Behavioural Cloning Project

Files Submitted.

- **model.py**
  Contains the script to create, train and save the Artificial Neural Network
- **drive.py**
  Script for driving the car in autonomous mode
- **model.h5**
  contains a a trained convolution neural network (the model used to generate the video)
- **run1.mp4**
  Video recording of the simulator driving in autonomous mode
- **This file**
  This file summarises how the problem has been solved

Final Solution - an overview

The artificial Neural Network is inspired by the one proposed in the video course.

Architecture

- **Normalising layer**: Input images are integer from 0 to 255. It is convenient to have inputs with values around 0 and with magnitude around 1.
- **Cropping layer**: The input images recorded during training contain data that is not useful for driving, for example background trees, the sky, part of the car, etc… Therefore, in order to reduce the dimensionality of the input and to make the Neural Network learn only from useful data, part of the image was cropped
- **3 Convolutional layers** with subsampling and ReLU activation function.
- **2 Convolutional layers** without subsampling and ReLU activation function
- **3 Fully connected layers** with L2 Regularization and ReLU activation function.
- **1 linear output layer**

HyperParameters and other variables

- **Batch size** was chosen to be 32 and never needed to be changed during the experiments
- **Loss function**: The goal is to predict a continuous variable in R. Ideally we want the Neural Network to predict a number as close as possible to the one provided during training. The natural loss function is therefore the **Mean Squared Error**.
- **Optimizer**: It is common to use the Adam optimiser because it has a good implementation of learning rate update and momentum. Only if training does not go as expected it is useful to look at other solutions.
  In this case the **Adam optimiser** served its purpose
- **Learning Rate:** Initial learning rate of 1E-4.
- **Validation data:** 20% of the available data has been used to validate the model on unseen data in order to prevent overfitting.
- **Epochs:** The neural Network showed overfitting after 20 epochs (and it was anyway performing sufficiently well), so the final model has beed trained for 20 epochs in total.

Appropriate training data

I used
- The training data provided as examples
- I personally recorded a few laps driving in the centre of the road both clockwise and counterclockwise
- Several short recordings of the car recovering from the side of the road to the centre.

For more information see next session

Solution Design Approach

Data is everything!

1. A simple start:

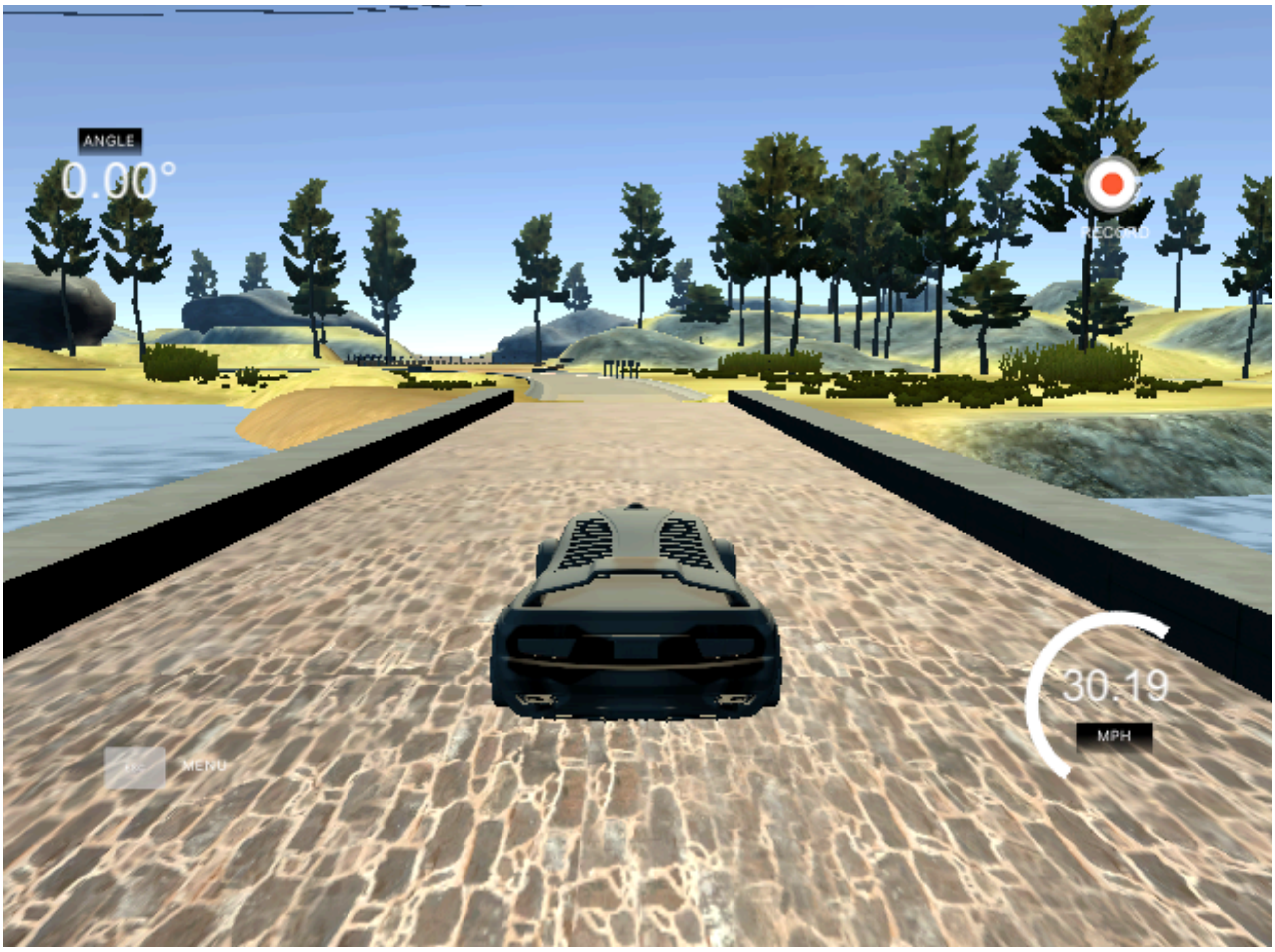
My first approach is always to get something (dirty) working.
I trained a Neural Network with 1 convolutional layer and 1 fully connected layer with only the data provided as example from the course.

What I learned:
- The car starts immediately to turn left and right abruptly.
- The car goes outside the road after a short time.

Not good!

2. More smooth data:

I first wanted to “teach” the car to drive more smoothly, therefore I drove and recorded a few laps around the circuit trying to stay as much as possible at the centre of the road.
Because the circuit is practically just curves to the left, I also recorded a few laps driving clockwise.
Here is an image of the car driving at the centre of the road:

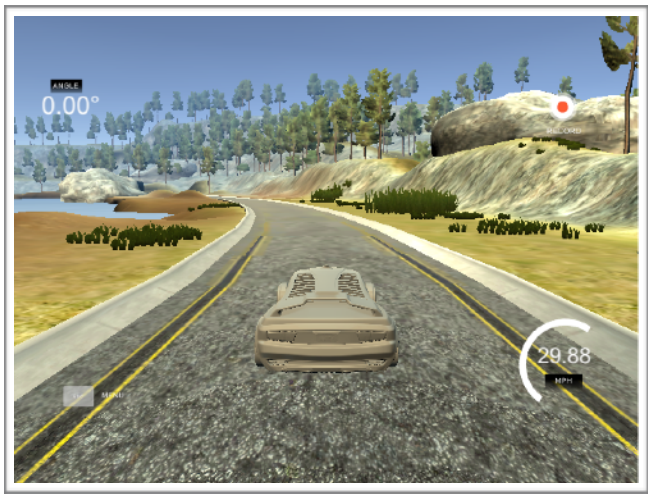


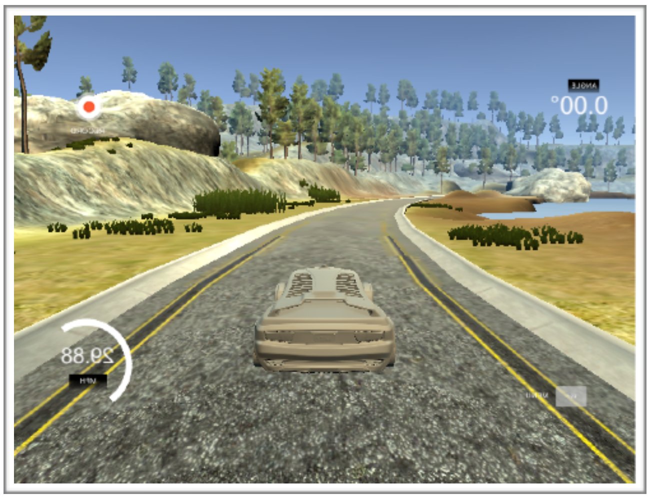
3. Flip the images:

A good trick suggested in the course, was to flip the images horizontally in order to simulate more data available (driving clockwise could have been avoided probably, but this option has not been investigated, so I just kept the extra data.. never harmful)

Here is an example of an image being flipped (left: original, right: flipped)

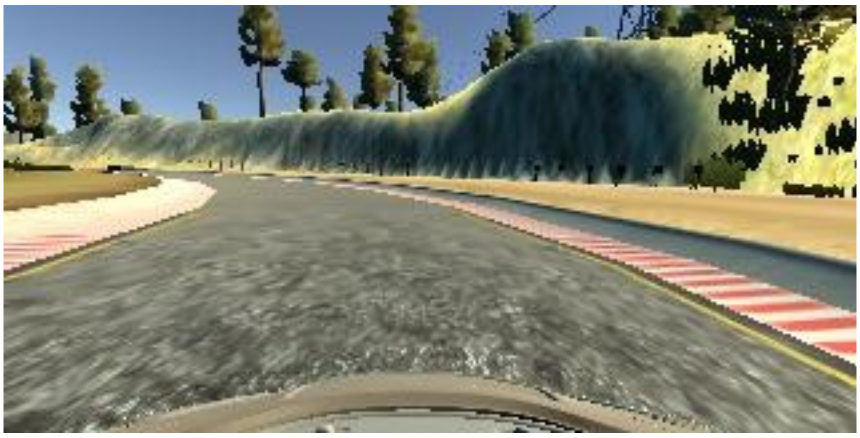
The flipping is performed in line 49 and 50 in the code (see model.py)
The steering angle is of course reversed.





4. Images used for training:

The input images used for training did not contain what has been showed, but only the view from inside the car.
Here an example:



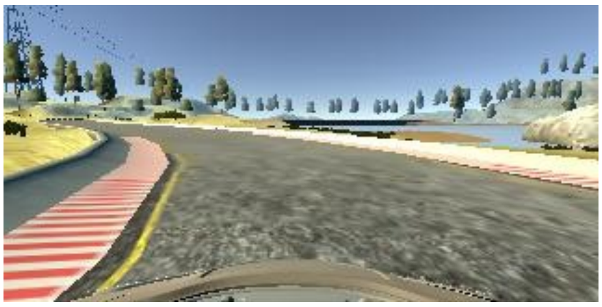
5. An other Architecture for the Neural Net

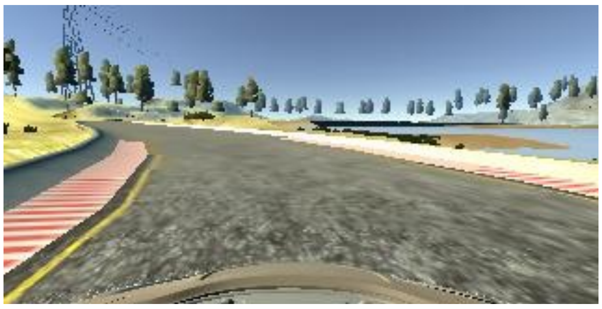
Despite all the new data, the vehicle wasn't behaving well yet.
I decided to try the architecture (described above), provided in the video lectures.

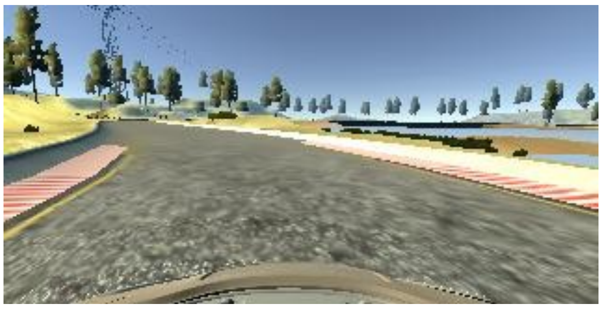
It immediately performed better.

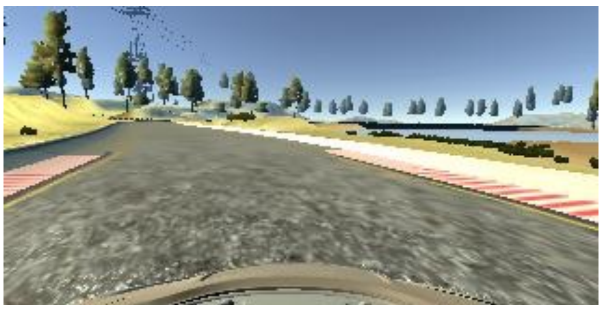
6. Recover from the side of the road

At this point the car was driving quite smoothly but it was always falling inside the river just before the bridge.
The assumption is that the Neural Network did not know what to do if the car was not perfectly in the middle of the road.
I took some time to record several short clips of the vehicle recovering from the side of the road.
Especially where the lane lines change and where I noticed it was failing during tests.
Here is a sequence of images showing the vehicle recovering from the side of the road:









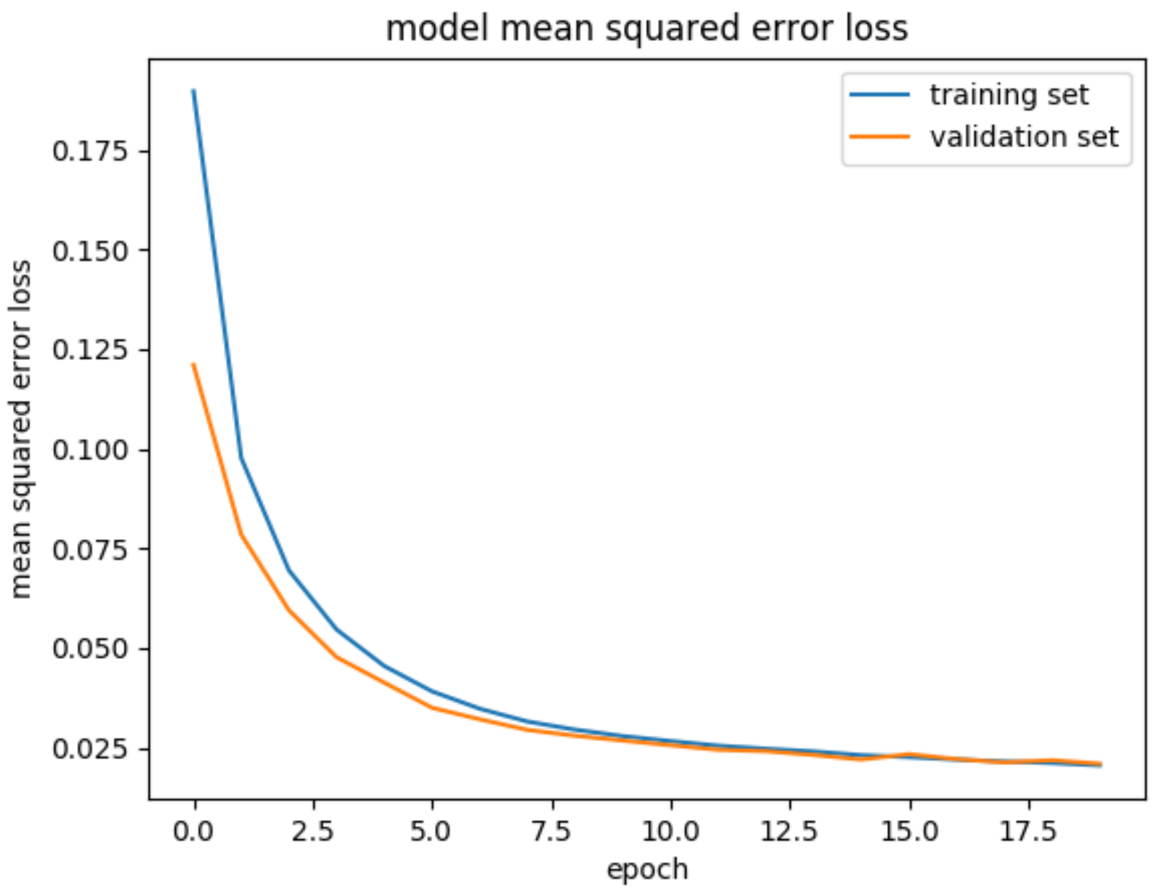
7. Overfitting.. no problem: L2 norm regularization and early stopping

After providing the data of the car recovering from the side of the road, the vehicle started behaving much better.
It would still fail sometimes and drive a bit outside the road before recovering anyway.
More data or regularization?
The Architecture used is powerful and has a high capacity.
This easily leads the Neural Network to overfit.
One solution is to have a lot of training data. In this case, given the time constraints and because recording data is time consuming (despite looking like a video game), I decided to perform regularisation on the Fully Connected layer.
Convolutional layers did not seem to need any regularization, probably because of their nature: parameters are shared therefore are much fewer than in fully connected layers.
I found that adding the **L2 norm** to the cost function with a multiplicative lambda of 0.001, was giving good results.

Plotting the learning curves also helped gaining an insight of whether (and where) the Neural Network was overfitting.
An other useful technique is **Early Stopping**, which consists of training the neural network until the training loss AND the validation loss decrease. When the validation loss starts to increase, stop training.

This is how I decided to train for 20 epochs

The figure below shows the learning curves for the final model:



8. Done

In the video I submitted it is possible to appreciate how the vehicles drives smoothly and almost always in the centre of the road.
A few times it gets close to the sides, but it never touches the lane lines at the shoulder of the road.

Left and right camera view?
It was my plan to use the extra images provided by the left and right cameras, but this has not been necessary.

What about the other circuit?
Again, the mission was accomplished without the need to train on the other circuit.
I am doing it anyway as a personal experiment.