Reinforcement Learning EL2805
Laboration 2

Ilian Corenliussen, 950418-2438, ilianc@kth.se
Daniel Hirsch, 960202-5737, dhirsch@kth.se

December 2020

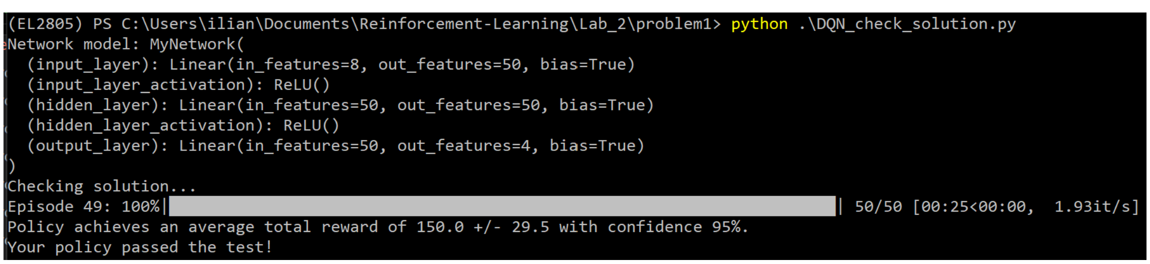
Problem 1:
Deep Q-Networks (DQN)

B)

In a DQN a replay buffer is used to stabilize the learning process. By mixing previous with new experience to make the network train on a mixed experiences rather then only consider new experience. This is done before the update of the neural network, to decorrelation the training samples for each batch. The target network is used to have a stable target values too compare to, instead of updating after every step we update at every C-step and thus we have a stable copy of the network(target network) too compare to.

C)

Requirements fulfilled as can be seen in Figure 1, with an total reward of 150.0 ± 29.5 with the confidence of 95%.



```
(EL2805) PS C:\Users\ilian\Documents\Reinforcement-Learning\Lab_2\problem1> python .\DQN_check_solution.py
Network model: MyNetwork(
  (input_layer): Linear(in_features=8, out_features=50, bias=True)
  (input_layer_activation): ReLU()
  (hidden_layer): Linear(in_features=50, out_features=50, bias=True)
  (hidden_layer_activation): ReLU()
  (output_layer): Linear(in_features=50, out_features=4, bias=True)
)
Checking solution...
Episode 49: 100%|████████████████████████████████████████| 50/50 [00:25<00:00,  1.93it/s]
Policy achieves an average total reward of 150.0 +/- 29.5 with confidence 95%.
Your policy passed the test!
```

Figure 1. Average total reward over 50 points on average over 50 episodes - test passed.

D)

We chose to implement a network with one hidden layer, the input layer uses 8 neurons to match the 8-dimensional state of the problem. The hidden layer had the size of 50 neurons. The optimizer chosen was the Adam optimizer as it was suggested for these types of problems. The clipping value was set to 1.

The size of the experience replay buffer parameter L was chosen to 20000 after iterating through values between $5000 - 30000$. The parameter N, i.e. the size of the training batch were chosen to 24 to match the complexity of our problem, as it generally is in the order of $4 - 128$. The update frequency of the target network, C, were chosen to be $C = L/N$ as it is the general suggestion for the update frequency. The number of episodes T_E where iterated between 100 to 1000, and the chosen value were 600. The number of episodes were chosen to 600 because no significant performance increase where found for more episodes, also when training the neural network to long it starts forgetting. ϵ were chosen to exponentially decay between the values of 0.99 and 0.05. The discount factor γ was set to 0.99.

E)

1)

As can be seen in Figure 2 the total reward goes up for the training and reaches a peak at 600 episodes. The number of steps does also steady decrease after 200 episodes. Where the hyperparameters where selected as described under Section D.

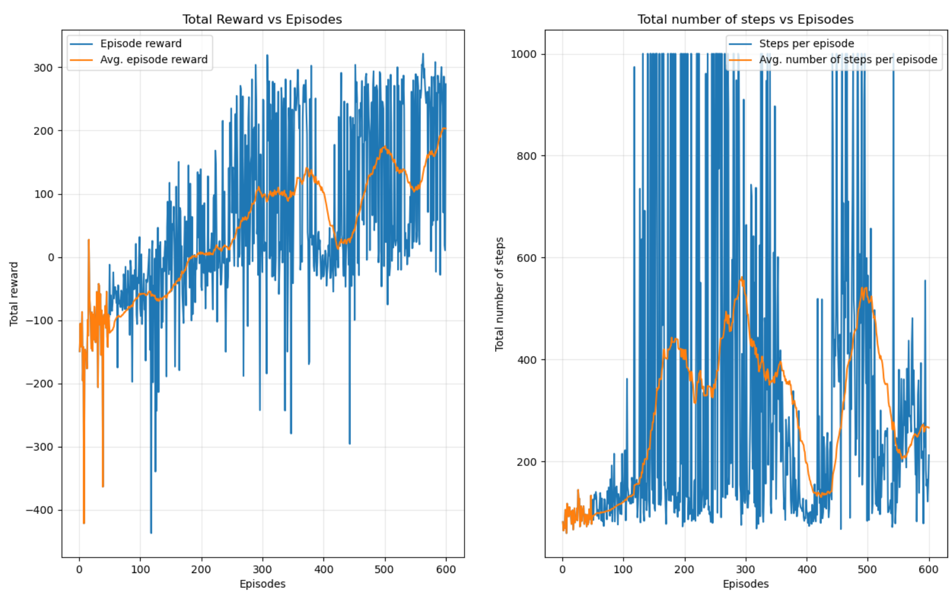


Figure 2. Total episodic reward and the number of steps taken per episodes during training for the final optimized policy.

2)

When using γ_1 the lander mostly took the approach of hovering, due to infinity sum of rewards from far in the future see Figure 3. A too low gamma, such as γ_2, the lander took a too aggressive approach of descent which resulted in a less satisfying reward score see Figure 4.

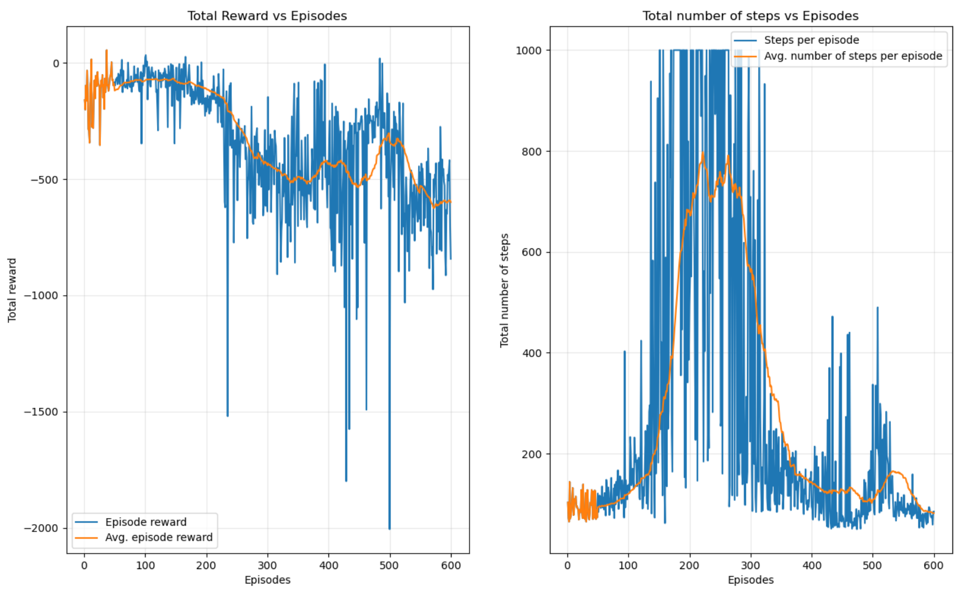


Figure 3. Total episodic reward and the number of steps taken per episodes during training with γ_1.

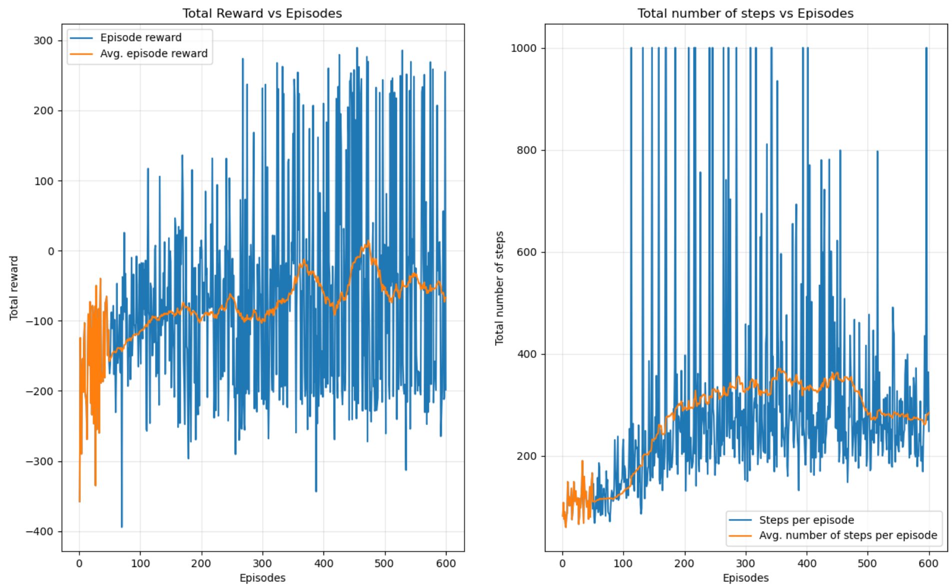


Figure 4. Total episodic reward and the number of steps taken per episodes during training with γ_2.

3)

When using 1000 episodes the average reward does not decrease, i.e. no catastrophic memory loss, but it does not have any significant improvement either see Figure 5. When increasing the buffer size to 5000, a decrease in the average reward is obtained, see Figure 6. The training process also seems to be more stable, at least for the first 350 episodes. But it fluctuates more in its behaviours due to it more frequently fills its buffer with new experiences.

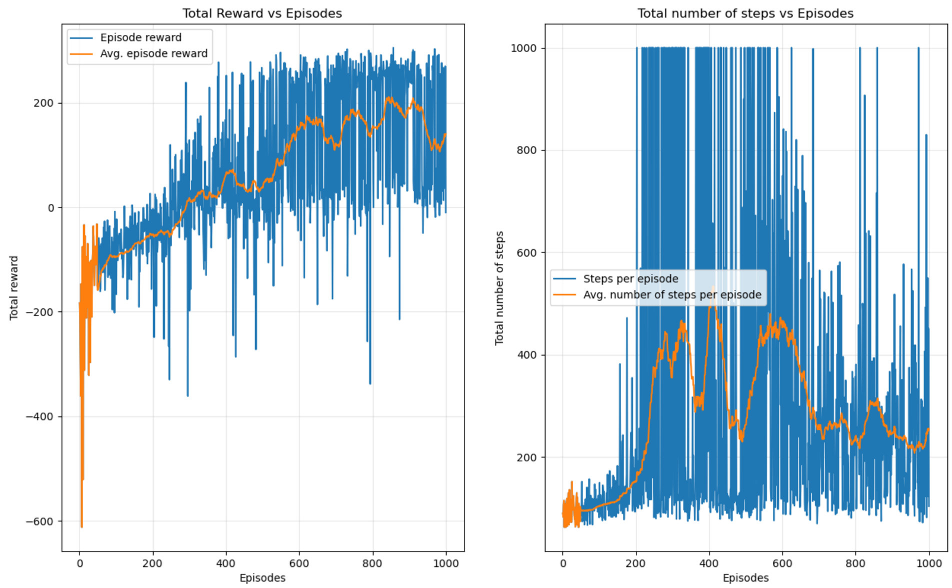


Figure 5. Total episodic reward and the number of steps taken per episodes during training, trained for 1000 episodes.

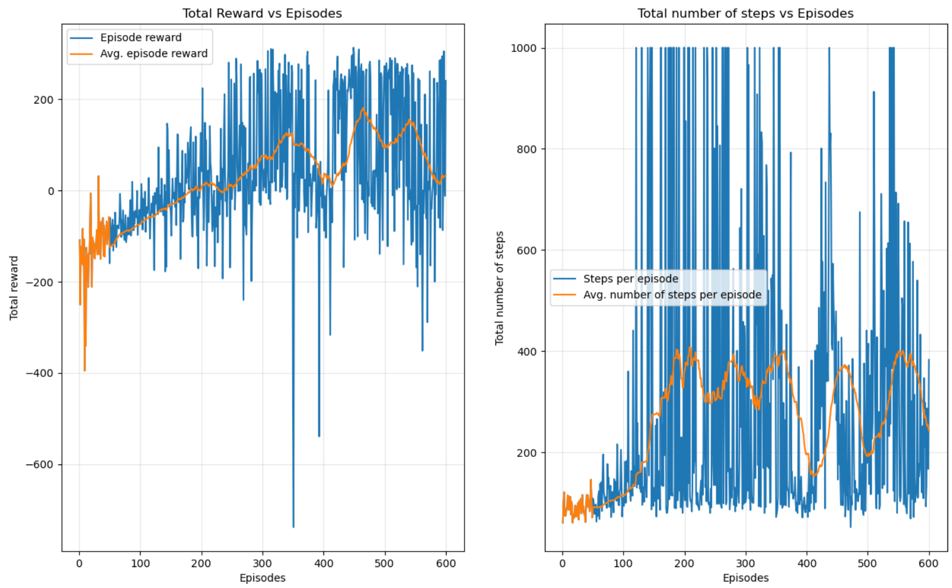


Figure 6. Total episodic reward and the number of steps taken per episodes during training, trained for 1000 episodes. With a buffer size of 5000.

F)

1)

The $\max_a Q_\theta(s(y, \omega), a)$ for the state $S(y, \omega) = (0, y, 0, 0, \omega, 0, 0, 0)$ with $\omega \in [-\pi, \pi]$ and $y \in [0, 1.5]$, see Figure 7, the highest values where proportional to the angel of the lander. Also there were a small incline depending on the height from the ground of the lander.

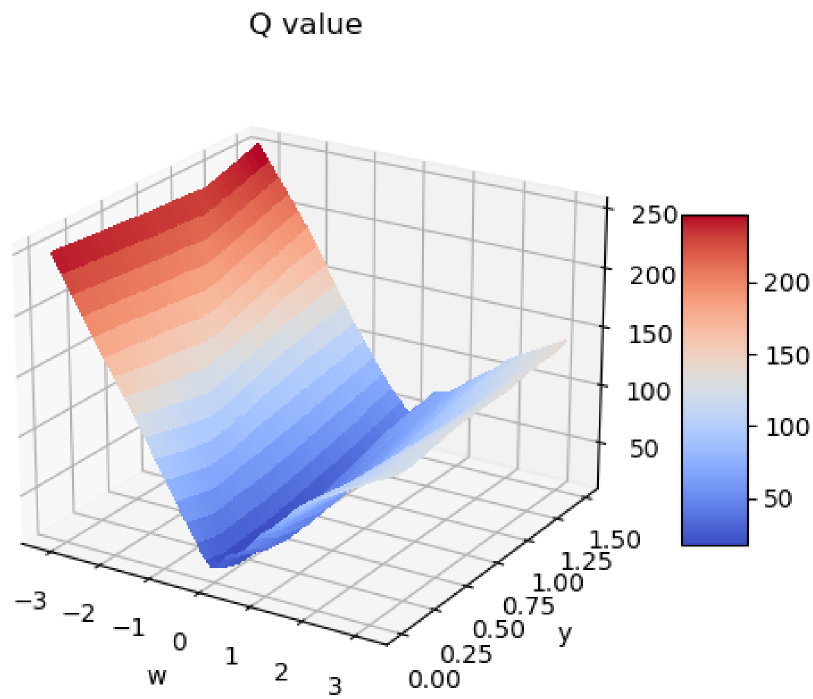


Figure 7. the Q values with $\omega \in [-\pi, \pi]$ and $y \in [0, 1.5]$.

2)

The $arg \max_a Q_\theta(s(y, \omega), a)$ for the state $S(y, \omega) = (0, y, 0, 0, \omega, 0, 0, 0)$ with $\omega \in [-\pi, \pi]$ and $y \in [0, 1.5]$, see Figure 8, one could notice that the lander compensates with the left and right motor depending on the angle of the lander. One interesting finding is that the lander does not activate the main motor in any of the states.

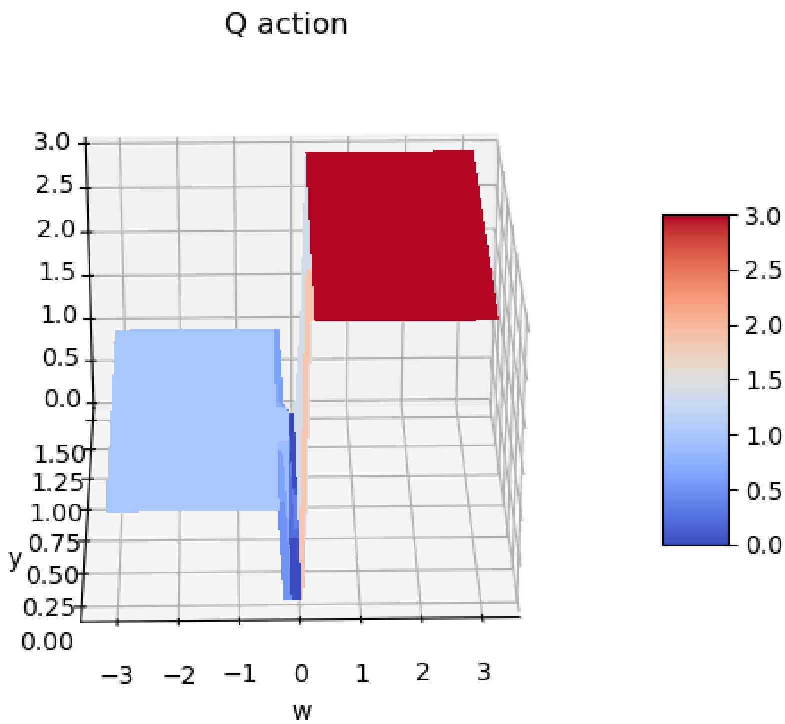


Figure 8. the Q actions with $\omega \in [-\pi, \pi]$ and $y \in [0, 1.5]$.

G)

If comparing Figure 2 to Figure 9, we see that the total average reward over 50 episodes for the RandomAgent is approximately -210 while our Q-network has a total average reward of 150. The comparison also shows that the RandomAgent does not learn and improve its reward during the whole period of episodes, which is expected.

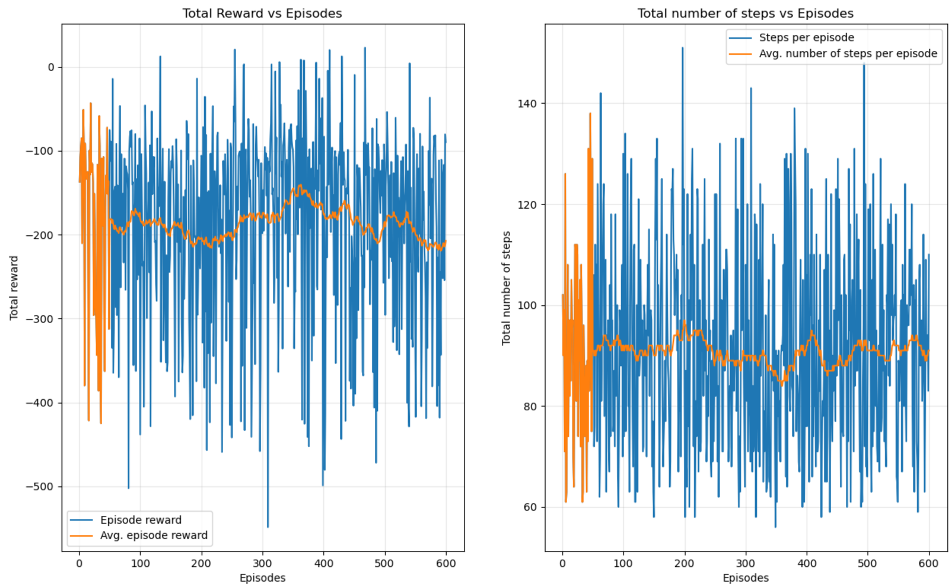


Figure 9. FTotal episodic reward and the number of steps taken per episodes during training for the RandomAgent.