
Self-Driving Car Engineer Nanodegree Program
Behavioral Cloning Project
John Reilly
Due date June 2018

Table of Contents:

Please note adjustments for the resubmission are in blue

Behavioral Cloning

Writeup Template

Behavioral Cloning Project

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behaviour
- 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

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Files Submitted & Code Quality

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

My project includes the following files:

- ***writeup_report_augmented.pdf*** , The write up file
- ***model.IPYNB*** containing the script to create and train the model
- ***model_notes.IPYNB*** containing the above script but with progress comments and notes
- ***drive.py*** for driving the car in autonomous mode
- ***model.h5*** containing a trained convolution neural network
- ***model_new.h5*** new model with dropouts same input data
- Cloning_report.pdf summarizing the results
- ***final_video_track1.mp4*** is track 1 generated by drive.py
- ***new_final_video_track1.mp4*** is track 1 generated by drive.py Track 1 and track2 same video. Good Track one crashes track 2
- **FULL_SCREEN_new_final_video_track1.mp4** is a full screen version of above video

- ***FULL_SCREEN_final_video_track1.mp4.*** This is the same track 1 but a video that shows the whole of the desktop including the output of drive.py and the simulator at the same time. It is a lot easier to view exactly what is going on. Generated by third party software.
- ***final_video_track2_with_a_little_help.mp4*** is track2 with manual override on the hairpin bend to prevent crashes. Generated by drive.py.
- ***FULL_SCREEN_final_video_track2_with_a_little_help.mp4*** is track2 with manual override on the hairpin bend to prevent crashes. This is the same track 2 but a video that shows the whole of the desktop including the output of drive.py and the simulator at the same time. It is a lot easier to view exactly what is going on. Generated by third party software.

2. 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
```

Note out of 30 test models I chose the one the drove Track 1 perfectly and Track 2 reasonably. Other models did better on track 2 but worse on track one. Given track 1 was the requirement I chose the model that did best with track 1

Upon revision I attempted another 5 models choosing the final version model35.h5 as the best.

3. 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_notes.py contains many notes and comments that show what I did and what I was thinking when I did it. I included this to show my method and also provided a version "*model.py*" with the many comments and notes removed to provide a clearer and easier to read final version.

I am resubmitting new_Model_notes.py as this shows my notes.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

2 models were used and tested, LeNet and NVIDIA architectures. Both were successful to some extent, but NVIDIA did better on track 2 while maintaining performance on track 1.

In final submission the NVIDIA model was used,with the below architecture. Note it varies from the lecturers model by not doing cropping within the model, also note Dropout was sometimes used and not used.

Upon feedback I experimented further with dropout in quantity and location. Dropout rate of 0.5, 0.25 and 0.1 in single and two locations were tested. Rates of 0.5 and 0.25 twice were not good enough with the car crashing. Rate of 0.25 once or 0.1 once or twice were best. Subjectively rates of 0.1 in two locations was marginally the best. See location of added dropouts below.

```
model = Sequential()
model.add(Lambda(lambda x: x/255.0 -0.5,input_shape = (65,320,3)))
model.add(Convolution2D(24,5,5,subsample=(2,2),activation='relu'))
model.add(Convolution2D(36,5,5,subsample=(2,2),activation='relu'))
model.add(Convolution2D(48,5,5,subsample=(2,2),activation='relu'))
model.add(Convolution2D(64,3,3,activation='relu'))
model.add(Convolution2D(64,3,3,activation='relu'))
#model.add(Dropout(0.5))
model.add(Dropout(0.1)) #new
model.add(Flatten())
model.add(Dense(100))
model.add(Dense(50))
model.add(Dropout(0.1)) #new
model.add(Dense(10))
model.add(Dense(1))
```

2. Attempts to reduce over fitting in the model

The model contains a dropout layer with 50% probability in order to reduce over fitting. However this did not improve performance. This was noted on datasets with 1, 3 and 6 laps of Data and none of them improved. All models were retained and it was consistent that dropout made the car drive worse. Initially it was assumed one lap of data was not enough data to need drop out but with 6 lap using 3 camera and flip to double the data resulting in over 100,000 images and still dropout made the car drive worse.

Dropout was introduced to help counter over fitting see above.

3. Model parameter tuning

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

4. Appropriate training data

Initially for experimentation the default Udacity provided data was used. The car drove track 1 reasonably well with this data using Lenet. But it did touch yellow road side lines a couple of time so I changed to the NVIDIA model and began to add more data. Ultimately 3 laps of data for track 1 and 3 for track 2 with the track 2 data being left lane, centre and right lane. Why this was chosen is explained below.

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to attempt to use 2 well known architecture LeNEt and NVIDIA. It is known that LeNet was originally designed for handwriting recognition and was first designed in 1998 while on the other hand the NVIDIA model was recently designed specifically for self driving cars. LeNet is simpler so my approach was to implement LeNet and see how far it could go and then try NVIDIA and see if it was better. However I began with a simple convolution to get the system up and running

20% of the data was split off for a validation set typically after 3 Epochs less than 3% error was achieved. With more than 3 Epochs the error rate tended to vary suggesting over fitting.

Below is a description of the evolution of the models with comments

Model 1: Basic convolutional network for test of set up

Comment: Just drove in hard right circles.

Model 2: Using Image Flipping to give left and right mirror images

Comment: The car did better but not good

Model 3: Implemented LeNet

Comment: Goes around the track but touches edges maybe good enough

Model 4: LeNet with Flipped images for extra data,

Comment: a bit better but goes off at “brown coloured” or dirt track corner

Model 5: LeNet plus Flipped Images and cropping

Comment: Better again and gets past “brown” corner

Model 6: Same as 5 but got rid of normalisation to see effect

Comment: much worse

Model 7: Same as 5 but got rid of normalisation and zero centring

Comment: Worse than 5

Model 8: Same as 5 with Left and middle cameras added.

Comment: Does steer left and right but goes into water before bridge

Model 9: Lenet with original data plus cropped in preprocess instead of Keras

Comment: Drifts right

Model 10: Same as 9 with cropped and flipped data

Comment: worse than 8 crashed before bridge

Model 11: LeNet , cropped images, 3 cameras,

Comment: funny driving ended up underwater and drove underwater for a while.

Model 12: Same as 11 but deleting image array after preprocessing to free RAM

Comment : same as 11 driving

Model 13: Same as 12 Correction Angle factor 0.1 changed from 0.2 as per lecturer

Comment: slightly worse than 12 but better before bridge

Model 14:Same as 12 Correction Angle factor 0.3

Comment: A bit better did not hit bridge but went off at “brown” corner. Problems at “brown” corner , likely different colour edge creates problem

Model 15: Same as 12 with correction 0.4

Comment : ***was not expecting better but surprisingly it went around the track touched red and white corners a couple of time, good enough to submit probably. Relatively large and perhaps technically “incorrect” correction factor may be successful because it exaggerates steering near edge of road. Visually is appears that car swerves as it approaches edge and large correction factor maybe exaggerating the steering in these cases. The result is hard turns near edges which keeps the car on the track. IT WORKS!***

Model 16 : same as 15 with correction factor 0.5.

Comment: Much worse near immediate crash

Model 17: Tried to use gray scale problems with array sizes and the keras model. So started using NVIDIA without grayscale

Comment: NVIDIA worked at correction 0.5 but touches yellow lines a few times

Model 18: Same as 17 NVIDIA model with correction 0.4,

Comment: did better but crashed on one test at higher speed. Seems like it is good enough at moderate speed. Checked with Slack channel and mentor they said it was good enough. I opt to continue testing until deadline

Model 19: same as 17 NIDIA with correction 0.1

Comment: 0.1 would seem a more natural and technically correct factor however it was actually worse

Model 20: same as 17 Dropout added

Comment:Much worse went off road before bridge, will try dropout again on bigger datasets

Model 21: same as 17 correction 0.4 and no drop out BUT with new track 2 data only

Comment: Near perfect on track 2 but goes into barrier on very last straight. And terrible track one goes off at first corner. Track 2 data is not enough for track 1

Model 22: Tired grayscale again trying to fix array resize problems, failed and reverted to normal images again. Tried New Track 2 data and no Drop out

Comment: Works without dropout not with dropout!

Also I note track 2 data is all middle of road on the broken white line. On track 1 the car drive on the kerb following the kerb with the kerb in the middle of the car. It seems to think the kerb is a white line to follow and follows it closely. I decided to make more data on this assumption to provide left and right lane driving of track 2 to see how it effect track 1

Model 23: Same as 22 with track 2 right lane driving added of one lap.

Comment: Worse it cannot stay in lane well enough and on track 1 it tries to drive between the kerb and the yellow line following them closely as if staying in lane. So it confuses the lines of the kerb and the yellow line as lane lines which is understandable but not what we need.

Model 24: Same with one track 1 and one tack 2 datasets

Comment: Worse at both, centred on kerb in track 1

Model 25: New track 1 data created. 3 laps slow and centred driving, no track 2 data

Comment: Perfect track 1, not bad track 2 crashes in a couple of places but can do most of track 2

Model 26: same as 25 but data is 3 laps track 1 and 1 lap track 2

Comment: Worse on track 1, straddles yellow lines

Model 27: same as 25 but with 3 laps of track 1 and one centre and one right side driving of track 2

Comment: Great at track 2 not good track 1, crashes in one test not another

Model 28: same as 27 but with drop out

Comment: bad at track 1

Model 29: same as 27 with 3 laps track 1 and now left , right and centre driving laps in track 2 no dropout

Comment: Just as bad track 1 not bad track 2 but does crash

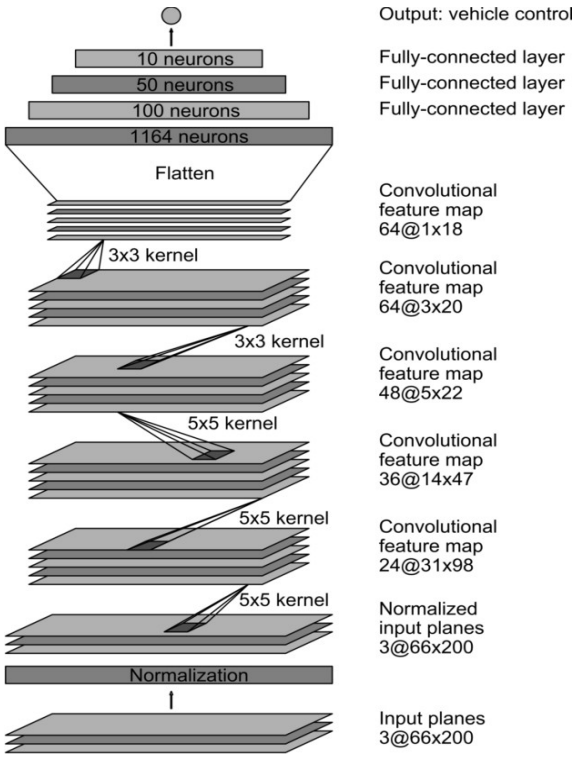
Model 30: For fair comparison same data used with LeNet, 3 laps track 1 and 3 track 2 like model 29

Comment: Terrible on both tracks. On greatly increased data from 1 lap to 6 , Lenet fails indicating more sophisticated NVIDIA does better on bigger data.

Picking the best of 30, Model 25 with 3 laps of track 1 data did the best on track 1 and better than many on track 2. It can actually do track 1 at maximum speed and almost does track 2 at 7mph getting caught on a hairpin turn and if you take over for a second it is fine. So quite good on Track 2 with no track 2 data.

2. Final Model Architecture

The final model architecture is the NVIDIA architecture and shown in the diagram below . ***Image source NVIDIA.com***



3. Creation of the Training Set & Training Process

Initially the default dataset for track 1 was used. This was to allow focus to be on getting all the systems up and running and experimenting with the architectures.

When Lenet had been implemented and seemed to be at a limit then new data was introduced.

3 laps of track 1 were recorded with the car remaining very close to the centre at all times. Slow speed was used to help make the process more precise and easier to achieve.

The new data improved result with the car staying off yellow lines and red and white kerbs when the previous attempt had these issues.

It was suggested in the teaching material that 2 laps of normal driving and one recovery driving would be useful but in this case only normal centre of the road driving was used and that was all that was needed. With the caveat that correction factor in the side cameras was set to exaggerate the

driving angles so when the kerb was approach a hard steering to the centre was produced. This seemed to negate the need for recording recovery data, which was a surprise.

Track 1 data gave reasonable track 2 performance but tight hairpins with barrier not seen in track 12 created difficulty in track 2.

Laps of track 2 were recorded but upon using them track 1 performance suffered. Adding track 2 data of centre driving made the car drive on the kerb and follow the kerb. It was deduced that this was because the kerb was a white line similar to a centre line and the track two data was making the car follow a line.

Laps of track 2 staying in the right and then the left lane were introduce but had the effect of making the car drive between the yellow lines and kerb on track 1. It was deduced that this gap between the yellow lines and the kerb which was just narrower than a car width was being interpreted as a lane and the model keep the car in this "lane" very well but not what we wanted.

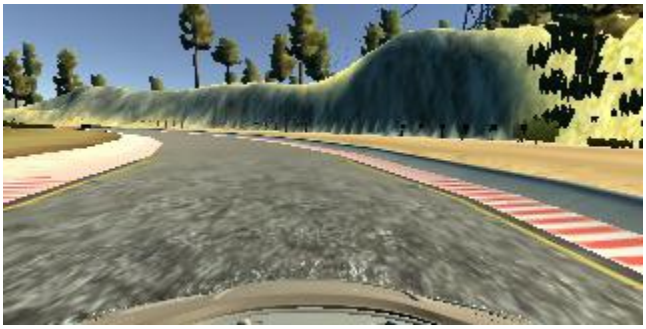
In this way track 2 data had a detrimental effect on track 1 but a positive effect on track 2.

With track 1 data the car would drive well on track 1 and reasonably on track 2.

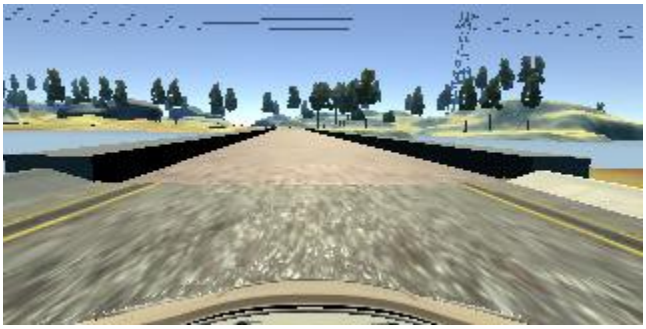
With track 2 data the car would drive well on track 2 and badly on track 1 no mater which variations of data were used.

Initial the data was just the centre camera then the side cameras and flipping to generate extra data were introduced and more than have the models were trained on all cameras and flipped images. (left to right flip)

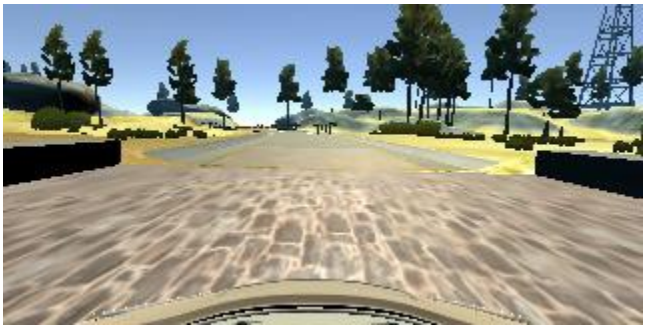
Some sample images are shown below



Above is a typical image from the data set. Note the clearly define edge to the road. This type of well defined road edge was not a problem with the car rarely touching or going over such edges.



Above is an image from the data set showing the approach to the bridge. Note that the edge of the road change in appearance quite dramatically and this often presented problems. If the car got passed the bridge is usually did well.



Above shown the section immediately upon leaving the bridge. Note that there are yellow lines after the bridge that are narrower than the bridge, the better models adjusted the car if it was near the edge of the road to avoid the yellow lines the less good ones did not often going over the lines or mounting the grey kerb at this point.

Above show an example of brown coloured edge. This was particularly troublesome. The unclear edge proved difficult for the models with the less good ones veering off on the brown section. This

ways likely for tow reasons , the colour was less contracting and there were only a couple of small sections with this appearance. Using more data, 3 laps or more, fixed this problem for the better models.

As the need to more data became apparent data from 3 cameras was introduced with a correction factor used for the side cameras. 3 images for the same time period are shown below. Note the relative position of the bonnet (hood to you in USA) in the image.

Images were also flipped from left to right to create mirror images that increased the data set without having top collect more data, as shown below.

Eventually over 100,000 images were getting processed by the models

The data was also shuffled and a 20% validation set split off.

3 Epochs were used as beyond that the accuracy rates seemed to vary suggesting over fitting.

An Adam optimiser was also used