Project Title: 3D Self Driving Car with Imitation Learning (Behavior Cloning)
Project Author: Daniel Newman

Aim:

The aim of this experiment is to explore new methods in teaching a neural network to learn. In this case we will be exploring a simplified version of Imitation Learning called Behaviour Cloning. This method of teaching will be tested on a self-driving car in an 3D environment to determine how it performs.

Hypothesis:

We believe that the self-driving car will be able to navigate the track for a period of time before course correction will lead to the car to collide with the environment. This belief stems from our knowledge of Behaviour Cloning's limitations.

Experiment

Requirements

Python 3.6

Python Libraries

- Numpy
- SciPy
- Reinforcement Learning Library
  - Tensorflow
  - PyTorch
  - Theano
  - etc
- OpenCV
- OpenAI Gym

Python related IDE

- Spyder
- Jupyter
- Pycharm

Experiment Specific Requirements

Driving Simulator – Github: https://github.com/udacity/self-driving-car-sim
End to End Behaviour Cloning Paper:
https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

Steps to Reproduce

Firstly we had to generate our dataset that our self driving car would evaluate and learn with. Our simulator had 3 camera's, one on each side and one in the front which recorded images of the road ahead. Our training footage was comprised of completing a few laps within the simulator whilst our cameras recorded the road footage. This gave us a total of 3 laps worth of content, or approxamitely 200,000 images with their accompanying steering angle & velocity data, which was used to understand just how much input was being given to achieve the associated turn, and any throttle or braking changes applied.

This data was then processed through our Convolutional Neural Network, which was comprised 5 layers of expanding nodes, all configured with the Rectifier activation function. This handled our image processing which was then flattened and run through a regular Neural Network with 5 layers, also configured with the Rectifier activation function. The entire network was based upon Nvidia's paper on End-to-End Learning for Self-Driving Cars, which made up much of the codebase.

In order to ensure that the code could be run on an everyday PC, we needed to batch the images to cut down on the required amount of memory we needed to utilise and process using our CPU.

This reduced our overall processing size from our 200,000 images to 64, which made processing on a regular PC manageable. However we had to do some processing on the images due to the overwhelming difference between images that showed a straight road, which would naturally be more prevalent, and turns.

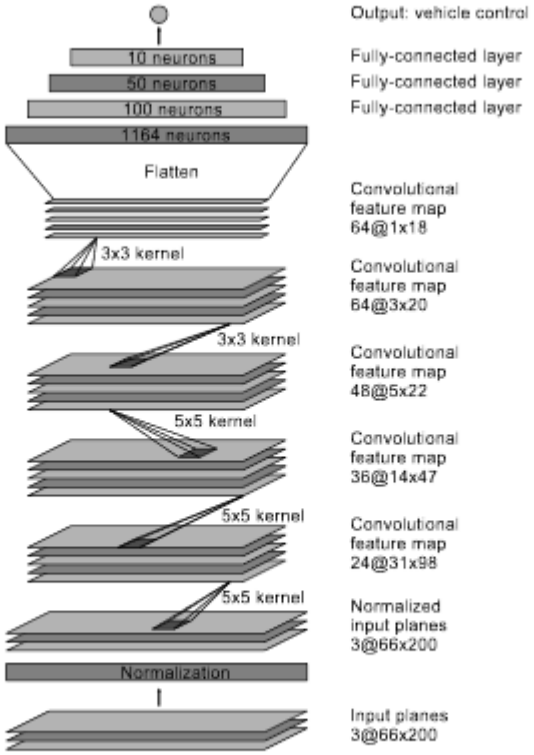
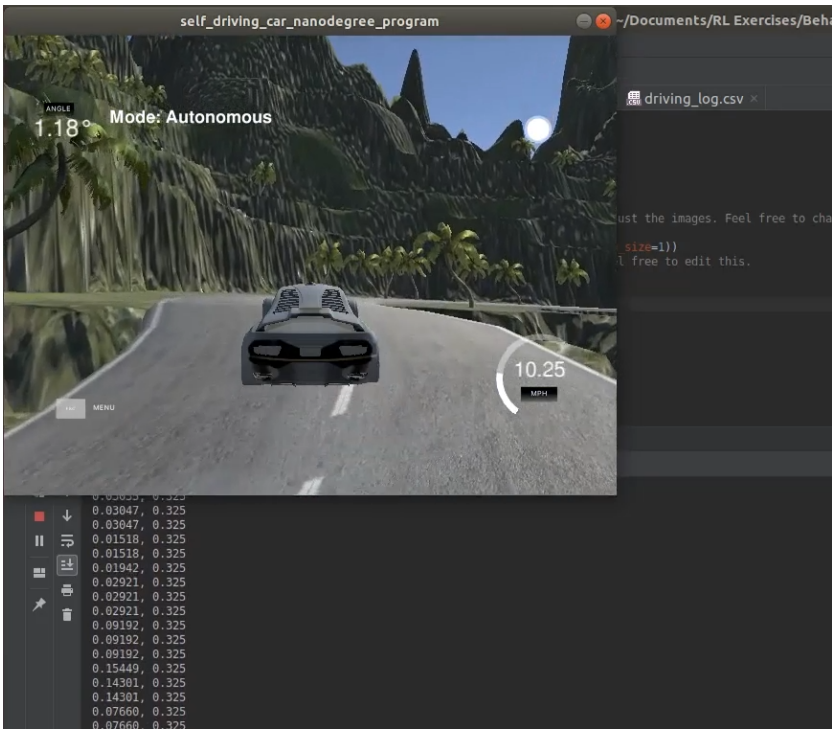


Figure 4: CNN architecture. The network has about 27 million connections and 250 thousand parameters.

We accomplished this by randomly applying a shear left or right to our centre images, ensuring that our images would appear as a corner rather than a straight. In doing this we could balance out some of the discrepancies with the dataset.

Observations:

After a few adjustments to the acceleration multiplier and steering coefficients, we arrived at a relatively accurate representation of our initial dataset. As the vehicle moved through the environment, we observed that its steering corrections were quite erratic and often lead to overcorrection. This overcorrection would often lead to the vehicle attempting to recover itself by applying an increasingly larger steering input, which ultimately lead to the vehicle colliding with the environment.



Discussion

The limitations in the agents behaviour was hampered by its erratic input, this is due in part to the input system we used to create the dataset, using a keyboard resulted in digital signals, which meant that the steering data was often represented by pulses of activity and long sections of nothing. This is a limitation of the hardware that was available during this experiment and could be mitigated by either smoothing out the steering input data over the frames or utilising an analogue input device such as an Xbox controller or similar.

This was further exacerbated by the way Behaviour Cloning operates, as it applies an action each frame and cannot forecast into the future more than a limited number of frames. This causes any error in the model data to perpetuate until it becomes irrecoverable, since course correction and adaptation isn't possible due to the nature of Behaviour Cloning, which attempts to mirror the user data exactly at that moment, with no regard for future events, this could be addressed using a more complex version of Imitation Learning, but would require further investigation.

Conclusion

Here we have uncovered how to utilise Neural Networks to learn from human data under the application of teaching an agent to operate a car to navigate a road course. With further investigation, advances in this current field could grow to provide better performance with a smaller dataset, but addressing the hurdles of generalisation and techniques that prevent over-fitting our model to the dataset are critical to facilitating those advances in Reinforcement Learning.