Faculty of Engineering
Albert-Ludwigs-Universitat
Autonome Intelligente Systeme - AIS
Wintersemester 2017/18

Deep Learning Lab Course

2017

Assignment 4

Author: Lior Fuks (4251285)
Markus Merklinger (4211461)
E-mail: liorfuks@gmail.com
merklingermarkus@gmail.com
Tutors: Aaron Klein,
Artemij Amiranashvili,
Mohammadreza Zolfaghari,
Maria Huegle,
Jingwhei Zhang,
Andreas Eitel
Data of submission: January 22, 2018

Deep Learning Lab Course 2017
Assignment 4

January 22, 2018

Abstract

In this assignment implemented a deep Q-learning neural network so that a agent can solve a grid work maze. We achieved a very high success rate of 0.93 only based on a local view and a history. The performance is not good as astare but can be even improved by running more episodes.

Contents

1 Introduction

In this exercise we used a convolution neural network to perform a planner task in a grid world based only on his local view. The training is done by using a model free reinforcement method called Q-learning, which predicts the best action at each state. The Q-learning prediction is then used in order to train the CNN by minimizing its error.

2 Implementing and training an agent using CNN

The network used in this exercise is similar to the deep mind Atari network. The input of the network is composed of the local view of the agent with a given history length. The network output is the predicted best action for the agent at its current state. The architecture of the network can be seen in Table 1. A history length of 4 we leads to trainable parameters 3,336,197.

The implementation can be found under https://github.com/RedHeadM/dl_lab_2017.

Table 1: CNN layer structure

Layer	No. of filters/Units	Kernel size	Stride	Activation function
Convolution 1	32	2x2	2	Relu
Convolution 2	64	3x3	1	Relu
Convolution 3	64	3x3	1	Relu
Fully connected 4	512			Relu
Fully connected 5	5			linear

3 Training

The agent is placed randomly in a grid world with a fixed goal. It can perform different movements and receive a reward of -1 in case of a collision and +10 if the goal is reached and -0.04 in every other case. Each game episode has few stop criteria: reaching the goal, reaching maximum of 75 steps or total reward smaller then -10. At the first stage the agent explores the map randomly and updates its Q values without training the network. Afterwards the agent is alternating between exploration and exploitation governed by the epsilon value, which determines how often the agent will choose a random action and how many times it will choose the best calculated action. The epsilon value is then slowly decreases, causing the agent to explore less and exploit more. The weights of the network is then trained every step, where the weights of the target network is updated only after each episode. The network is trained for 700 episodes using Adam optimization algorithm with a learning rate of 1e-5. and a batch size of 32.

3.1 Results

In figure 1 we can see that total reward and the total number of steps per episode.

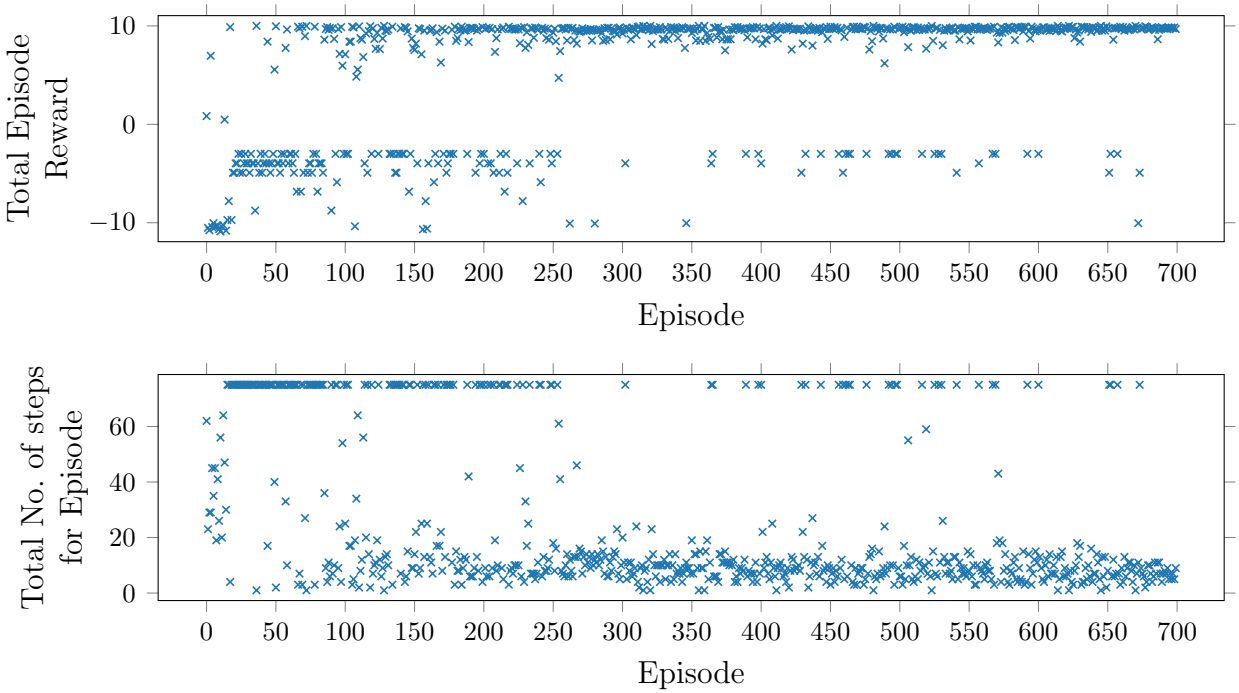

Figure 1: Agent training for performance for 700 episodes.

4 Testing

After the training phase, we evaluate the agent performance by computing the success rate and comparing the amount of steps taken with A*.
No exploration is performed by the agent. The success rate is 0.902 for 1000 episodes, while the mean difference of steps compare to A* in successful episodes is 4.4 steps. In Figure 2 we can see the total reward per episode as well as the number of extra steps compared to A* for each successful episode.

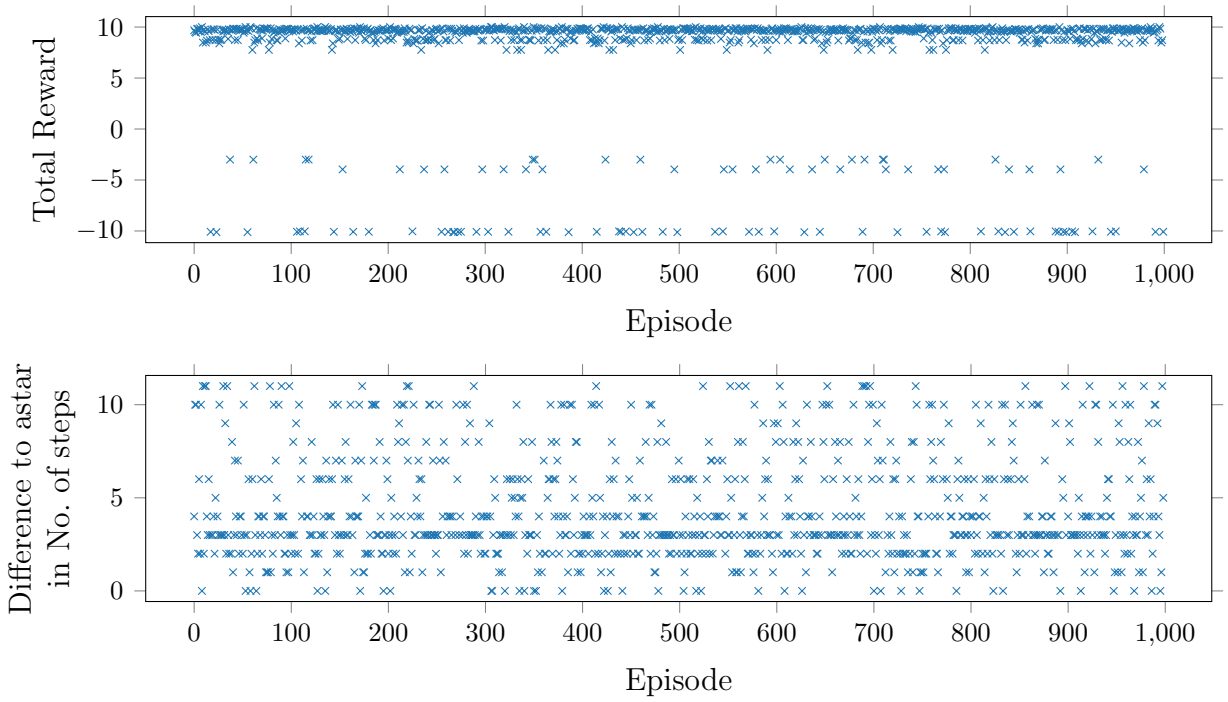

Figure 2: Learning progress of the agent

5 Discussion

In the Figure 1 one can see the different stages of the learning progress of the agent. After the first 20 episodes the agent learns to avoid collision with the walls since the total episode reward of -10 is not reached anymore and the maximum steps for an episode is reached. After around 300 episodes the agent has increased the total reward close to the maximum possible, therefore it could reach the goal.
In Figure 2 we can see that the agent has a very high success rate, but that the solution is not yet optimal as it has an average of 4.4 steps more then A*. Training the agent for more episodes would lead to better performance.

6 Conclusion and summary

We could implement a the deep q leaning algorithm and some the gird maze problem for an agent only based on a local view and history with a very high success rate of 0.93.

List of Figures

List of Tables