Deep Reinforcement Learning and Cellular Automata

Tasiopoulos Ioannis

Technical University of Crete
Electrical and Computer Engineering
Autonomous Agents 2023-2024

14 March 2024

Abstract

Deep reinforcement learning (deep RL) is a subfield of machine learning that combines reinforcement learning (RL) and deep learning. There have been numerous examples where the use of it led in remarkable performance in games like Go, Dota 2, Atari etc. In this assignment Deep Q-Network (DQN) is applied to solve gym-cellular-automata, a Gymnasium's third party environment, where the agent interacts with Cellular Automata by changing its cell states. This automaton simulates an active forest fire. The objective is to judiciously remove trees from the forest in order to prevent fire from spreading to neighbouring cells, thus to minimize loss from burned trees. DQN achieves far better results when comparing with random agent, successfully confining the fire.

Contents

1 Theory **2**

1.1 Reinforcement Learning 2

1.2 Deep Reinforcement Learning 2

1.3 Q-Learning 3

1.4 Deep Q-Network 3

1.5 Cellular Automata 3

2 The environment **4**

3 Implementation **6**

3.1 9x9 grid 6

3.2 84x84 grid 7

4 Evaluation **11**

4.1 9x9 grid 11

4.2 84x84 grid 12

5 Conclusions **20**

5.1 Future work 20

Chapter 1

Theory

1.1 Reinforcement Learning

Reinforcement Learning is a third machine learning paradigm alongside supervised and unsupervised learning. [5] The core idea is that an agent learns which action to take in each state of it's environment. The reward for each action is yielded by experience. The goal is to maximize the expected reward for each action.

One important challenge that arises in reinforcement learning is the trade-off between exploration and exploitation. The agent must find the actions that produce the best results. In order to find them, it has to *explore* the environment. Doing so means taking actions that may not be effective. Simultaneously, past experience from actions should be *exploited*. In the long term, actions that yield the biggest reward should be favorably chosen. However, neither exploration nor exploitation can be pursued exclusively without failing at the task.

1.2 Deep Reinforcement Learning

Deep reinforcement learning is the combination of deep learning and reinforcement learning[3]. *Deep* comes from the deep neural networks used to find approximations for high dimensional environments like image or voice inputs.

1.3 Q-Learning

Q-Learning is a model free, off policy Reinforcement Learning algorithm ([6]), that tries to maximize cumulative future rewards. Model free means that the environment and its dynamics do not have to be known and the agent learns from experience. Off policy means that the agent learns the value of optimal policy regardless of the policy being followed. The agent estimates the value (Q-value) of each action in each state. Updates take place using the Bellman equation, which expresses the optimal value as the sum of the immediate reward plus the discounted future reward, of taking the best action in the next state. Usually, ϵ-greedy is used to balance exploration-exploitation. This algorithm under circumstances is guaranteed to converge to the optimal action value function(Q-function)

1.4 Deep Q-Network

Deep Q-Network is an approach developed for applying Q-Learning with experience replay and a convolutional neural network for raw visual inputs in Atari 2600 games [2]. Experience replay is a mechanism used to store previous transitions of the agent in the game. A random sample (batch) is taken in each training step, which smooths the training distribution. The raw input is grayscaled, resized and cropped before being fed to the network to reduce computational cost.

1.5 Cellular Automata

Cellular automata (CA) are discrete, abstract computational systems composed of finite, homogeneous cells [1]. At each time step they evolve based on some update function or transitional rule. There are two common types of neighborhood in two-dimensional square lattice. Von Neumann, comprising of the central cell and its four adjacent cells and Moore, composed of the central and the eight cells that surrounds it.

Chapter 2

The environment

Gymnasium is a maintained fork of OpenAI's Gym library. It provides an API for all single agent reinforcement learning environments, and includes implementations of common environments: cartpole, pendulum, mountain-car, mujoco, atari, and more. There are also third-party environments that comply to Gymnasium's API. Gym-cellular-automata [4] is such an environment.

It has two kinds of environments, both of them evolving in a m x n grid . Each environment has a standard configuration for benchmarking RL algorithms and a prototype configuration for experimenting. The most parameterized variable is the grid size, which is a proxy for task difficulty.

The first environment *ForestFireHelicopter* simulates an ever evolving fire in the grid. The task is to put down the fire controlling the helicopter. The standard configuration of the grid is 5x5.

The second environment *ForestFireBulldozer*, which this assignment is about, simulates a wild fire. The task is to extinguish the fire by controlling a bulldozer that removes trees to prevent its advance. At each time step the bulldozer moves to a cell within it's Moore's neighborhood and can remove a tree, thus resulting in eighteen different combinations of (move, removal decision). The simulation ends when the fire is extinguished either by the bulldozer or by the complete destruction of the forest.

Every cell in the grid is in one of the following states with a corresponding value:

- empty, with value 0
- tree, with value 3
- fire, with value 25

The reward is maximized when the forest is kept with its maximum number of tree cells. The standard configuration of the grid is 256x256.

Chapter 3

Implementation

The approach that was used to solve gym-cellular-automata ForestFireBulldozer environment was implementing a Deep Q-Network. The grid size used for experimenting was 9x9 and 84x84.

The first step that was taken, towards implementing the DQN algorithm, was to embed the position of the bulldozer in the observation array, that is returned from the environment. Each element in the observation array has three possible values as described in Chapter 2. The state fed into the network is the same as the one returned from the environment, except for the cell where the bulldozer is currently located, which has a value as follows:

- $1 \rightarrow$ empty cell and bulldozer
- $2 \rightarrow$ tree cell and bulldozer
- $4 \rightarrow$ fire cell and bulldozer

Also, the observation state is normalized before feeding into the network.

The network makes a prediction for eighteen different actions, as mentioned in Chapter 2.

3.1 9x9 grid

Grid size 9x9 was tested with the following configuration:

- $\epsilon_{min} = 0.1$
- $batchsize = 32$
- $\gamma = 0.99$
- $replay\ memory\ capacity = 1000$
- $target\ network\ update\ frequency = 1000$

The neural network used to make predictions, as well as the target model had the configuration as shown in Figure 3.1.

3.2 84x84 grid

Grid size 84x84 was tested with the following configuration:

- $\epsilon_{min} = 0.1$
- $batchsize = 32$
- $\gamma = 0.99$
- $replay\ memory\ capacity = 100000$
- $target\ network\ update\ frequency = 10000$

Two models were used for this grid size, as shown in Figure 3.2 and Figure 3.3.

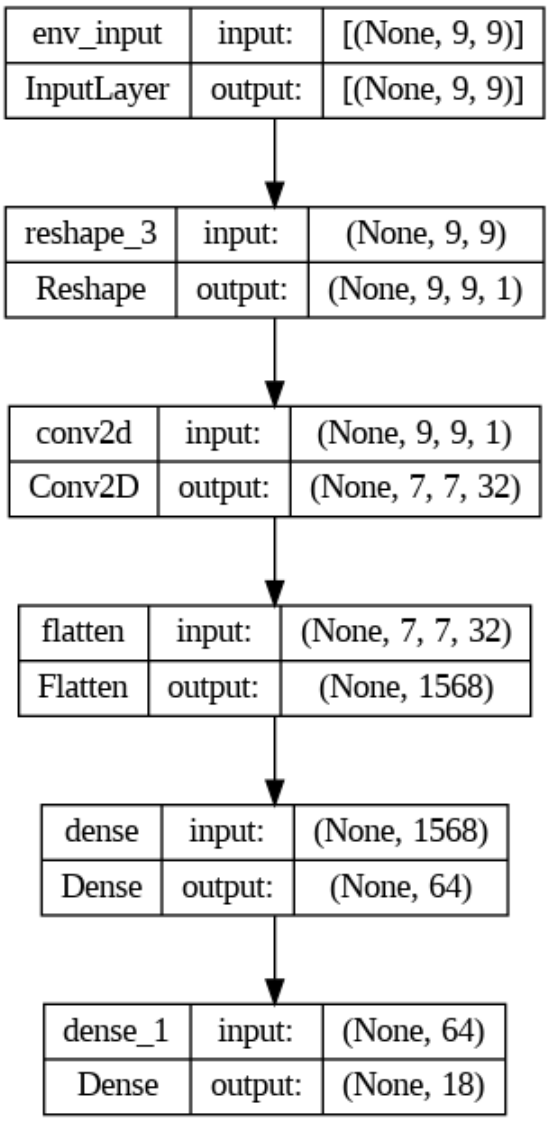

Figure 3.1: Neural network configuration used for 9x9 grid

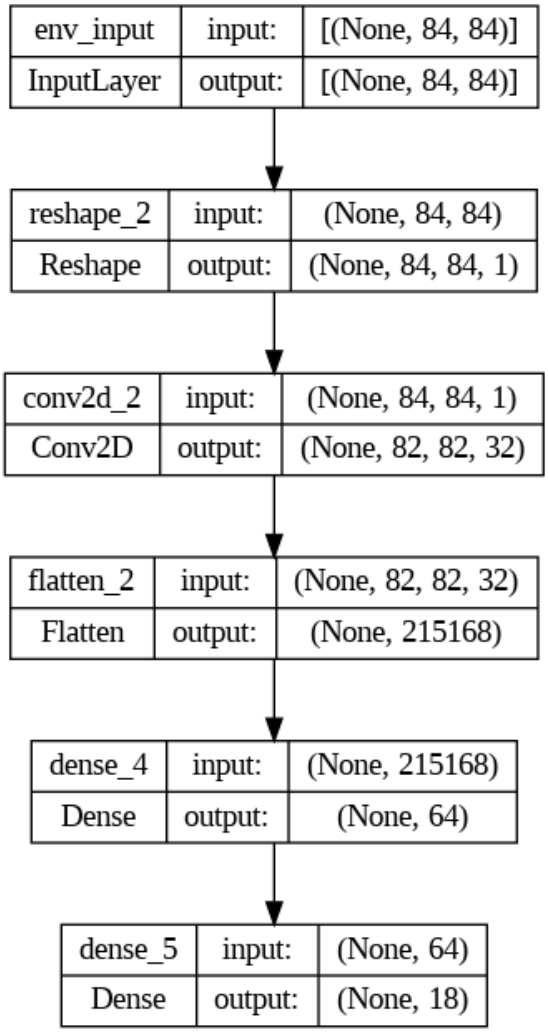

Figure 3.2: Model 1 used for 84x84 grid

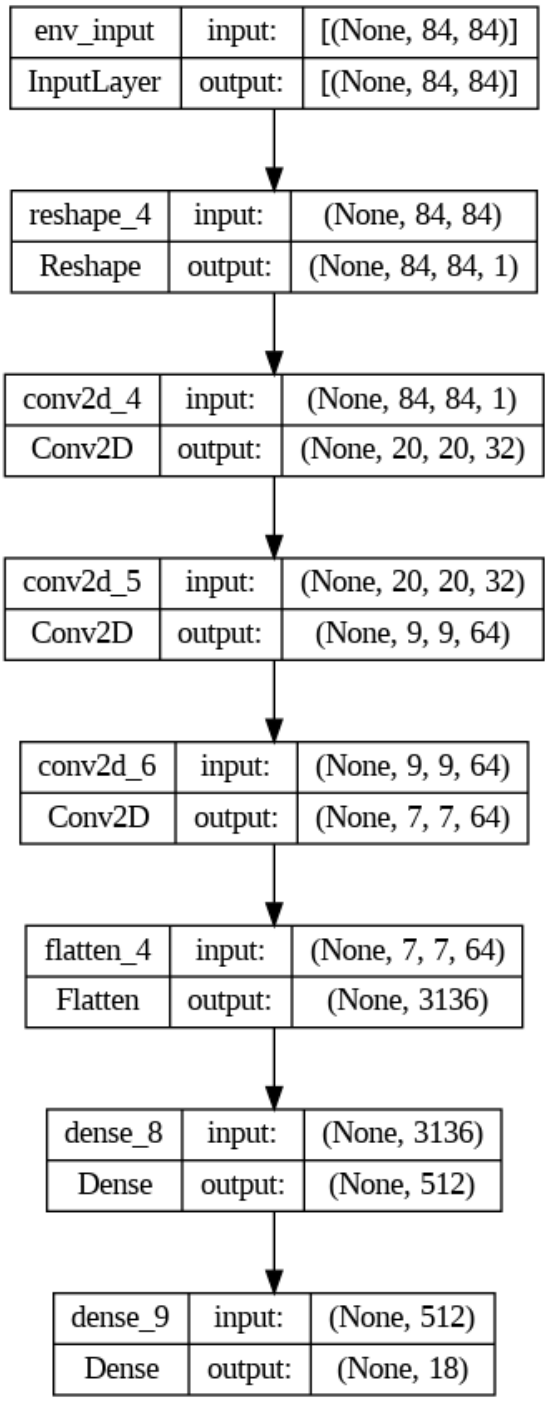

Figure 3.3: Model 2 used for 84x84 grid

Chapter 4

Evaluation

The models used were trained for at most 1000 episodes. Every 100 episodes models were saved and put under test. Both models during training took the first 300 episodes random actions, in order to explore their environment. In this chapter the results of the training will be presented. It is worth noting that the evolution of fire is different in each simulation as well as the environment itself (i.e. the placement of tree or empty cell is not fixed). This results in some simulations to have premature ending and thus, minimal losses.

4.1 9x9 grid

Simulation for grid size 9x9 took two - four steps on average until it was finished. This number of steps is small and an agent's actions have little effect in the result. Apart from the models trained for 100 and 300 episodes the others have similar performance.(see Figure 4.1). However, even in this small grid model's performance(see Figure 4.2) is promising to have greater difference in a larger grid.

4.2 84x84 grid

In this setup 300 - 500 steps were needed for each episoded to end. As a result, the agent has the chance to act effectively and make an impact in the outcome. Three different version were trained in this setup. Model 1 shown in Figure 3.2 was trained in each steps of an episode. Model 2 shown in Figure 3.3 was trained in two manners. In each step (version 1) and every two steps (version 2). Each model performed its best at different number of training sessions.

As shown in Figure 4.3 model 1 agent trained for 200 episodes is performing better than the others.

Similarly, model 2 (ver.1) performs better when trained for 800 episodes (see Figure 4.4) and model 2 (ver.2) when trained for 900 episodes, as shown in Figures 4.5 and 4.6.

Finally, best models for 84x84 grid configuration where put under test together with Random agent (see Figure 4.7). The results show that all of the models behave much better than the random agent. The comparison of the models show that model 1 has similar performance as the model 2 v1. Also, model 2 v2 shows slightly worse performance than the others. This is probably caused is due to the fact that it is trained every two steps and not in every step as the other models. However, it required only the half training steps, thus less computation and time, and it achieved a very similar performance.

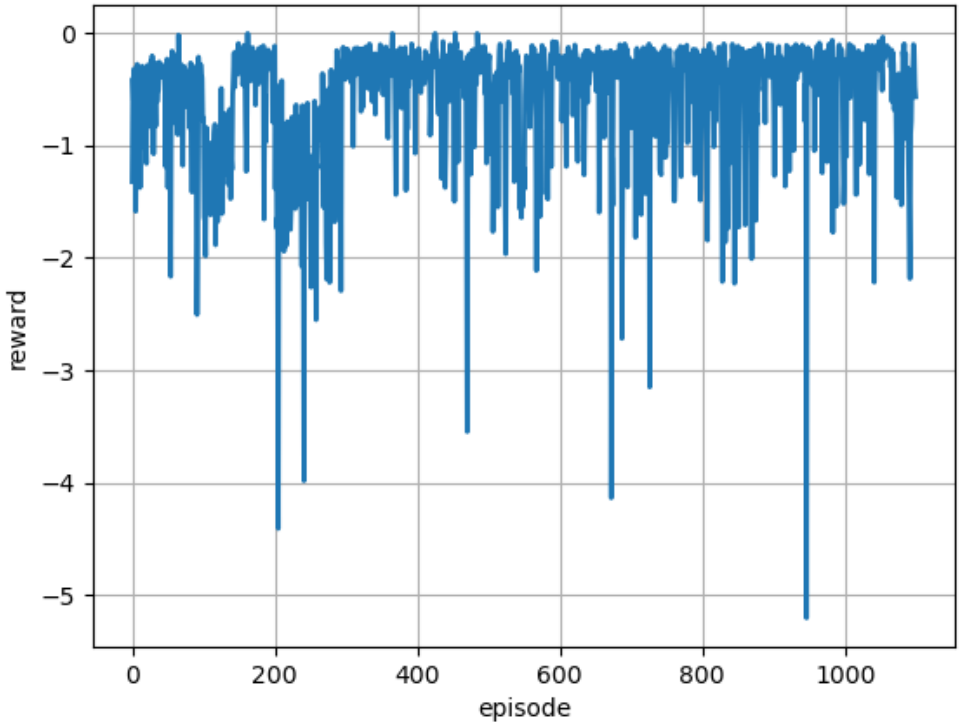

Figure 4.1: Comparative results for agent in 9x9 grid. Every 100 episodes different agent is playing. Each next agent is trained for 100 more episodes than the previous (i.e. 0, 100,...,1000)

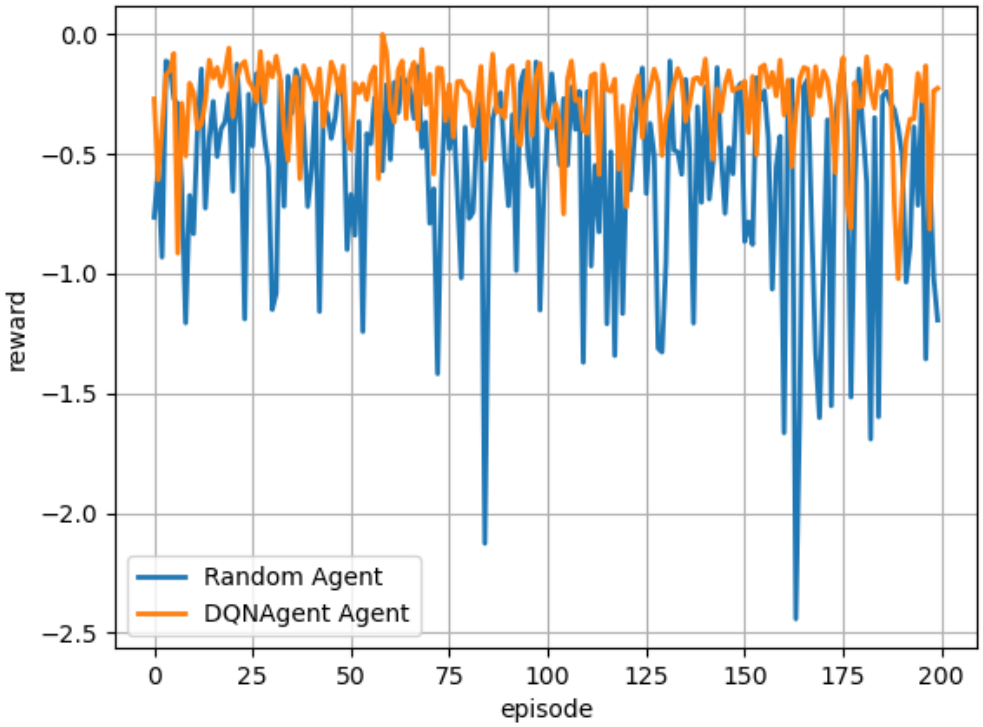

Figure 4.2: Random agent vs DQNAgent trained for 500 episodes. Difference in performance is small but promising.

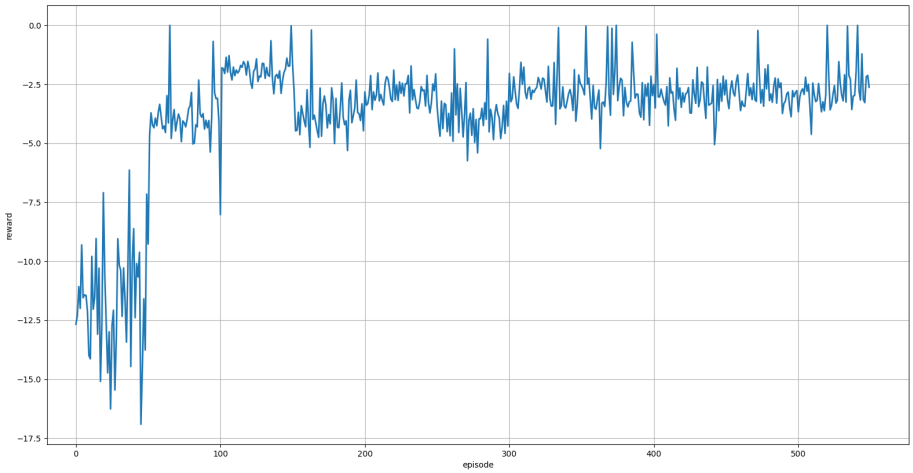

Figure 4.3: Comparative results for model 1 in 84x84 grid. Every 100 episodes different agent is playing. Each next agent is trained for 100 more episodes than the previous (i.e. 0, 100,...,1000)

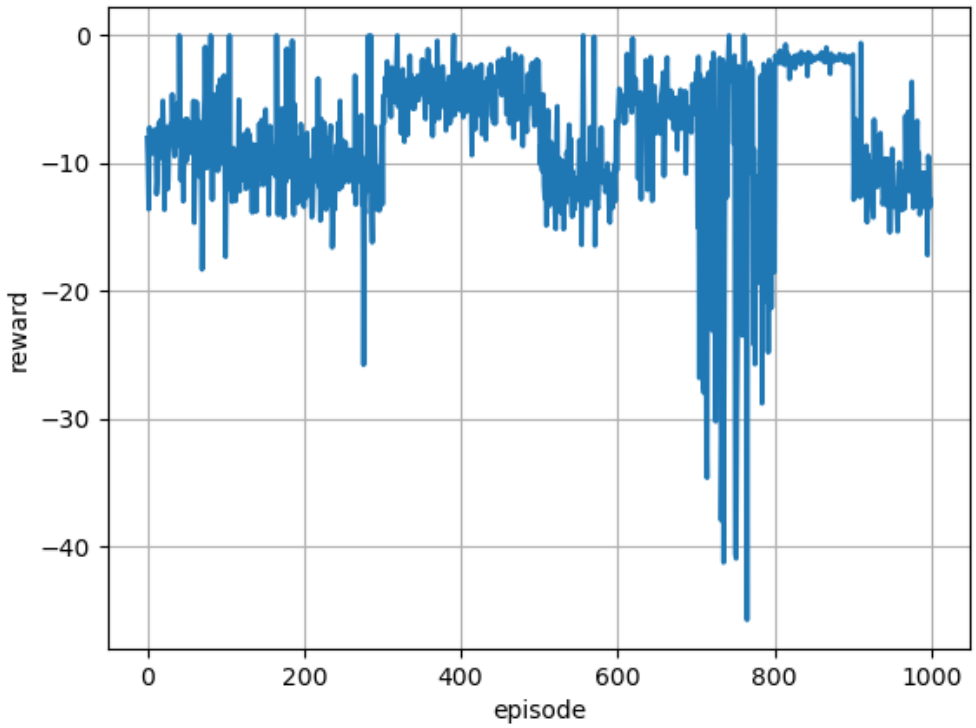

Figure 4.4: Comparative results for model 2 ver. 1 in 84x84 grid. Every 100 episodes different agent is playing. Each next agent is trained for 100 more episodes than the previous (i.e. 0, 100,...,1000)

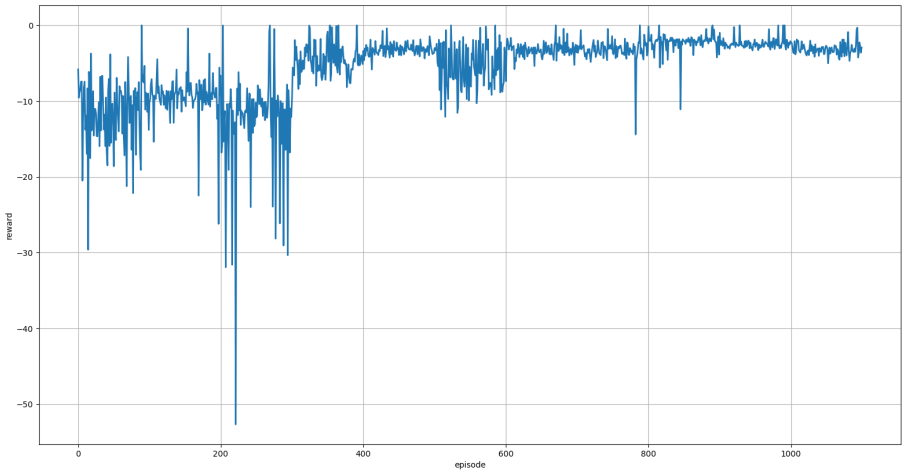

Figure 4.5: Comparative results for model 2 version 2 (trained every two steps) in 84x84 grid. Every 100 episodes different agent is playing. Each next agent is trained for 100 more episodes than the previous (i.e. 0, 100,...,1000)

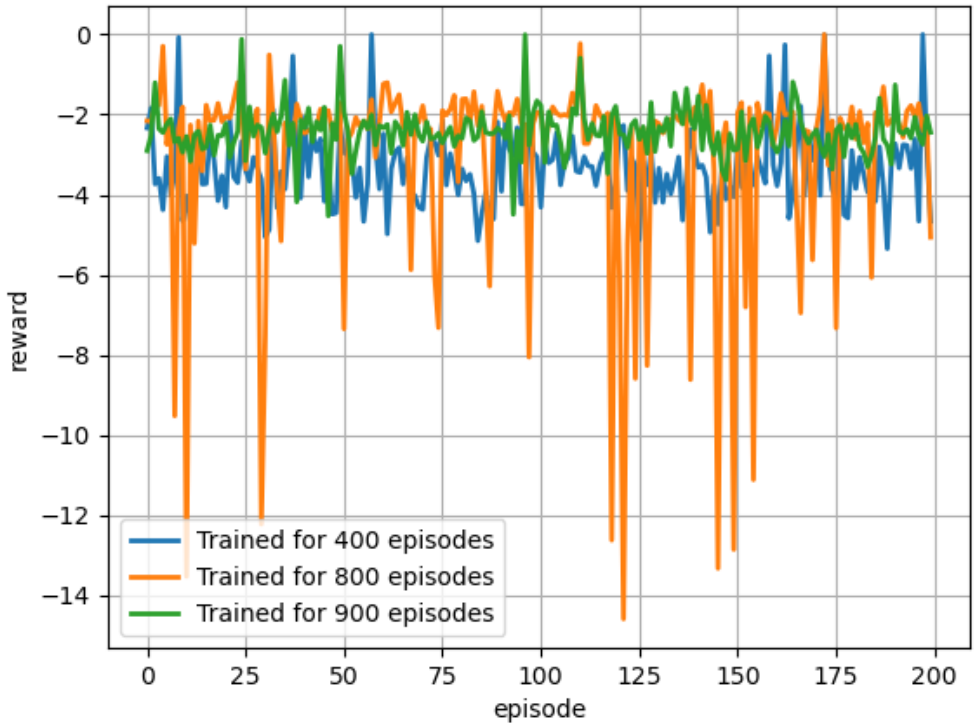

Figure 4.6: The three most promising models of model 2 ver. 2 were put under test. The one trained for 900 episodes shows the most steady performance.

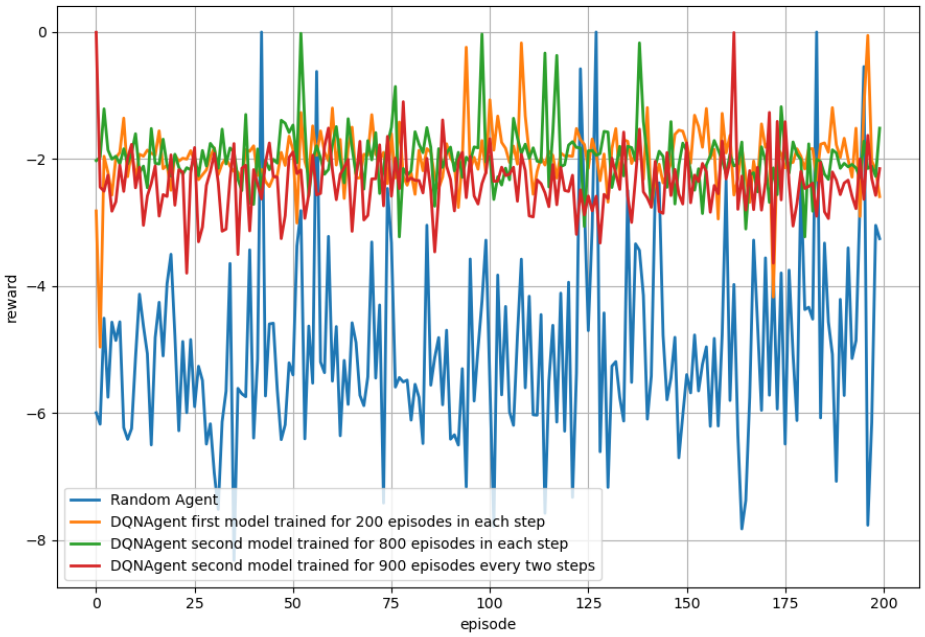

Figure 4.7: Comparative results Random agent vs DQN model 1 vs DQN model 2 v1 vs DQN model 2 v2

Chapter 5

Conclusions

This assignment was a great chance to study Deep Q-Network algorithm and enhance my knowledge in Reinforcement Learning. The results show that DQN is indeed working in gym-cellular-automata environment even in small grid sizes like 9x9 and 84x84.

5.1 Future work

It is interesting to experiment with different parameters during training (e.g. batch size, replay memory capacity) and different algorithms like Proximal Policy Optimization (PPO) and compare the results. As a step forward, a more complex environment can be tested e.g. 256x256, which is the default grid size the gym-cellular-automata uses for benchmarking.

Bibliography

[1] F. Berto and J. Tagliabue. Cellular Automata. In E. N. Zalta and U. Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2023 edition, 2023.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning, 2013.

[3] A. Plaat. *Deep Reinforcement Learning*. Springer Nature Singapore, 2022. ISBN 9789811906381. doi: 10.1007/978-981-19-0638-1. URL `http://dx.doi.org/10.1007/978-981-19-0638-1`.

[4] E. Soto, B. G. D. L. Paz, and López. Cellular automata environments for reinforcement learning. `https://github.com/elbecerrasoto/gym-cellular-automata`, 2023.

[5] R. S. Sutton and A. G. Barto. *Reinforcement Learning An Introduction*. MIT Press, second edition, 2018. ISBN 978-0-262-03924-6.

[6] C. Watkins. *Learning From Delayed Rewards*. PhD thesis, King's College, 1989.