Deep Learning Lab - Report - Exercise 3

Rabea Laura Turon

December 4, 2018

1 Introduction

This report summarizes my findings of Exercise 3 of the Robotics track of the Deep Learning Lab Course at the University of Freiburg. The goal of the exercise was to implement a Neural Network that clones the behavior of an expert driver (in this case me) in the CarRacing game of the OpenAI Gym benchmark suite.

2 Expert data

In order to train a network on expert data the first step was to create some expert data. For this I first practiced how to drive the car in the CarRacing game and then collected the expert data with the given drive_manually.py file. For the training in the next part the images of the CarRacing environment are used as input and the actions performed on these images are the outputs the network should learn to predict. I created 30,000 of these input-output pairs but only used 10,000 of them for the final training due to some memory issues when using all the data. The mean reward of the expert data was about 880 with a standard deviation of 31. For the network training I transformed the actions into five discrete actions (doing nothing, acceleration, left, right and brake) that were then translated into a one-hot encoding. Note that the data set did not actually contain any brake actions which is why four classes would have been sufficient.

3 Training the cloning model

The resulting agent of the first model always chose to do nothing which made sense since most of the training data consisted of the action doing nothing and only few samples of the other actions existed. Therefore I adapted the minibatch sampling method to sample such that every minibatch consists of roughly the same number of samples for each action (except brake because I did not have any samples for braking).

Unfortunately I underestimated the time for finding an appropriate model which is why I do not have any working cloning agent. I tested several architectures for the model with 2 to 4 convolutional layers (and maxpooling between them) with a kernel size of either 3 or 5 and 16, 32 or 64 kernels. I was not able to train the models on a GPU because of some Cuda errors on the pool computers and therefore had to use my own laptop. This meant rather long training times (for 100 steps about 15 to 40 min depending on the size of the network) and I did not find any network that performed well. When testing the models with test_agent.py they either did nothing or only accelerated.

Note that all these mentioned models only used one image as input and were trained for a maximum of 200 steps. Most models already seemed to have converged after about 100 steps with an accuracy of 50-60% on the validation set. I did not get to testing models with a history of several images as input.