EL2805 Reinforcement Learning - Computer Lab 2, 2022

Authors
Lea Keller - lmjke@kth.se - 19980209-4889
Jannik Wagner - wagne@kth.se - 19971213-1433

1 Problem 1: Deep Q-Networks (DQN)

2 a)

Done. We implementd an DQNAgent and a SimulationAgent to share the same interface.

2.1 b) - Why do we use a replay buffer and target network in DQN?

Replay Buffer
A replay buffer is used because otherwise successive updates would be correlated since they would be according to successive steps of a trajectory in the system. Furthermore, the replay buffer enables us to use mini batches in gradient descent. Both increases training stability.

Target network
A target network is used so that the target is fixed over a certain number of update steps. Without a fixed target, the target may change drastically leading to unstable training.

2.2 c)

Done. Check the submitted code.

2.3 d)

We implemented the modification Combined experience replay (CER).

The conducted experiments can be seen in table 1. h and l are the number of hidden units and number of hidden layers. The score is the total episode reward calculated over 50 episodes with confidence 0.95. DQN8 achieved a score of 163.2 and thus passed.

We used 2 hidden layers of 128 neurons each because it was the recommended maximum depth and width for a network and the training was still stable. The network should be more expressive than narrower or shallower networks.

A high value of γ seemed reasonable since high positive reward will only be given in the end of an episode. A recommended, we set $C = \frac{B}{N}$. Empirically, we found that the total episode reward decreased when training with $\epsilon_{min} < 0.3$. Thus, we set it to that value. Linear decay also outperformed exponential decay in our experiments.

2.4 e)

1. The plots of the total episodic reward and the total number of steps of DQN8 can be seen in figure 2.

Name	γ	L	T_E	C	N	ϵ_{max}	ϵ_{min}	ϵ-decay	α	h	l	score
DQN8	0.99	10,000	200	1,250	8	0.99	0.3	linear	0.0002	128	2	163.2 +/- 19.0
DQN9	1	10,000	200	1,250	8	0.99	0.3	linear	0.0002	128	2	97.4 +/- 30.4
DQN10	0.1	10,000	200	1,250	8	0.99	0.3	linear	0.0002	128	2	-108.1 +/- 49.7
DQN21	0.99	10,000	150	1,250	8	0.99	0.3	linear	0.0002	128	2	-66.8 +/- 19.8
DQN22	0.99	10,000	300	1,250	8	0.99	0.3	linear	0.0002	128	2	79.0 +/- 30.2
DQN24	0.99	5,000	200	625	8	0.99	0.3	linear	0.0002	128	2	-16.5 +/- 5.8
DQN25	0.99	20,000	200	2,500	8	0.99	0.3	linear	0.0002	128	2	-41.7 +/- 5.7
DQN19	0.99	10,000	200	156	64	0.99	0.3	linear	0.0002	128	2	209.0 +/- 21.6

Figure 1: Experiments

2. γ

   The experiment with $\gamma_1 = 1$ is DQN9. The plot can be ween in figure 3. The total episode reward is 97.4 +/- 30.4 and thus still passing but not as good as with $\gamma = 0.99$ in DQN8.

   Experiment DQN10 was conducted with $\gamma_2 = 0.1$. The plot can be seen in 4. The total episode reward is considerably lower with -108.1 +/- 49.7.

   The training process with $\gamma_1 = 1$ seems to be similar to the one with $\gamma_0 = 0.99$ reward-wise. The training process with $\gamma_2 = 0.1$ has considerably slower increase in the total episode reward and ends with much lower rewards and episode step counts.

3. T_E

   Experiment DQN21 was conducted with 150 episodes and achieved a total episode reward of -66.8 +/- 19.8. The plot can be seen in 5.

   Experiment DQN22 was conducted with 300 episodes and achieved a total reward of 79.0 +/- 30.2. The plot can be seen in 6.

   Both experiments performed worse than DQN8. DQN21 considerably worse.

   B

   Note that we also scale C by the same factor as B since it is recommended that $C = \frac{B}{N}$.

   Experiment DQN24 was conducted with a memory buffer size of 5000 and achieved a total episode reward of -16.5 +/- 5.8. The plot can be seen in 7.

   Experiment DQN25 was conducted with a memory buffer size of 20000 and achieved a total reward of -41.7 +/- 5.7. The plot can be seen in 8.

   Both experiments performed considerably worse than DQN8.

In experiment DQN19 we increased the batch size to 64, which let to achieving a higher total episodic reward of 209.0 +/- 21.6. The plot can be seen in figure 9.

2.5 f)

1. The 3D plot of the value function can be seen in figures 10, 11, and 12 from different perspective. As can be seen, the values are lowest with ω close to 0. This is contrary to what we would have expected since it should be easiest to successfully land with the correct angle. Furthermore, with $\omega = 0$, both with y close to 0 and 1.5, the values are slightly higher than with y in between those. We would have expected the values to be highest with y close to 0 since then it should be easiest to successfully land.

2. The policy can be seen can be seen in figure 13. The policy makes sense. In order not to crash, the agent should seek to have $\omega = 0$. Thus, if it is rotated to the left ($\omega < 0$), it should rotate to the right by firing the left engine. Similarly if it is rotated to the right ($\omega > 0$), it should rotate to the left by firing the right engine.

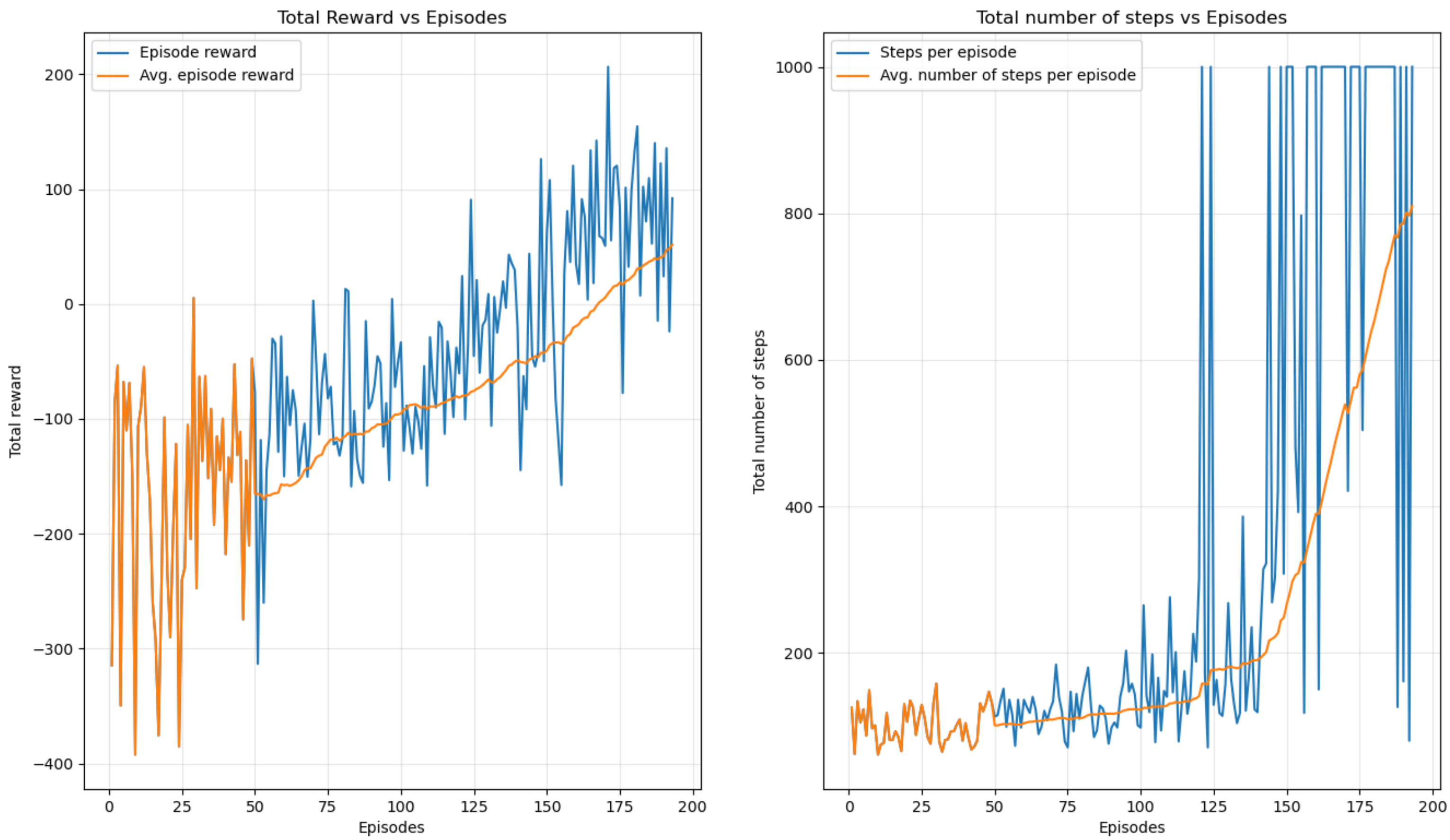

Figure 2: Reward and steps of DQN8

3 g)

The Policy given by DQN8 achieves an average total reward of 163.2 +/- 19.0 with confidence 0.95 over 50 episodes. The random policy achieves an average total reward of -161.9 +/- 24.5 with confidence 0.95 over 50 episodes.

As expected, the DQN8 agent performs considerably better.

4 h)

Done.

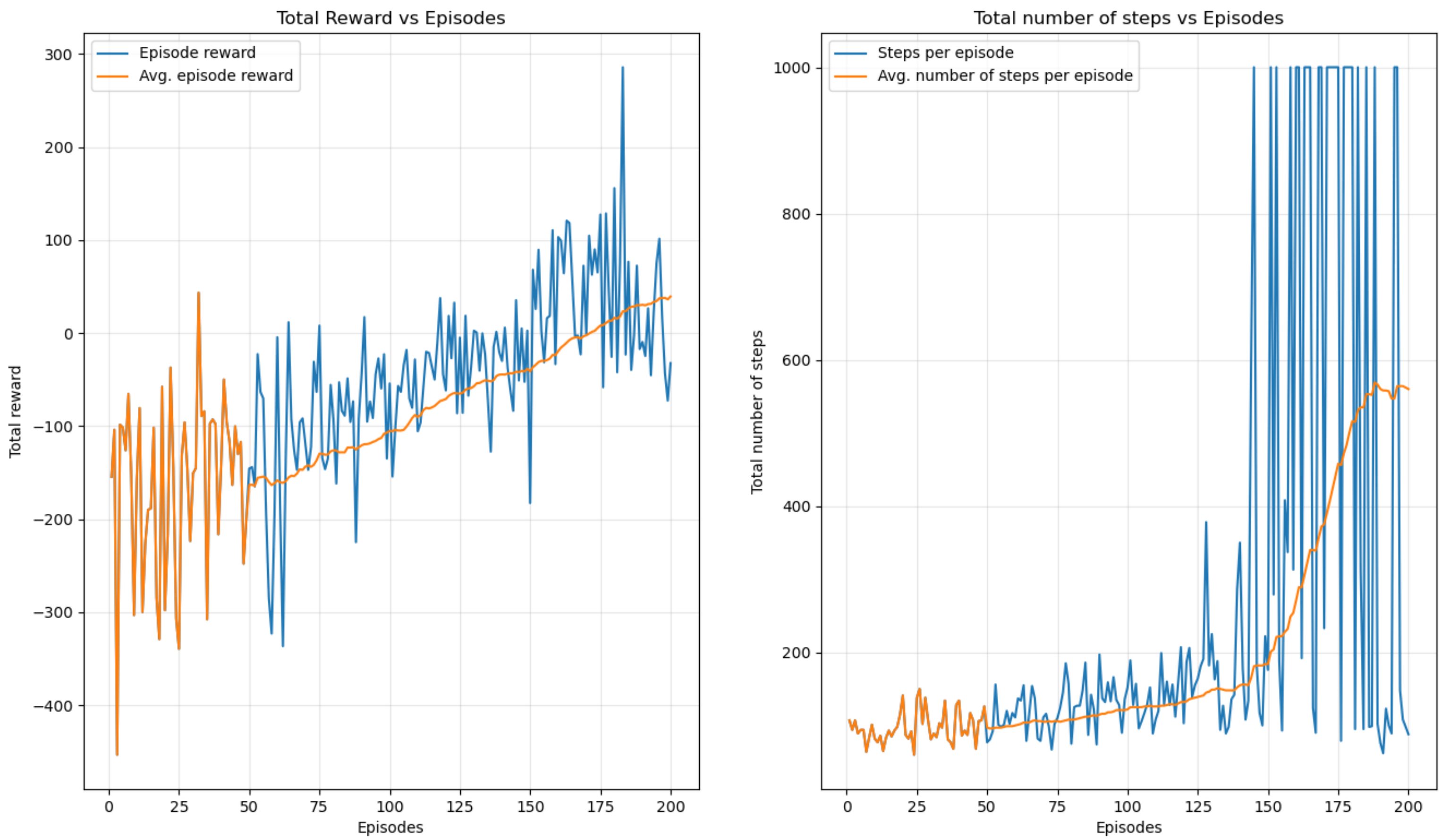

Figure 3: Reward and steps of DQN9

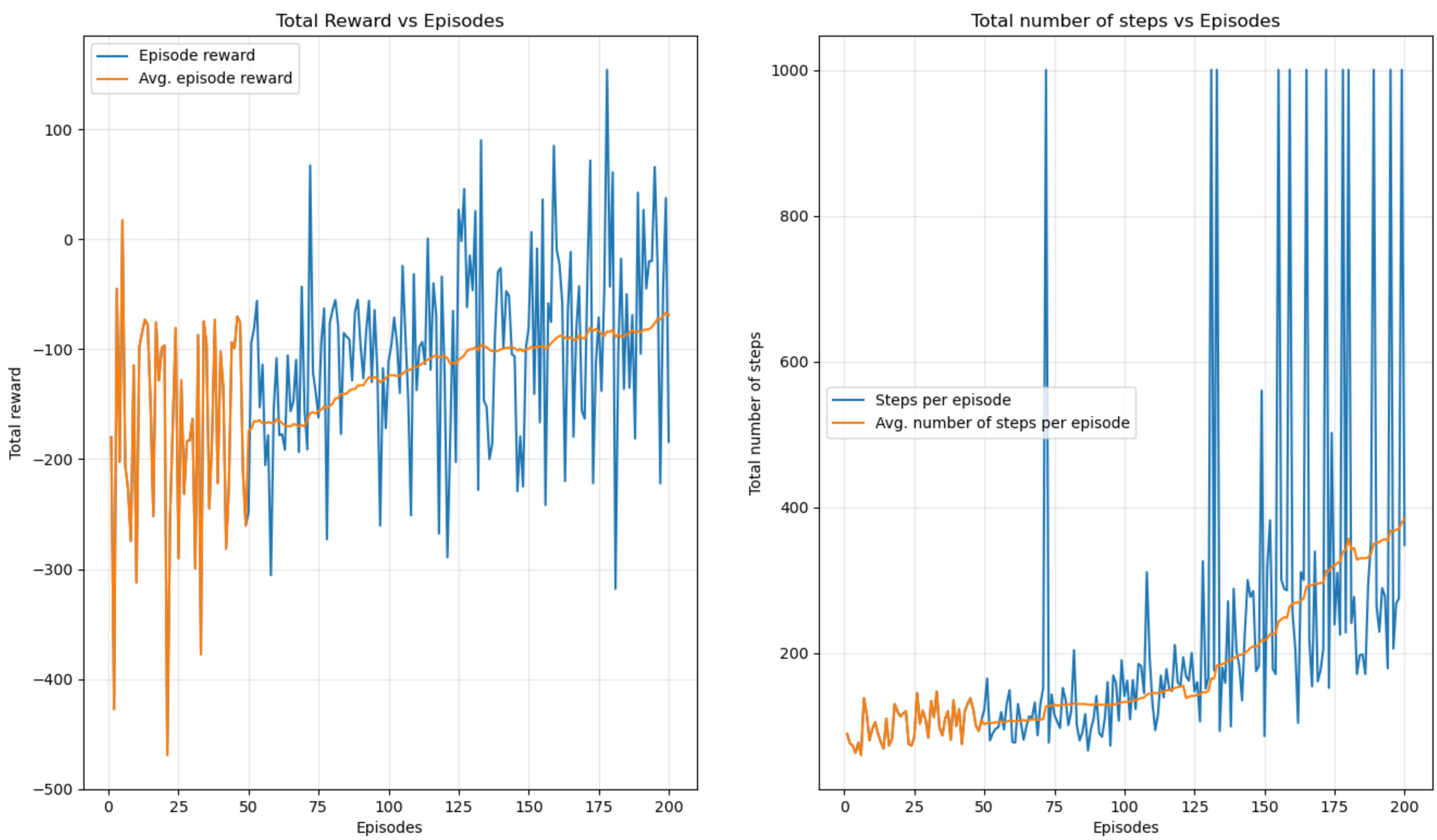

Figure 4: Reward and steps of DQN10

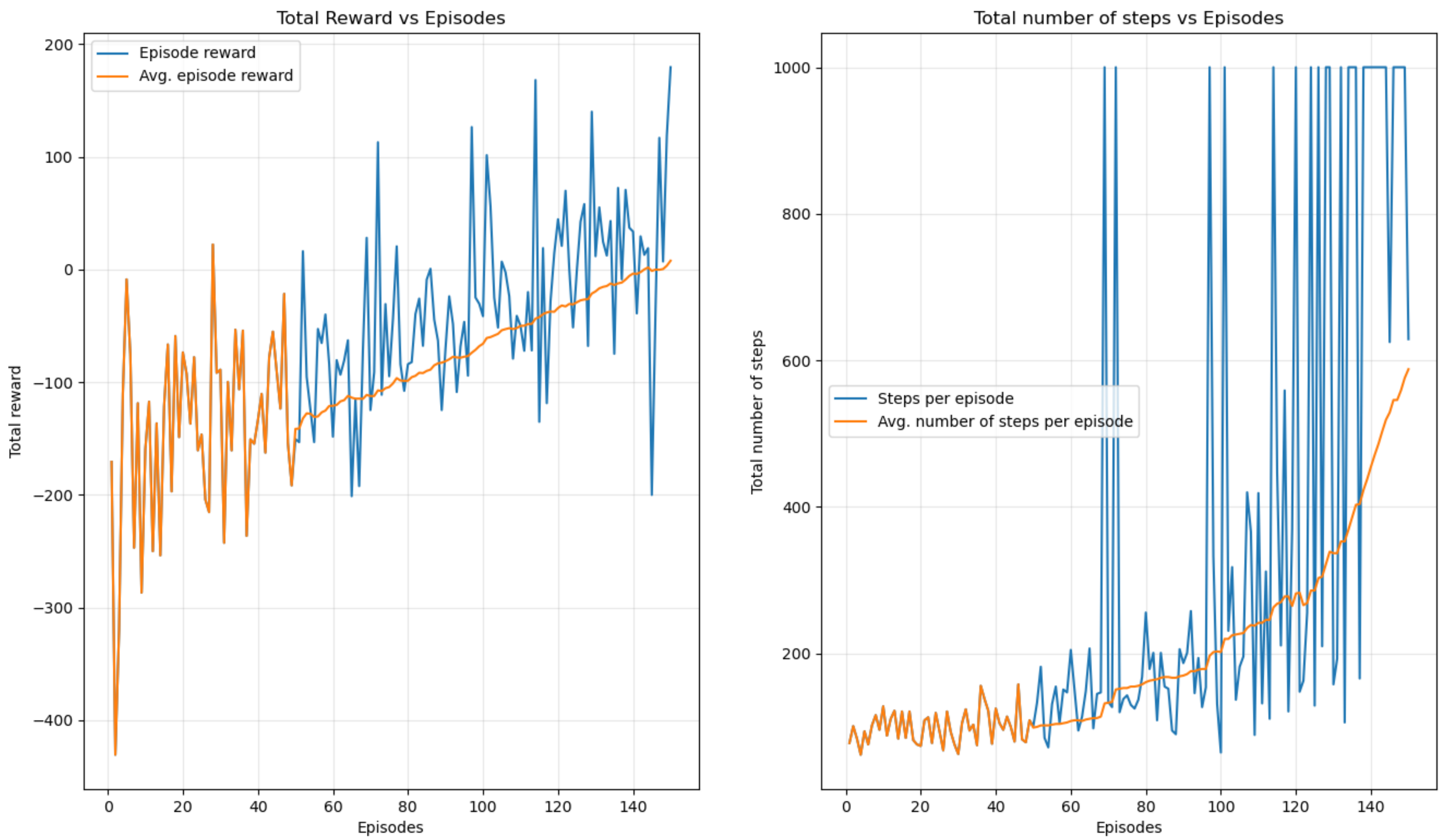

Figure 5: Reward and steps of DQN21

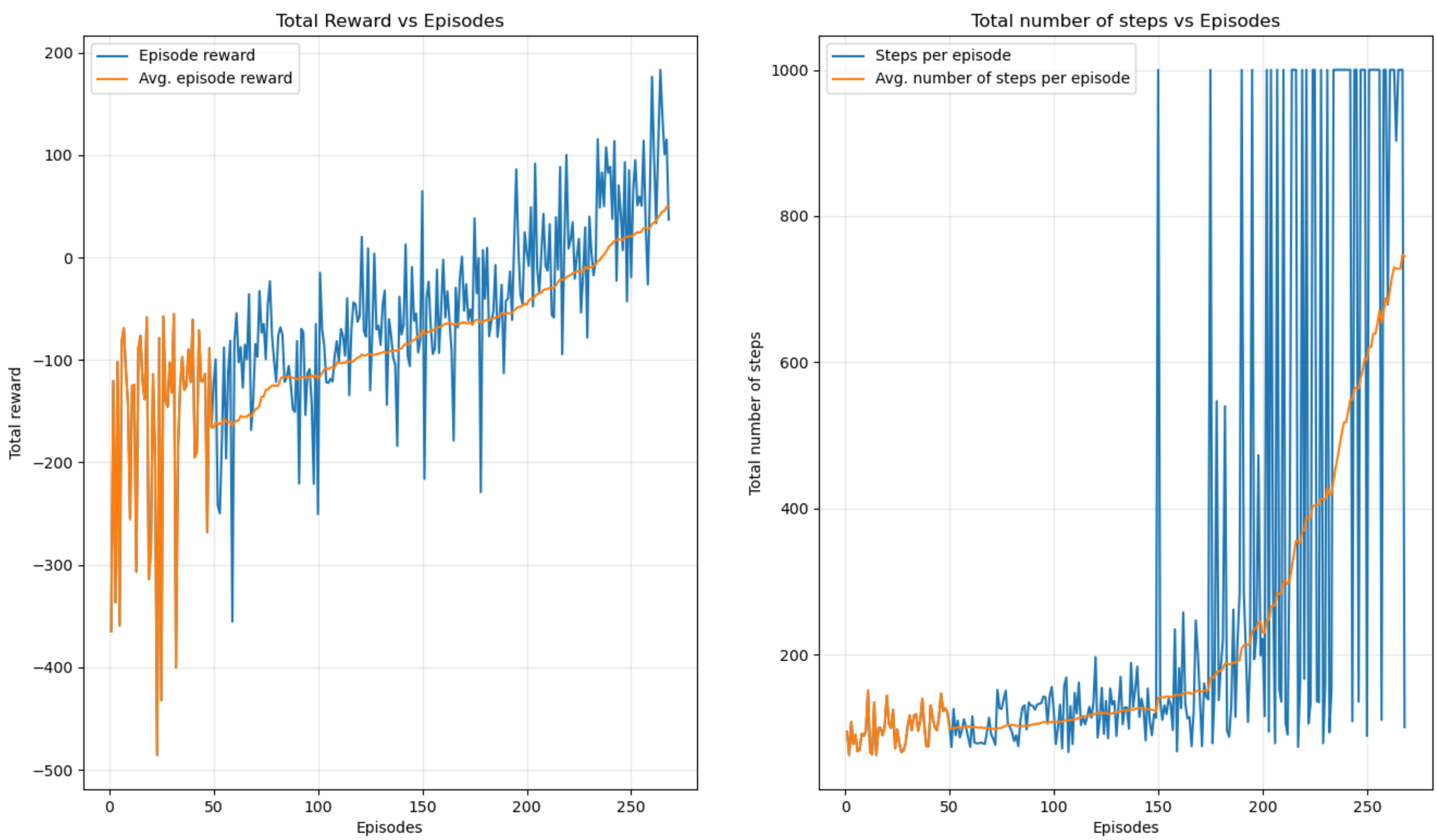

Figure 6: Reward and steps of DQN22

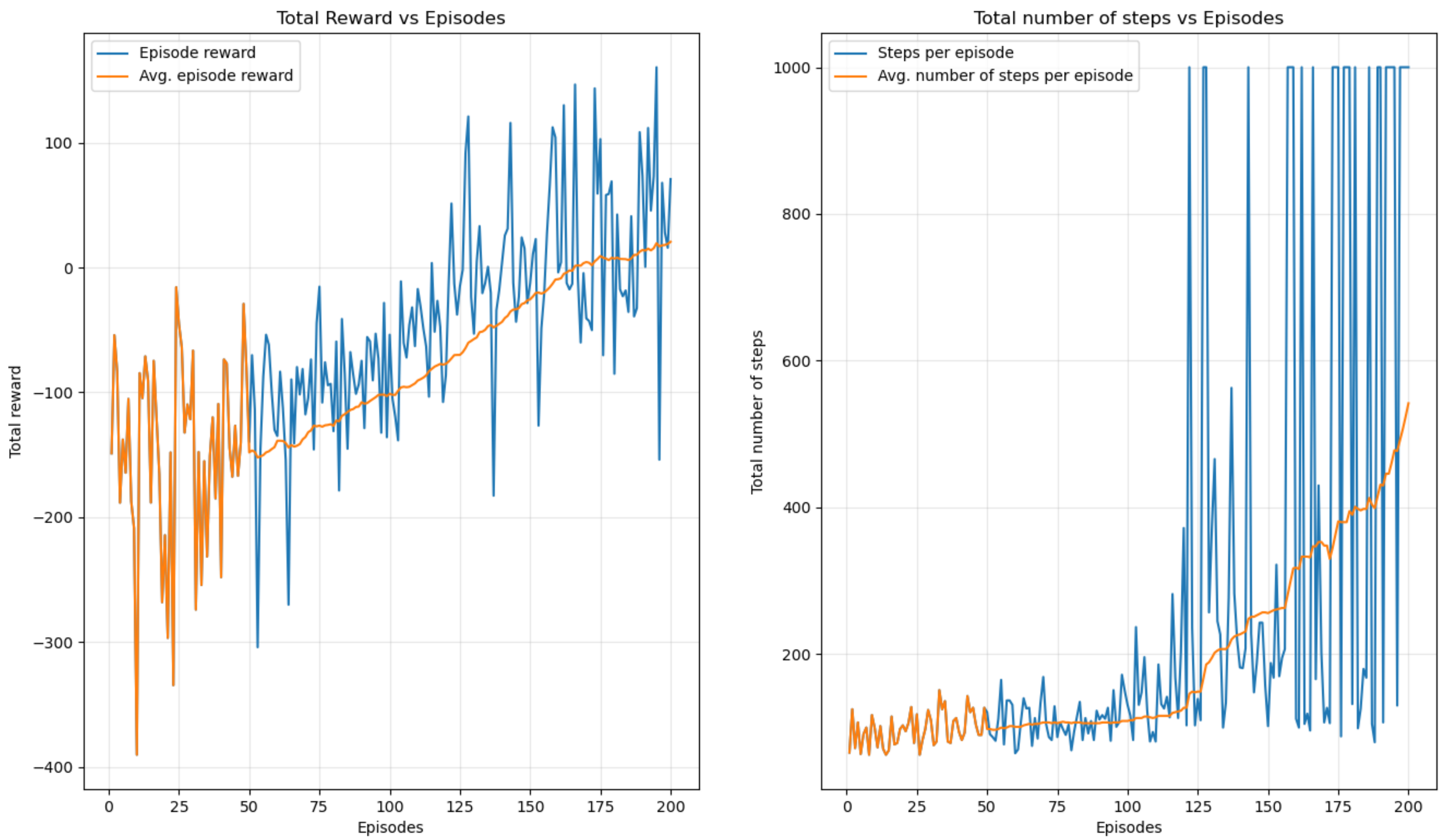

Figure 7: Reward and steps of DQN24

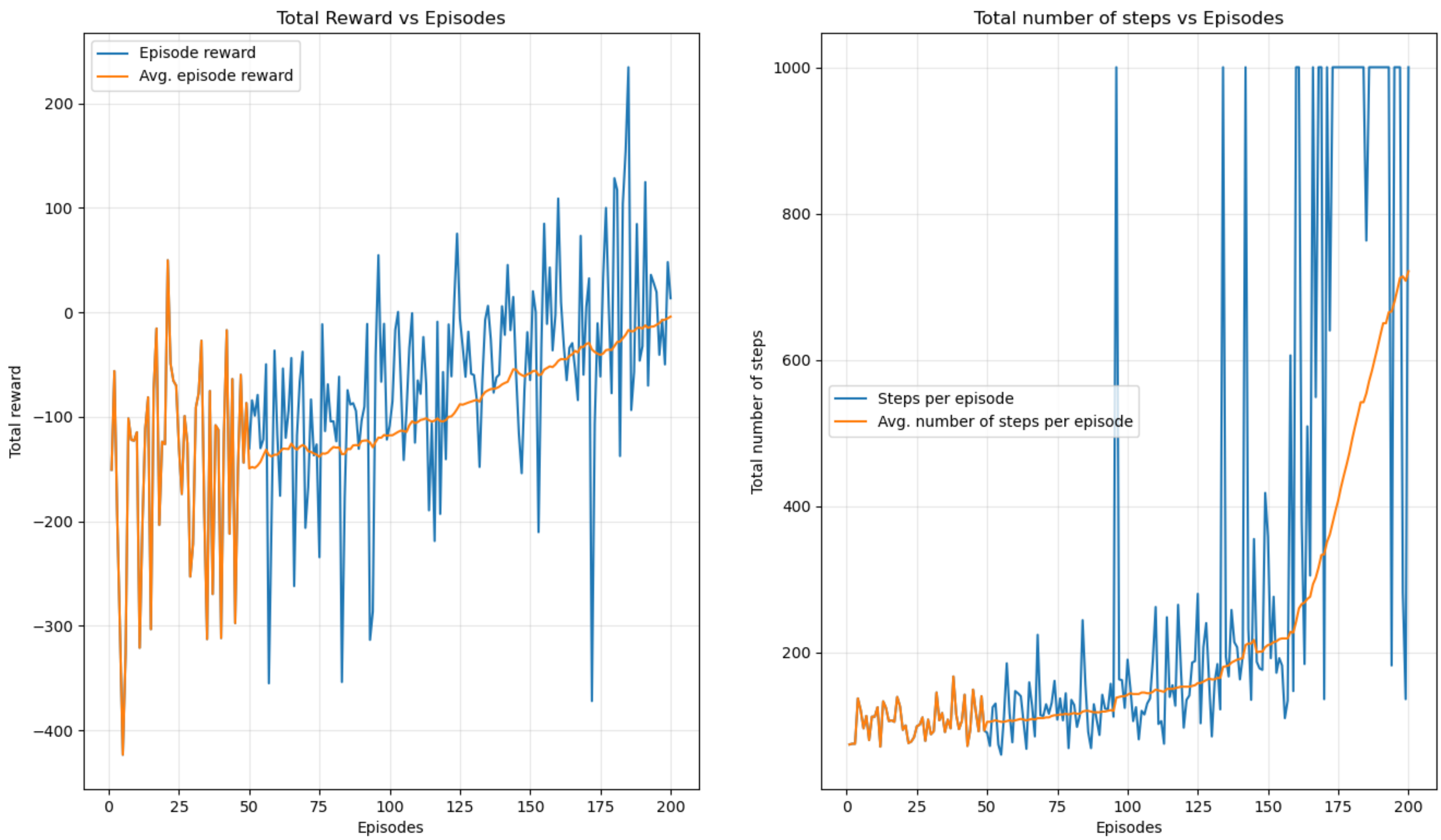

Figure 8: Reward and steps of DQN25

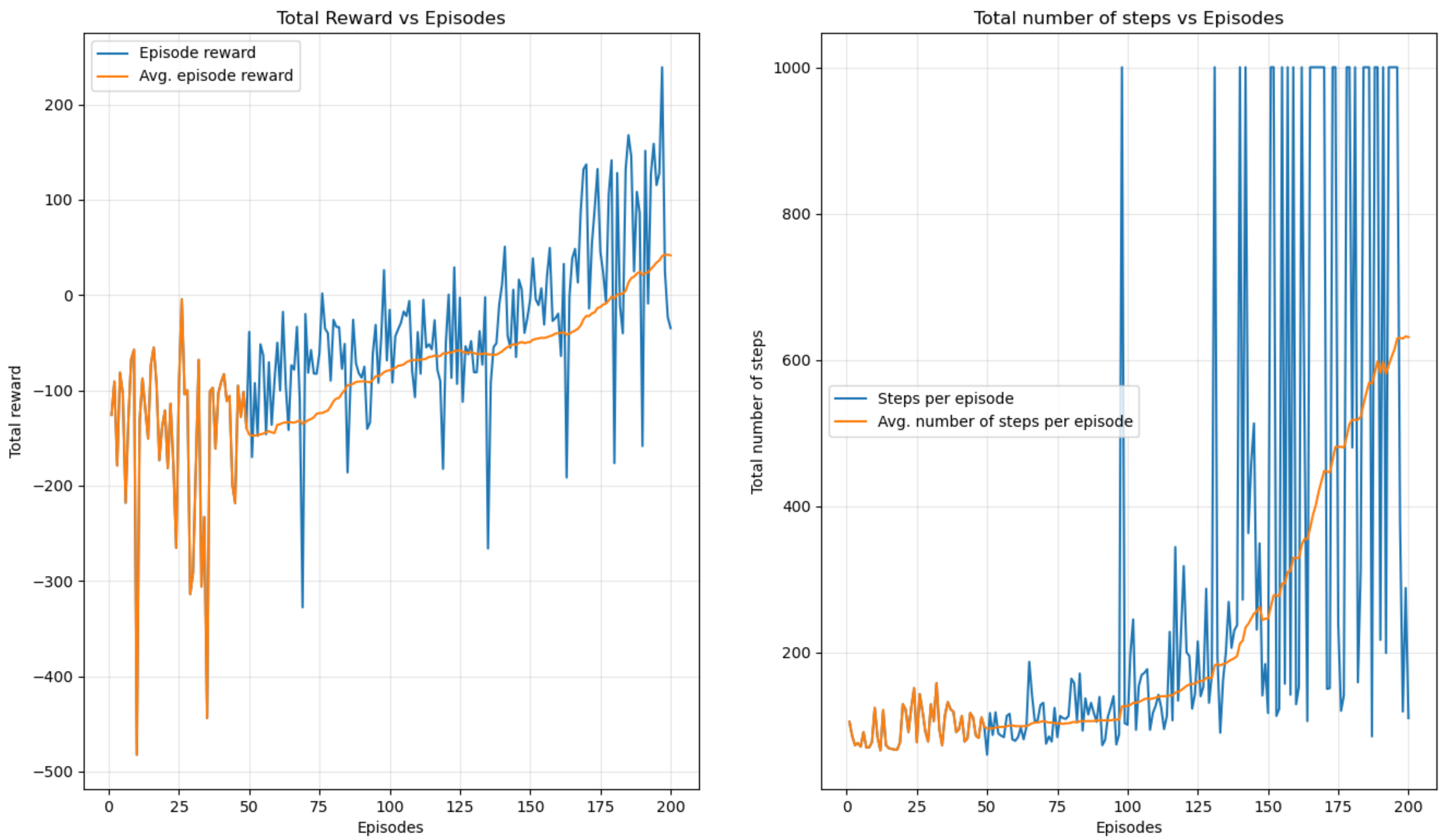

Figure 9: Reward and steps of DQN19

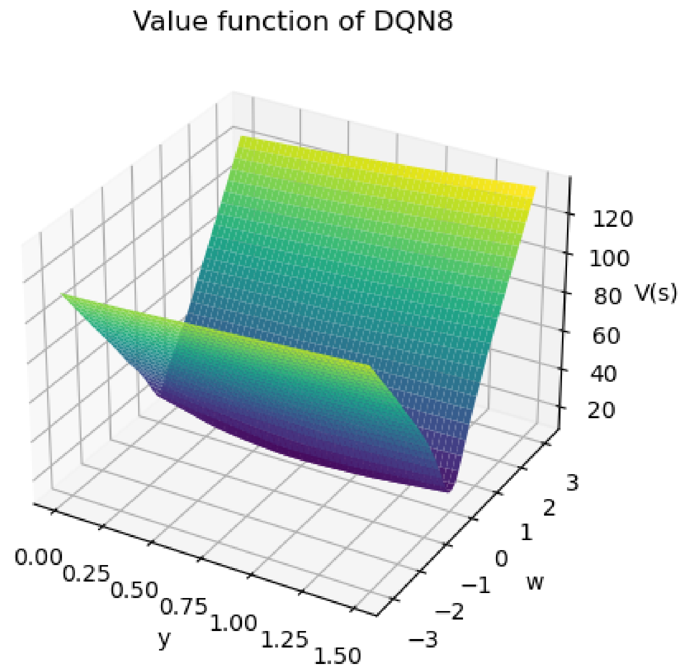

Figure 10: Value function of DQN8

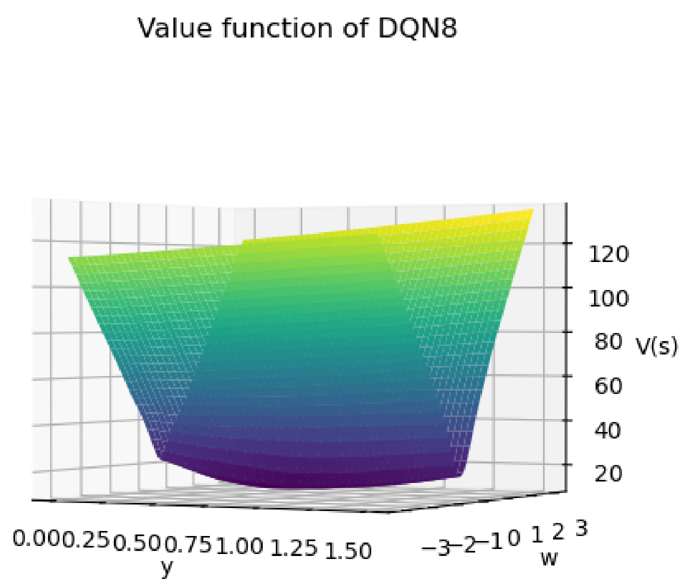

Figure 11: Value function of DQN8

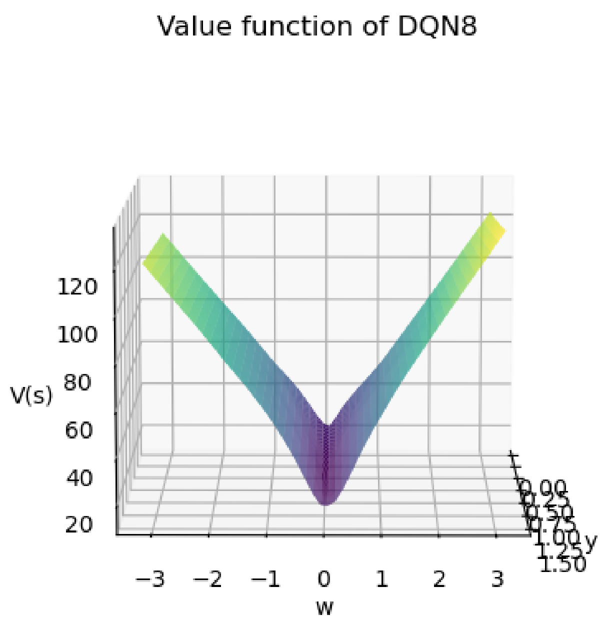

Figure 12: Value function of DQN8

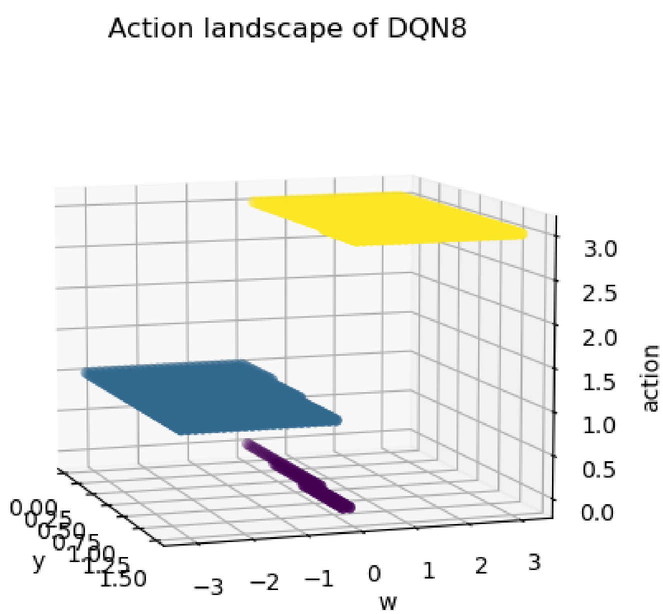

Figure 13: Policy given by DQN8