**Laboratory - Deep Learning Lab, WS 2018/2019 - Exercise 4**

**Submitted by: Amadeus Hovekamp,**
**mail@amadeus-hovekamp.de,**
**Matr. no: 4603934**
**Amadeus Hovekamp,**
**mail@amadeus-hovekamp.de,**
**Matr. no: 4603934**

The repository with the source code can be found under the following link:
https://github.com/Schokokugel/RoboticsLab

# 1 Getting Set Up

As the set up for this exercise was the same as for the last exercise, there were no issues here.

# 2 Reinforcement Learning: Deep Q-Networks

## 2.1 CartPole

All in all the implementation of the DQN went well, the provided components helped to implement it. We implemented some additional features that helped us training the network and generating the required data for generating the report.

The first runs were not successfull due to the small default learning rate of 0.0001. Changing the learning rate to 0.001 lead to good results, which can be seen in the following figures 1 and 2.

The total run had a length of 500 episodes, the reward for each episode is displayed by the dots in figure 1. The orange line shows the evaluation reward after every tenth episode, averaged over 5 evaluation runs using only deterministic actions.

It took the agent about 120 episodes to learn how to prevent the pole from tilting but still drove off to the side quickly. After 180 it was sometimes able to balance the pole for the complete episode consisting of 1000 steps. 40 episodes later it managed to do so consistently. Somewhat suprisingly it *unlearned* to how balance the pole after 60 episodes later. This could be due to more and more of the same states (the pole is almost upright, neither cart or the pole move much) filling up the replay buffer and thus overfitting the network to highly similar states. This would also explain the learning-unlearning oszillations later on. A hint in the source code defined the task as *solved* when the average reward is greater than or equal to 195.0 over 100 consecutive trials, which was reached after 170 episodes.

Figure 2 shows the test rewards achieved with the trained networks over 15 episodes, once where the network was trained until it is conisdered as solved, and once after training for the full run of 500 episodes. It is visible, that the longer trained network achieved better results.

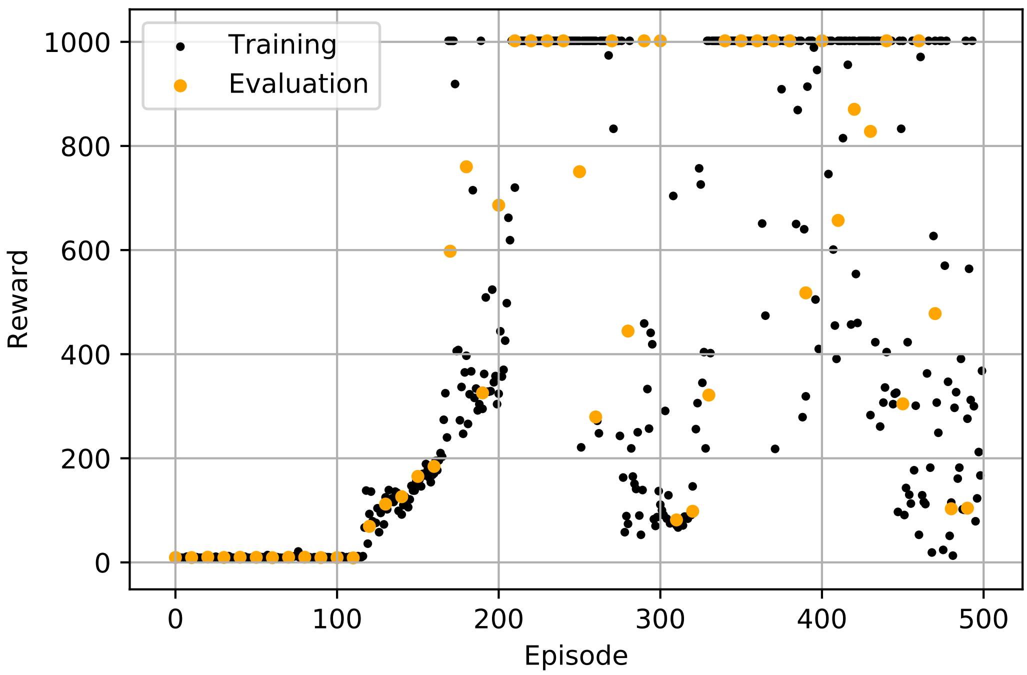


Figure 1: CartPole - Episode Reward during Training

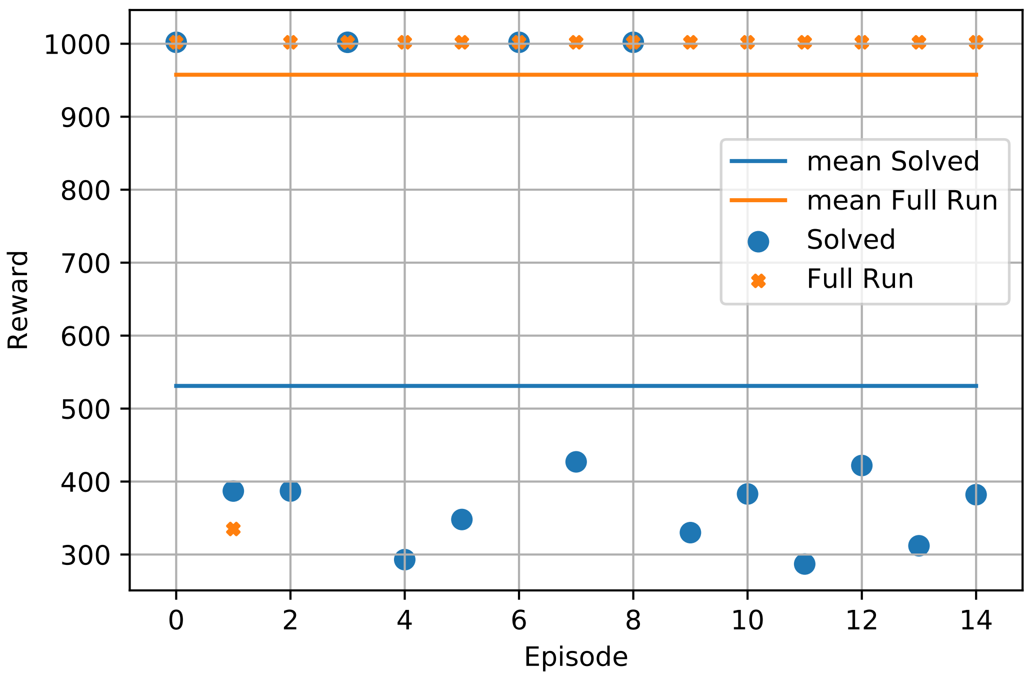


Figure 2: CartPole - Episode Reward for Testing

## 2.2 MountainCar

For the MountainCar environment we used the same setup and network as for the CartPole environment, leading also to good results, which are displayed in the figures 3 and 4.

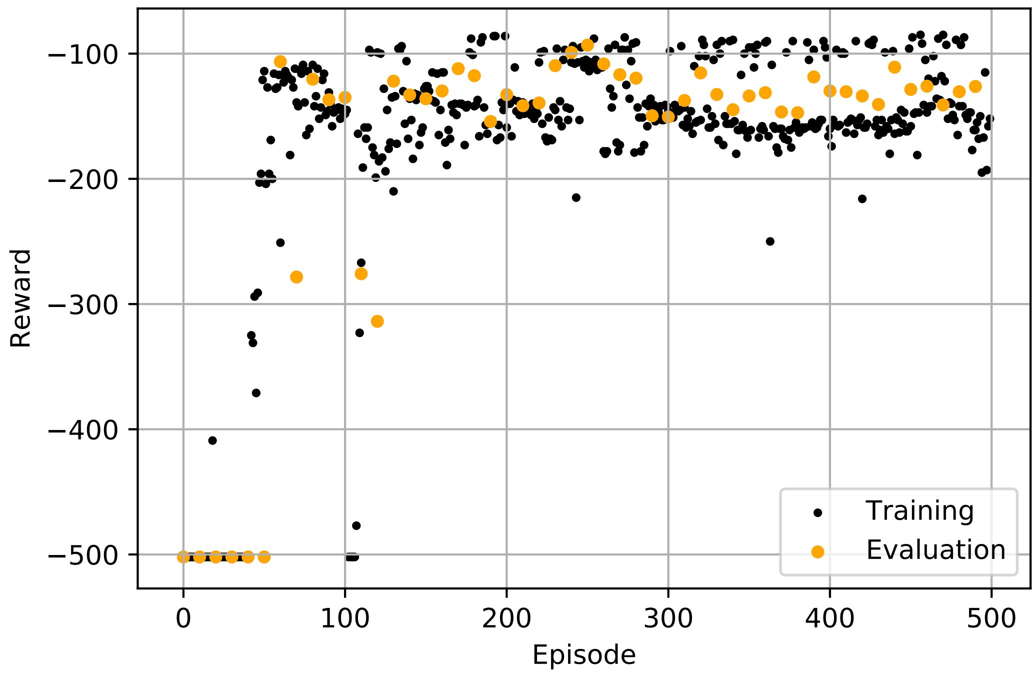


Figure 3: MountainCar - Episode Reward during Training

In figure 3 the training and evaluation rewards are displayed as in the CartPole environment. Here, already after 60 episodes the network performs good. But later on again an oscillation behavior can be seen, though the bad performances are not dropping are not getting as bad as some performances for the CartPole. A reason for this difference is probably that the CartPole is more difficult, consisting of two tasks at the same time (balancing the pole and not drifting off to the side).

Finally, in figure 4 the test rewards for the solved model and the full run model are displayed. Here, we have defined solved as reaching an average score larger than -200 over 100 consecutive runs, similar so the definition as for the CartPole. In contrast to the CartPole, Solved scores higher on average but the Full Run, which is in turn better when the starting conditions are favorable for its risky driving style trying to minimize drives trough the valley.

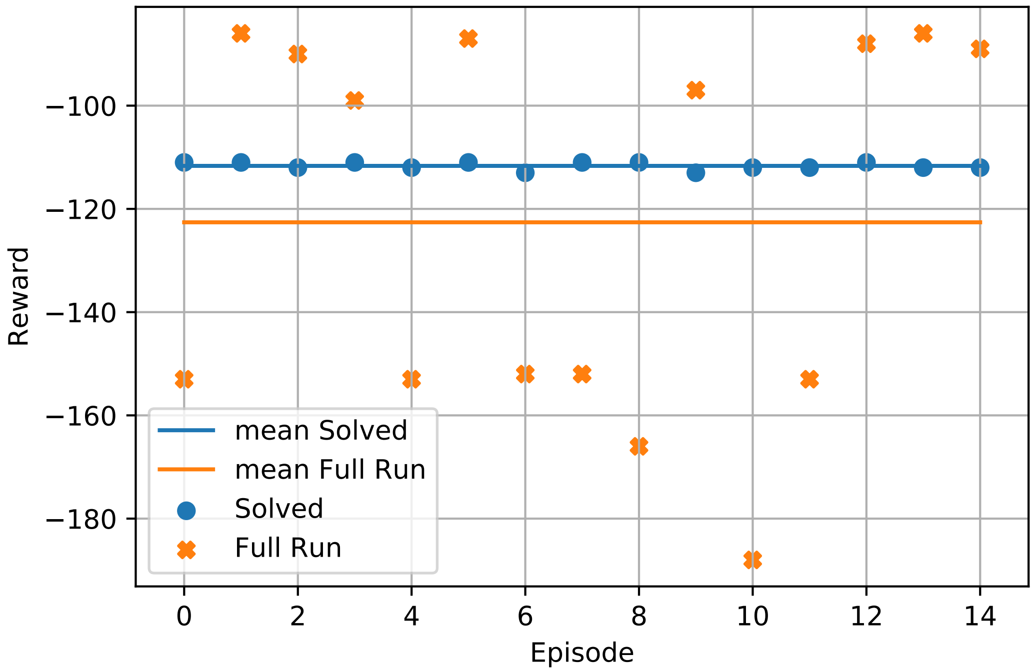


Figure 4: MountainCar - Episode Reward for Testing

### 2.3 CarRacing

| | epochs | mean | std | max | min |
|---|---|---|---|---|---|
| **small data set** | 6 | 917.92 | 8.28 | 929.50 | 902.30 |
| **big data set** | 36 | 891.40 | 39.48 | 931.70 | 796.20 |

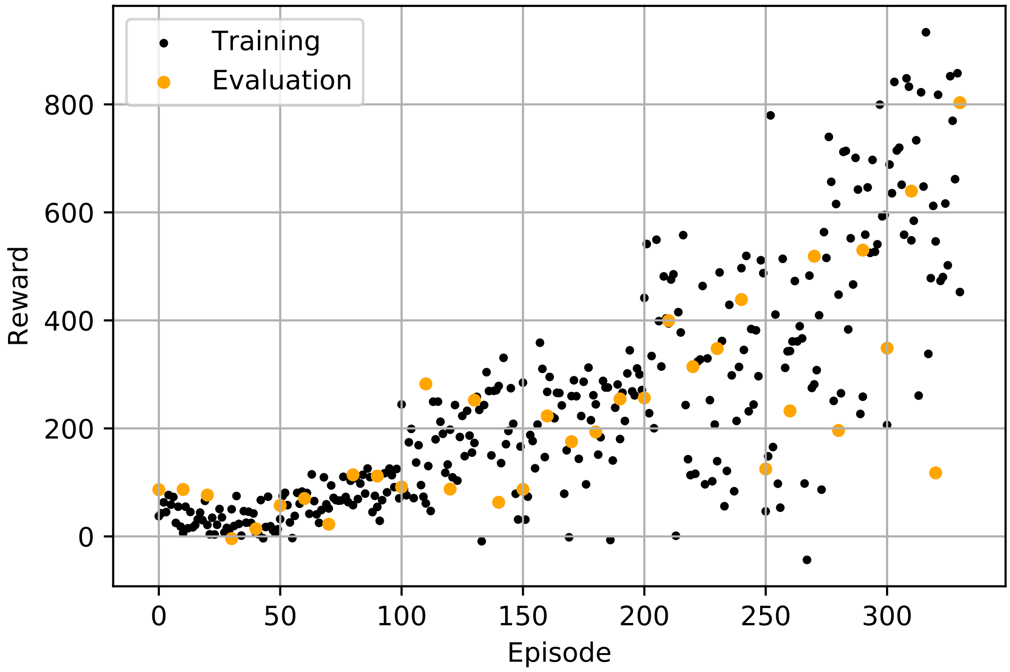


Figure 5: CarRacing - Episode Reward during Training

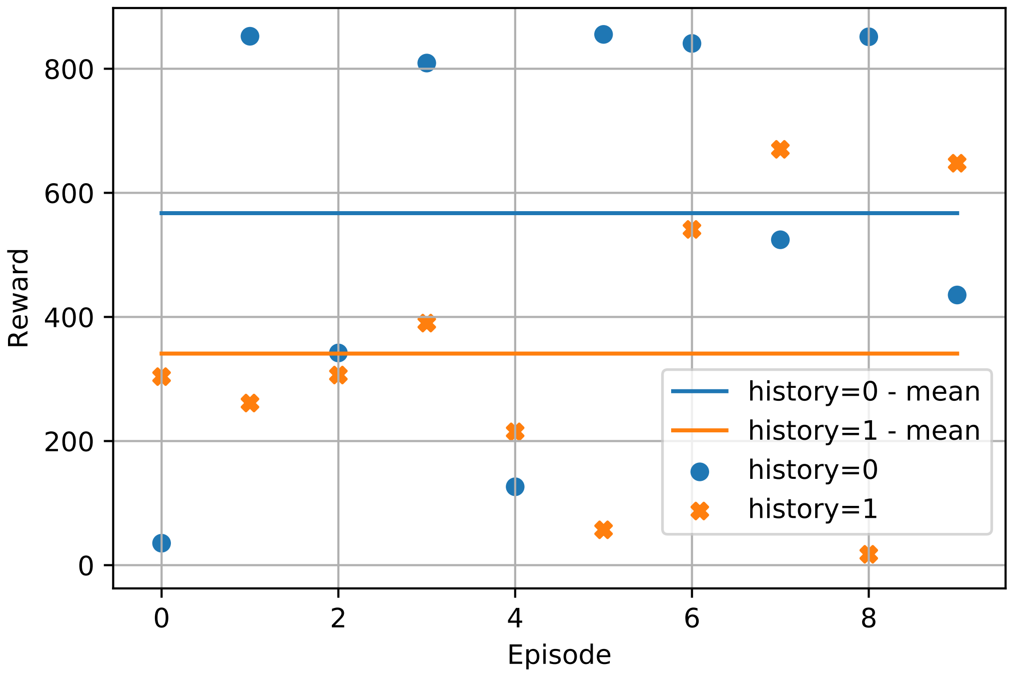


Figure 6: CarRacing - Episode Reward for Testing

### 2.4 An improved network

## 3 Influence of hyperparameters