time to this project as I'd like to). The second track is more challenging not because of (apparently) lower colour gamut of the scene but because the road is changing also vertically, i.e., it contains slopes. This is particularly challenging when going

uphill since the visible part of the road is severely reduced when close to the summit.