Self-Driving Car Engineer Nanodegree Program

Behavioral Cloning Project

François Masson
November 17nd, 2018

Pipeline Software

The project consists of using a deep learning model architecture to copy the way a human drive through a simulation software. In order to fulfill this task, this guideline is applied :

- Collecting data using a simulator
- Building a convolution neural network in Keras to predict the steering angles from images
- Training and validating the model
- Test on the simulator

Collecting data using a simulator

The interface of the simulator is similar to the following images (Fig 1 and Fig 2):

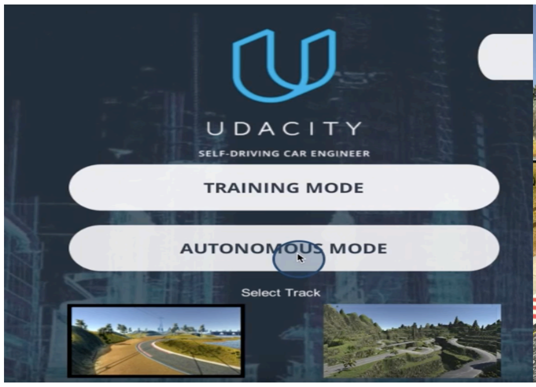


Figure 1 - Simulator interface

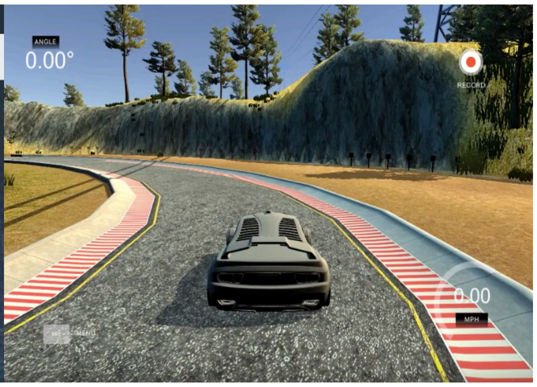


Figure 2 - Recording interface

When recorded, the simulator saves images and frames of the driving emulated by three cameras (left, center and right). In the same time, a csv file is produced in which each row contains the steering angle, throttle, brake and speed of the car. It will be explained later on this project but in order to fulfill this challenge:

- Two tracks are done in clockwise
- One track is done in counter-clockwise
- Images are cropping to remove unnecessary pattern in the model
- Images are flipping to augment the data
- Left and right images are incorporated in the dataset and a correction steering factor is applied.

The data processing step can be seen in model.py lines23:51. The data used in the challenge are placed in the /opt directory in the lake folder.

Building a convolution neural network in Keras to predict the steering angles from images.

The final convolutional neural network used is sketched below. It is directly inspired by the one provided by NVIDIA team:

https://devblogs.nvidia.com/deep-learning-self-driving-cars/

The network architecture consists of 9 layers, including a normalization layer, 5 convolutional layers, and 3 fully connected layers (Fig 3 and Fig 4). In this case, some dropout layers were added as well as maxpooling layers. This is done to reduce overfitting in the parameters. The model used an adam optimizer, so the learning rate was not tuned manually. The output label is in this case the value of the steering angle. So instead of using a softmax function evaluation, the mean squared error is evaluate the value of the steering angle.

This can be seen in model.py line75:101.

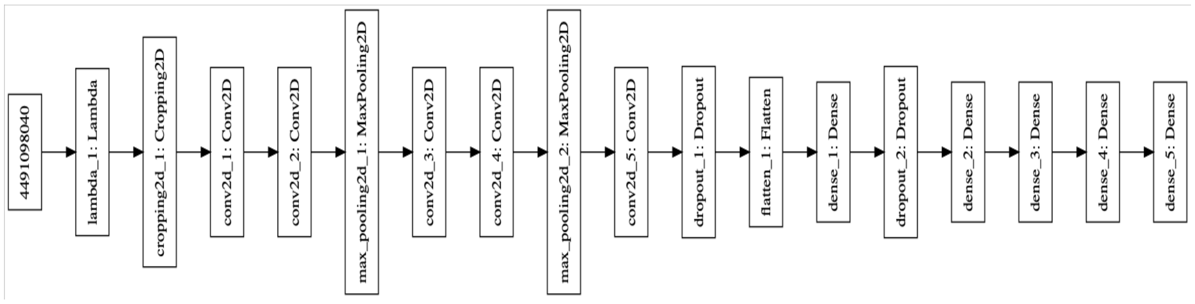
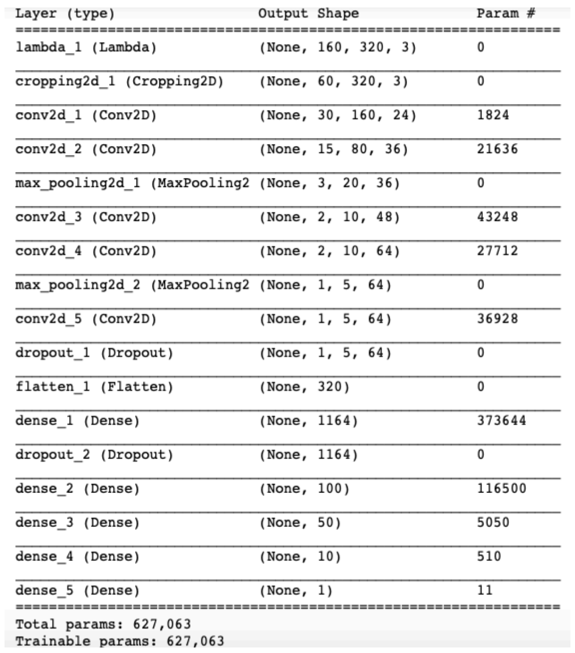


Figure 3 - CNN architecture

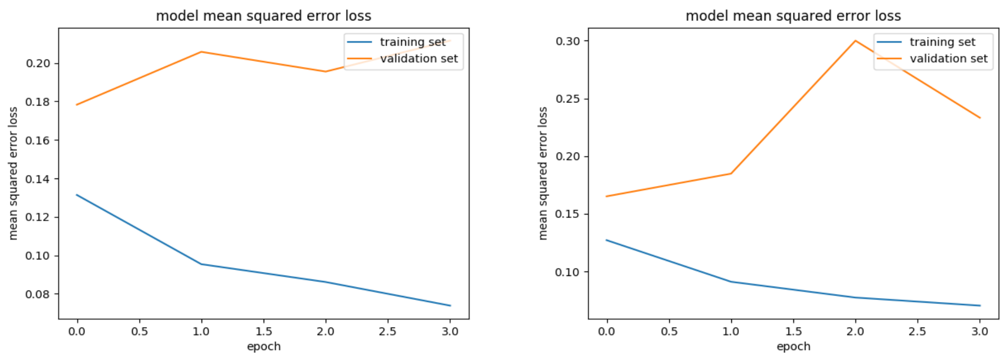


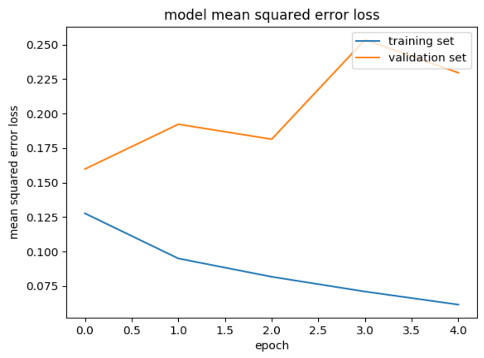
```
Layer (type)                 Output Shape              Param #
=================================================================
lambda_1 (Lambda)            (None, 160, 320, 3)       0
_________________________________________________________________
cropping2d_1 (Cropping2D)    (None, 60, 320, 3)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 30, 160, 24)       1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 15, 80, 36)        21636
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 20, 36)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 10, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 10, 64)         27712
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 5, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 5, 64)          36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 320)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1164)              373644
_________________________________________________________________
dropout_2 (Dropout)          (None, 1164)              0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               116500
_________________________________________________________________
dense_3 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_4 (Dense)              (None, 10)                510
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 11
=================================================================
Total params: 627,063
Trainable params: 627,063
```

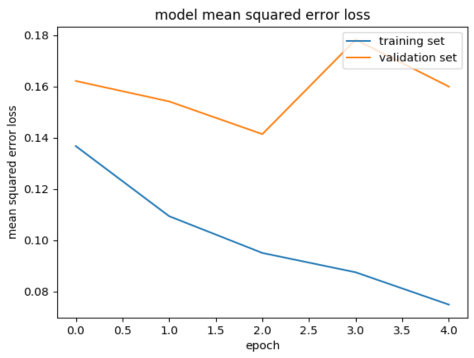
Figure 4 - CNN architecture

Despite the fact that some dropout layers and maxpooling layers were added, the model suffers from overfitting. Indeed, the loss on the training set decrease with the number of epochs while the loss on the validation set increase. This is the sign of overfitting (Figs 5). Later on, one possible solution is to add data to help generalize the model (See optional challenge).

Figure 5 - Losses evaluation





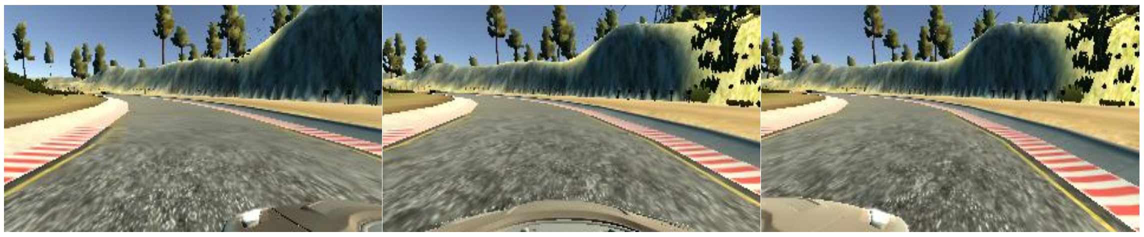


Training Documentation

As said previously data were recorded during a simulation consisting of:

- Two tracks are done in clockwise
- One track is done in counter-clockwise
- Left and right images are incorporated in the dataset and a correction steering factor is applied.

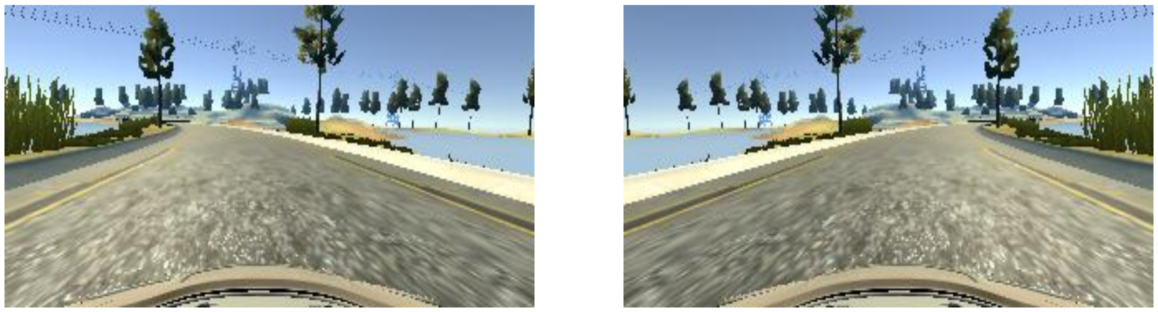
Figure 6 - Data Generation Left Center Right



For each frame, the left, the right and the center images were added to the dataset leading to data augmentation and it allows to provide a fine-tune steering parameter if the car coming to close to the lanes.

- Images are flipping to augment the data

Figure 7 - Original and flipped images



- Images are cropping to remove unnecessary pattern in the model

Each frame was also cropped to remove unnecessary pattern for the model.

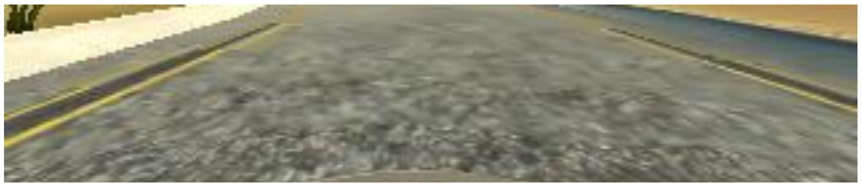
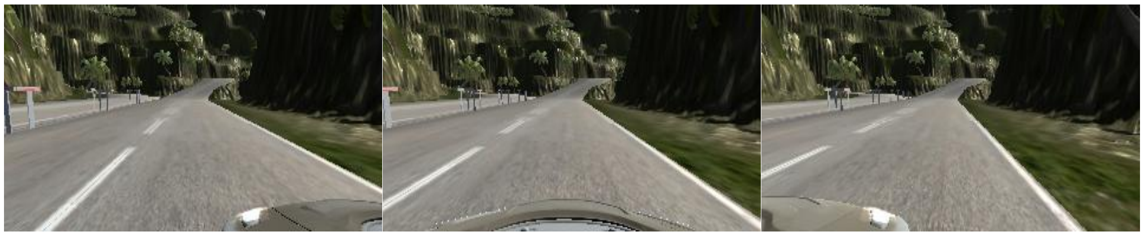


Figure 8 - Cropped Image

- Data from the other track (Forest) were also added.

Figure 9 - Data Generation Left Center Right Forest Dataset

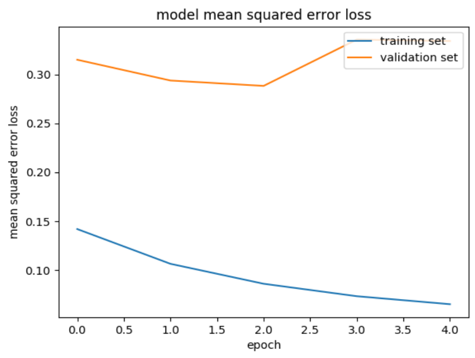


Output

- model.py - The script used to create and train the model.
- drive.py - The script to drive the car.
- model.h5 - The saved model.
- video.mp4 - A video recording of the vehicle driving autonomously.

Optional challenge

I add data from the forest dataset to the previous model.



It reduced the overfitting and I could possibly reduce the number of epochs to 3 to obtain the best solution.

Output

- model_opti.py - The script used to create and train the model.
- drive.py - The script to drive the car.
- Model_opti.h5 - The saved model.
- Challenge_Lake.mp4 - A video recording of the vehicle driving autonomously.