Reinforcement Learning
Exercise 04

Finn Hülsbusch & Dennis Heinz

Task 1. DQN for Breakout

For the following DQN implementations, we used the hyperparameters in Table 1.

Hyperparameter	In Code	Value
Start Epsilon	start_epsilon	0.4
End Epsilon	end_epsilon	0.01
Exploration Fraction	exploration_fraction	0.1
Nr. Episodes	nr_episodes	20000
Maximum t	max_t	4000
Gamma	gamma	0.99
Replay Buffer Size	replay_buffer_size	100000
Warm Start Steps	warm_start_steps	500
Sync. Rate	sync_rate	128
Optimizer	optimizer	Adam
Optimizer learning rate	optimizer.lr	0.0000625
Optimizer Epsilon	optimizer.eps	0.00015

Table 1: Used Hyperparameters

a) **Plain DQN**

The DQN was run twice with identical hyperparameters and seeds to understand the problem. It became clear that, despite seedings, different behaviors were observed. For the first one million steps, fast learning can be observed. After that, rewards and episode length increase only slowly. This can be seen in Figures 1 and 2.

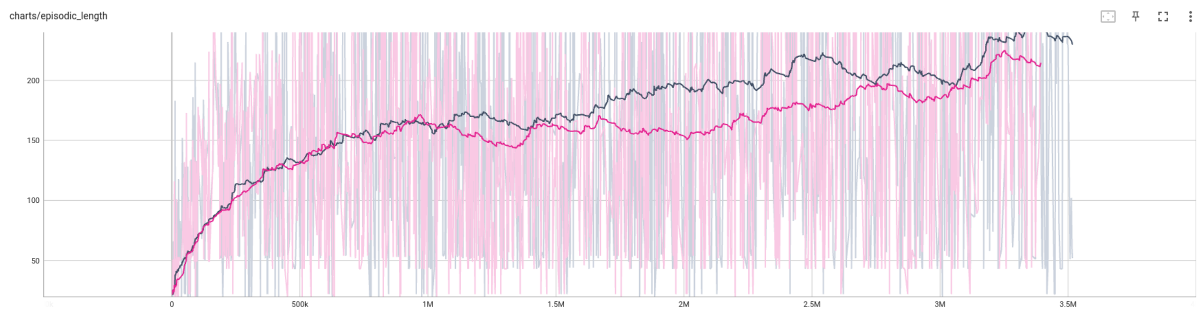

Figure 1: Episode Length

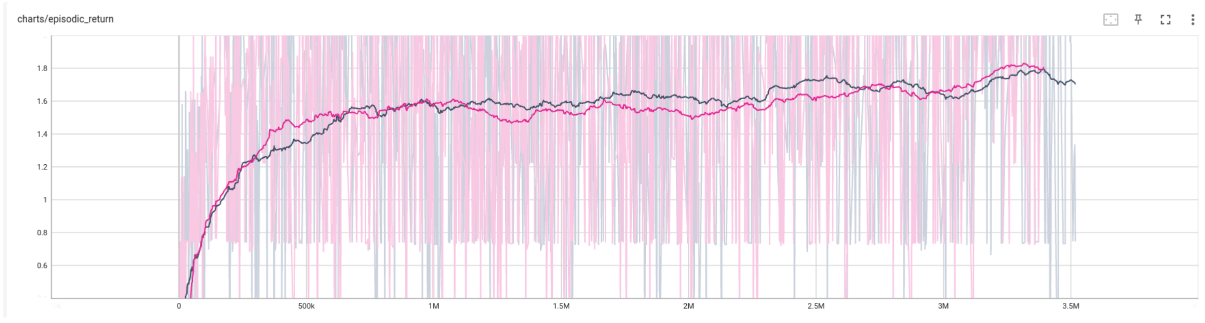

Figure 2: Episode Return

b) Double DQN

In Double DQN, the target network was used to determine the Q-values of the next_state, using the action selected by the learning_network. No significant difference can be seen in the graphs with respect to episode length and return. The same parameters were used for the baseline. Noticeable is a strong drop directly at the beginning of the training. This can be seen in Figures 3 and 4.

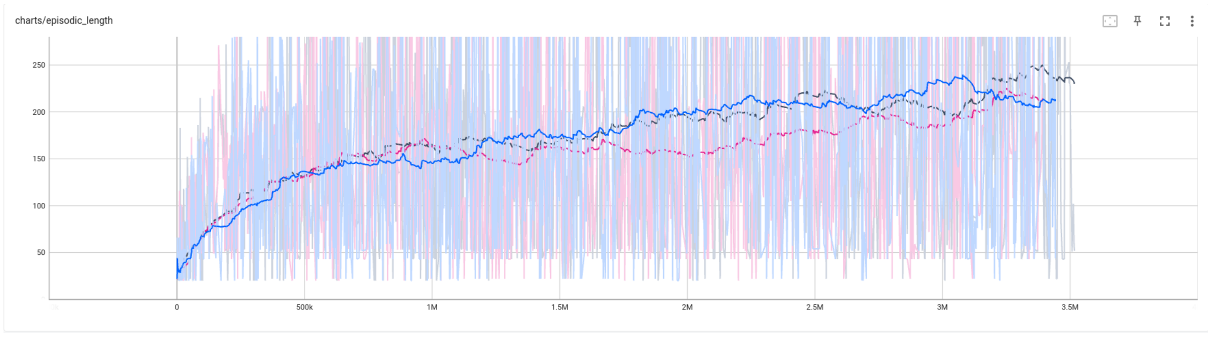

Figure 3: Episode Length

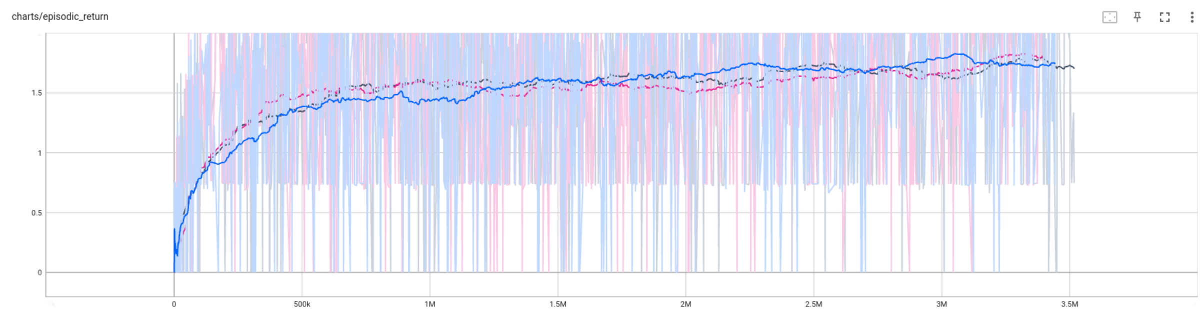

Figure 4: Episode Return

A clear difference can be seen for the q_values and the loss over time (here only a baseline is compared with the double since a baseline was collected before the Tensorboard was adjusted). The double DQN overestimates the q values within the first 250,000 steps and then corrects them slowly. This can also be seen in the loss (Figure 5).

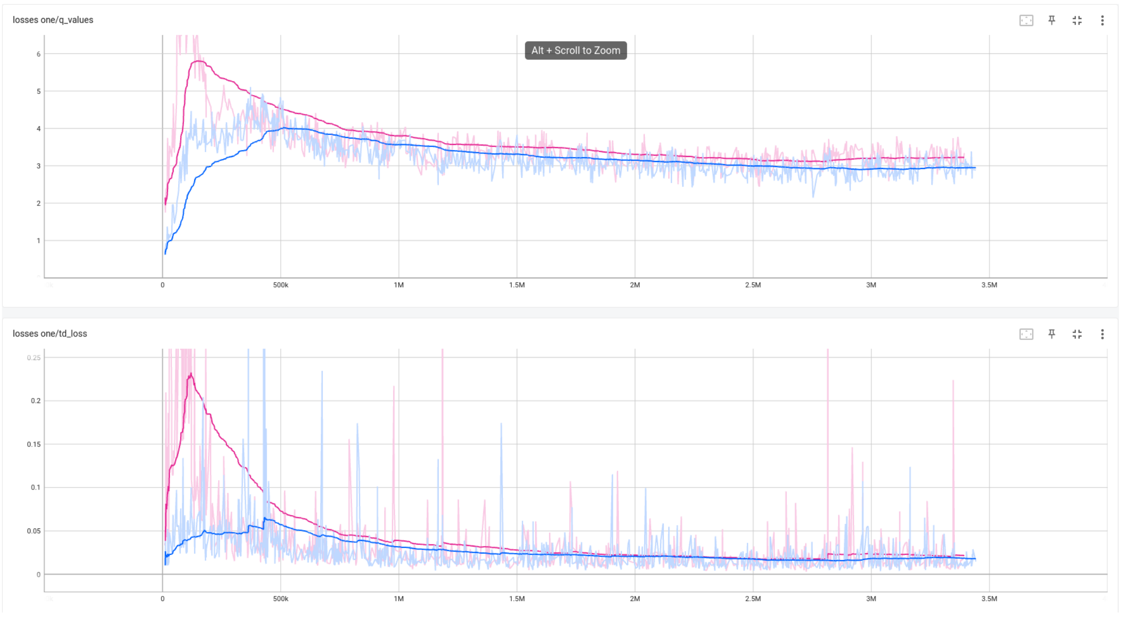

Figure 5: Loss

c) Dueling DQN

For Dueling DQN, only the artificial neural network was customized. The consideration of the two streams takes place directly in the architecture and forward method. Overall, more steps could be achieved at the same number of episodes, since the episodes were longer than the baseline in the meantime (see Figures 6 and 7). Here, too, as with double DQN, a drop can be seen at the beginning of the training. Overall, however, there is no clear difference between the two approaches (baseline, dueling).

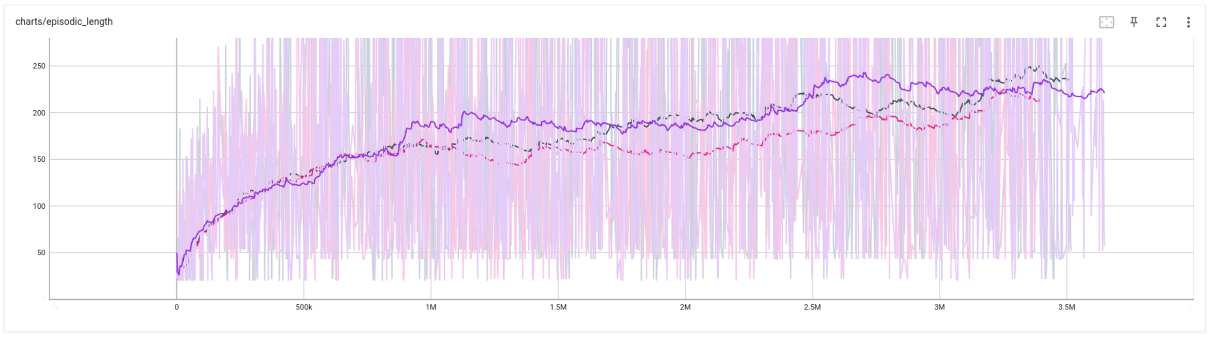

Figure 6: Episode Length

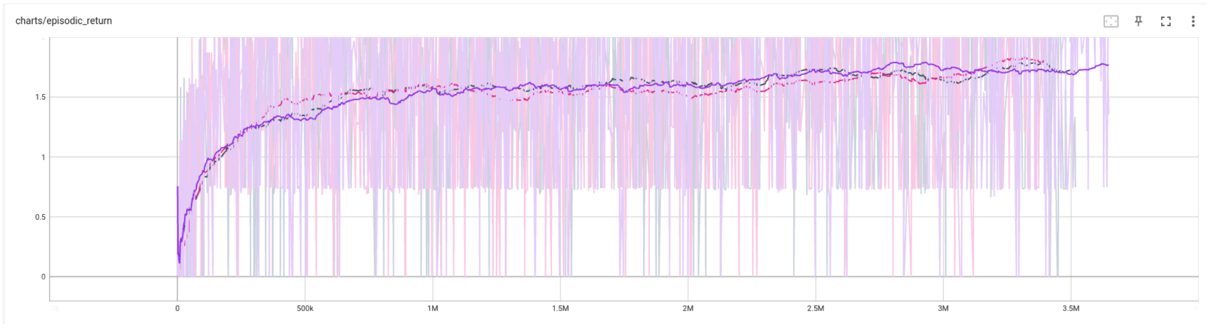

Figure 7: Episode Return

Only the overestimate is somewhat lower for the Dueling DQN than for the baselines. This may explain the earlier achievement of a longer average episode length. Even if this does not bring a clear advantage for the entire training. This can be seen in Figure 8.

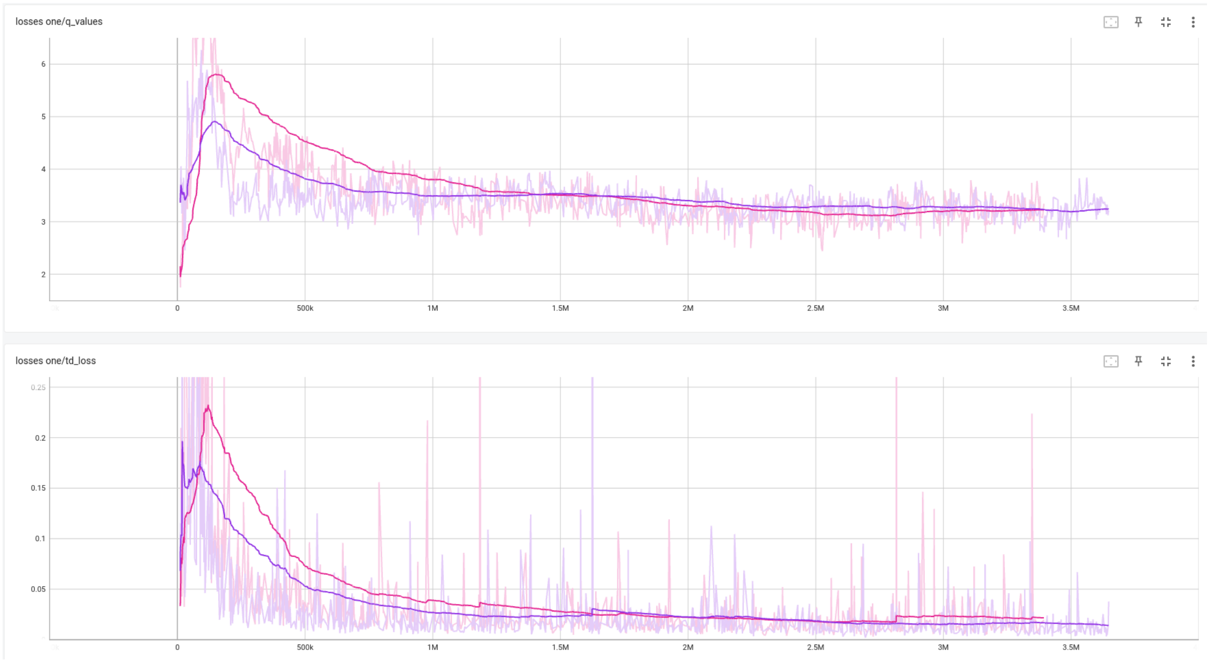

Figure 8: Loss

d) n-step DQN

At the n-step, a total of two approaches were implemented for *n=6*. The first approach uses only the replay buffer for implementation. Here, an adjusted n-step return is returned when the buffer is sampled for learning. However, this approach is error-prone, because in the case that no terminal state is reached within the next n steps, the Q-value to be added cannot be collected correctly with the existing structures. For this, either the network would have to be passed during sampling or the Q-values would have to be stored additionally. This error became conscious only after the execution shown below. As before, there is no difference in the return or the length of the episodes, but the loss and the overestimation are clearly lower than in the baseline. This can be seen in Figures 9 and 10. The loss is depicted in Figure 11.

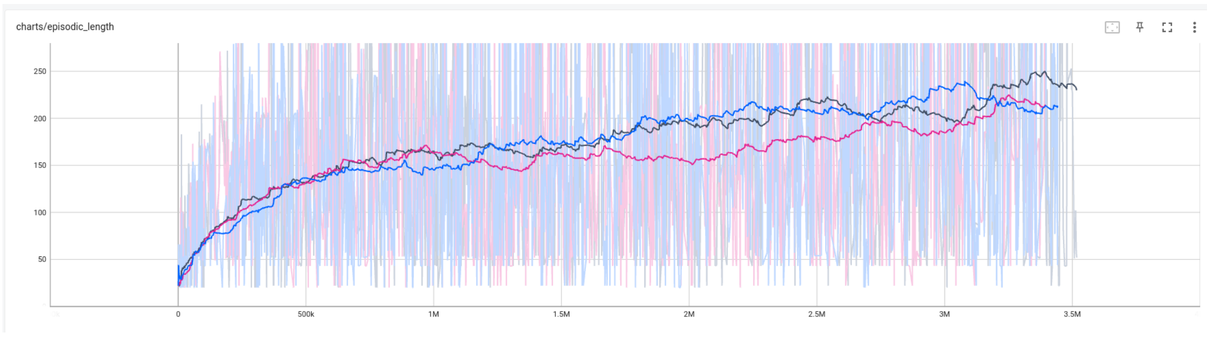

Figure 9: Episode Length

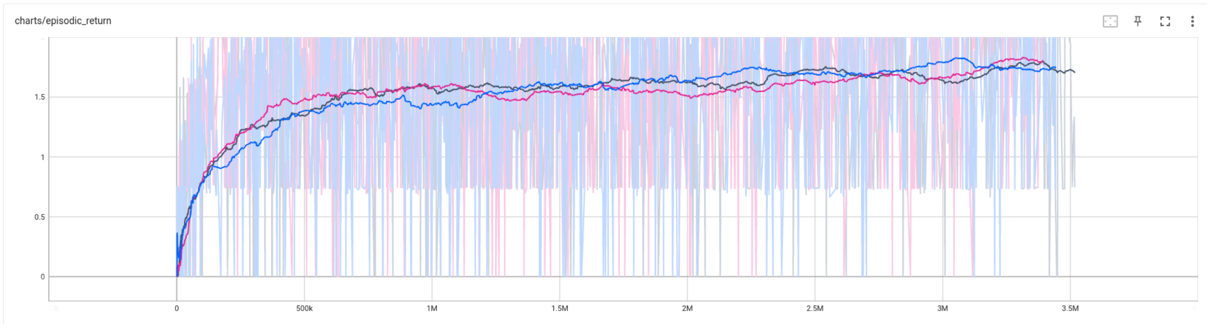

Figure 10: Episode Return

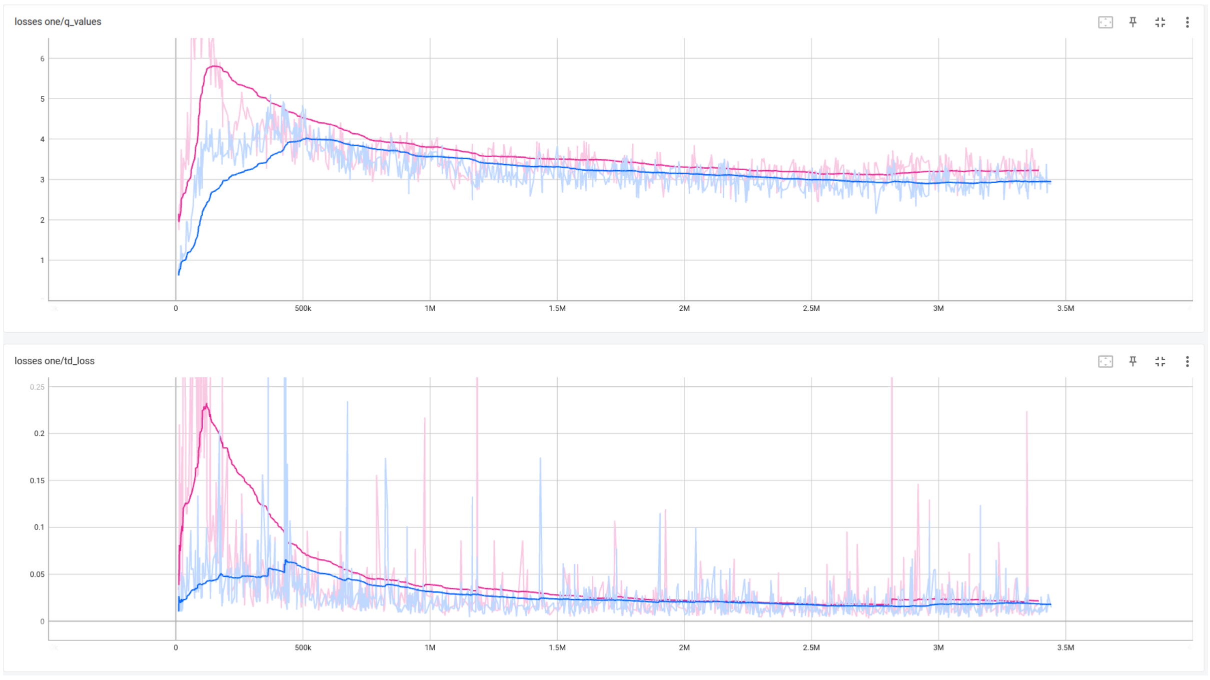

Figure 11: Loss

Instead of fixing the errors described above by adjusting the buffer, we decided to immediately store the n-step state-value prediction target in the buffer. The results can be seen below (Figures 12 and 13). This approach seems to be flawed as well since no improvement can be seen from about 200000 steps onwards while the loss and the q values grow rapidly. The corresponding error could not be found despite an intensive search. Note, that although the graph is much shorter (due to the short episodes) the hyperparameters were identical.

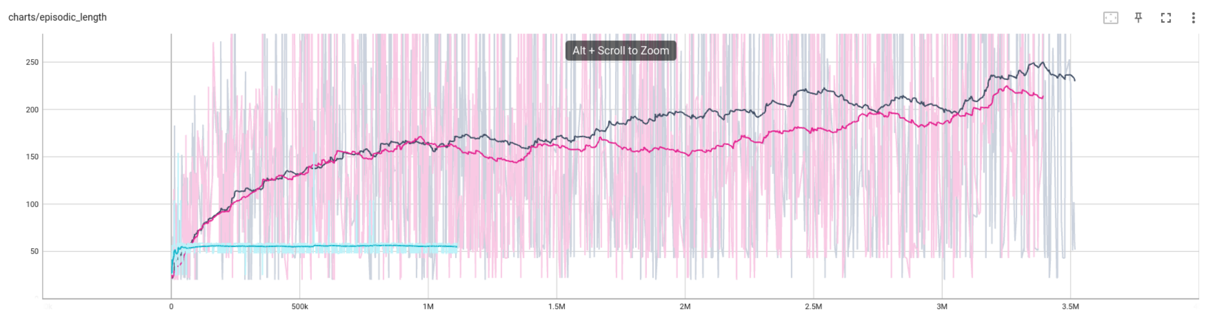

Figure 12: Episode Length

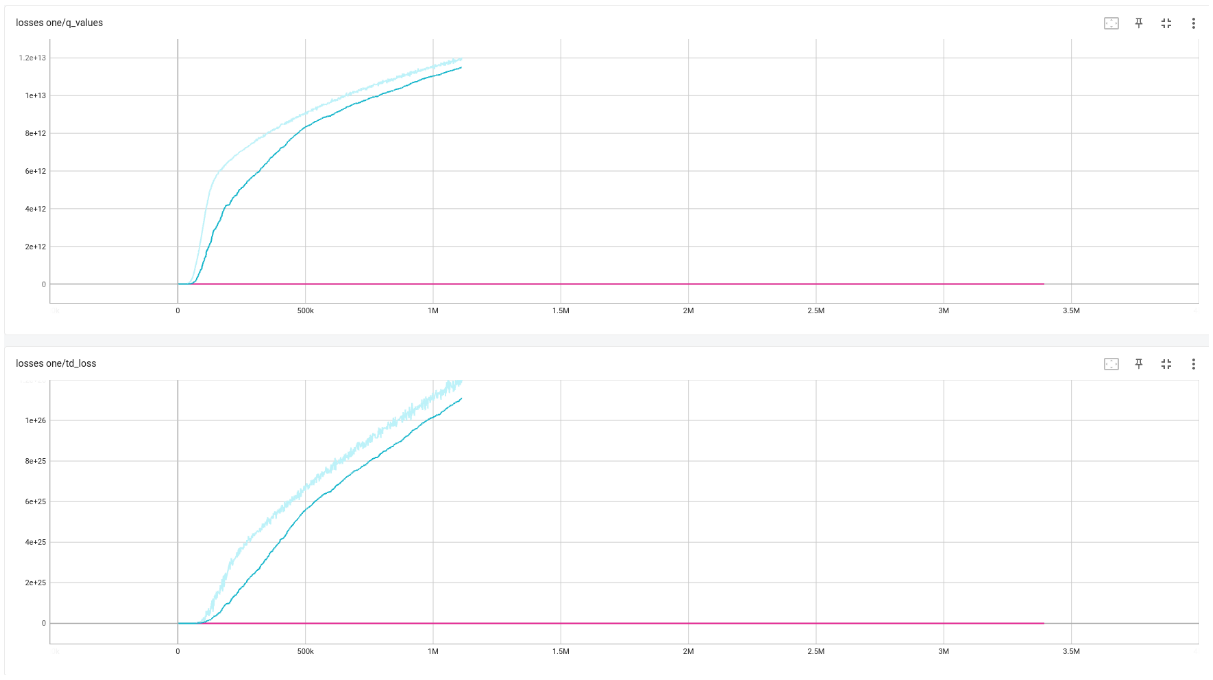

Figure 13: Loss

Best Model:

Our best model is a variation of Double DQN. Here, four DQNs are used in total. Two targets and two learning networks. For prediction, we take the max action of the summed prediction of the two learning networks. For training the learning_network1, we use the learning_network1 for q-values and best action and the target_network2 for the next q-values. For training the learning_network2 we use the learning_network2 for the q-values and the best action. While we use the target_network1 for the next_qvalues.

Every X steps randomly one of the target networks (1 or 2) is synched with its respective learning network.

Since it is always randomly selected which of the two networks are learned, but the learning frequency is identical to the previous models, as well as all other parameters, the computational effort is identical. In general, it can be seen that the models learn slower than all previous models, but achieve better results in the long run. Both for the episode length and for the return. However, one drawback is that this method doubles the number of parameters. Figures 14 and 15 illustrate the episode lengths and returns.

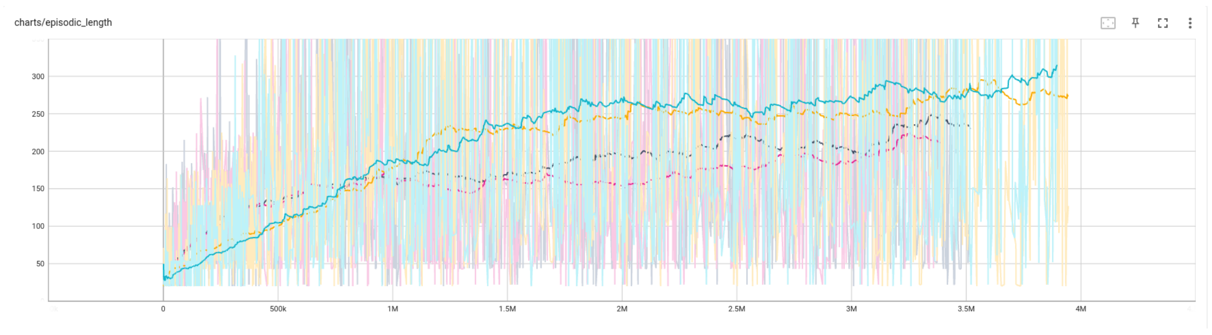

Figure 14: Episode Length

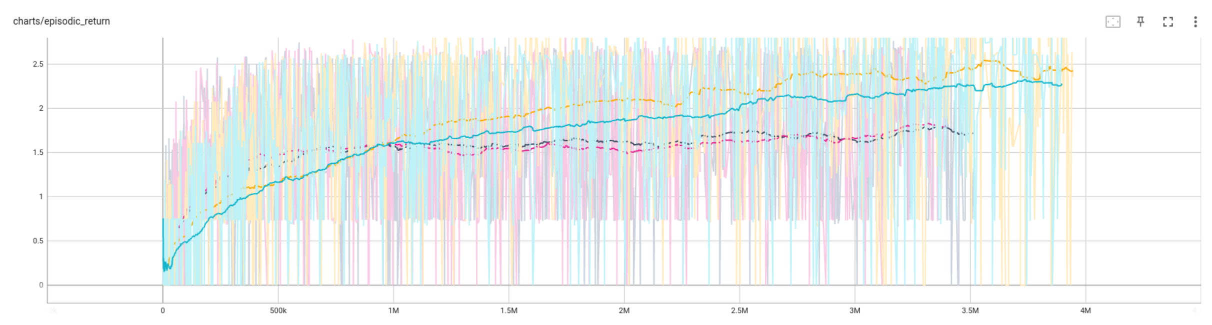

Figure 15: Episode Return

If we look at the loss and the q-values, however, we can see that there is no overestimation, but the loss increases with time, contrary to all other models.

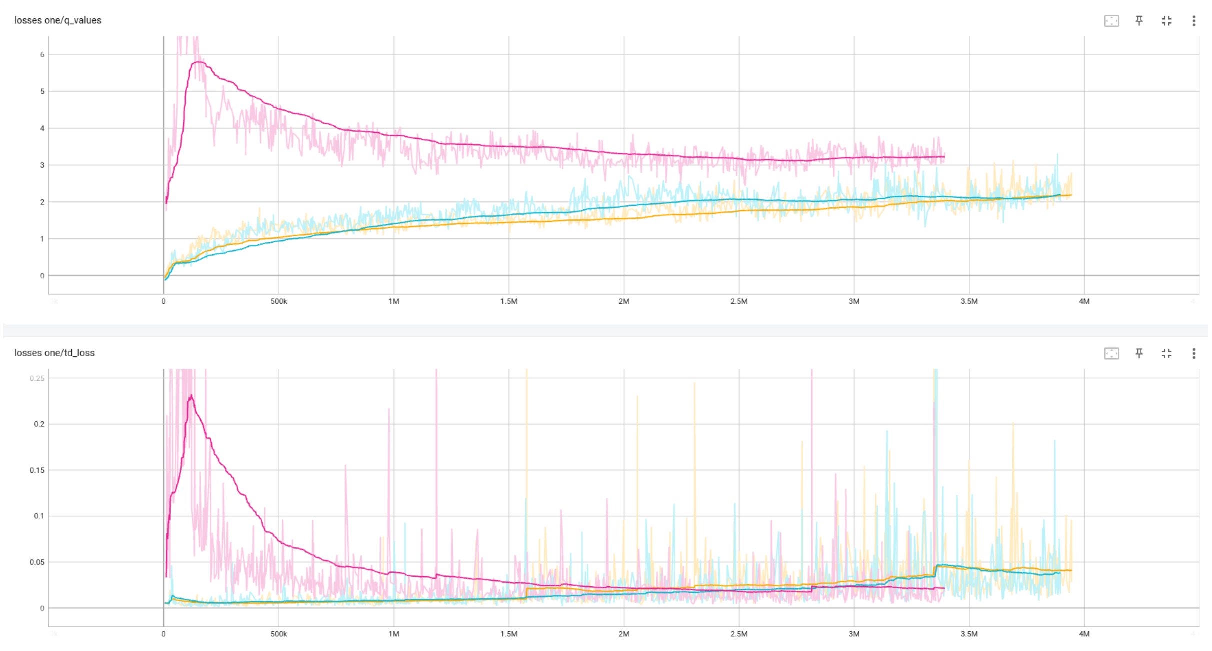

Figure 16: Loss

Method	SPS	Eps. Len	Return	TD Loss	Loss Q-Val
Plain	204	214	1,807	0,021	3,22
Dueling	187	220	1,765	0,014	3,24
Double(2)	113	213	1,751	0,018	2,95
Double(4)	118	273	2,422	0,041	2,19
n-step flaw	161	213	1,734	0,628	21,39
n-step	177	54	0,742	1,11e+16	1,2e+13

Table 2: Overall Results after 20000 episodes

Note: If multiple runs were executed the best reward was chosen.