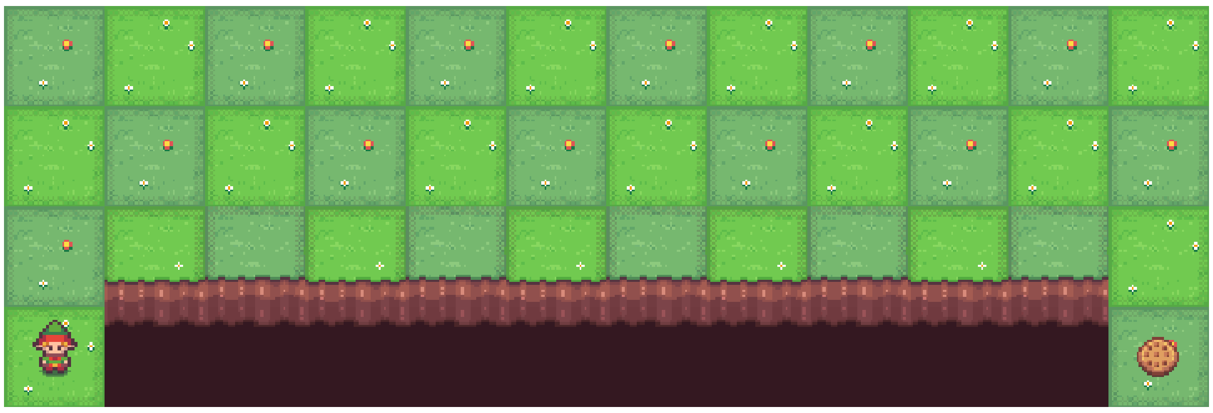
Laboratory 6

Variant #3
Mariya Zacharneva and Ruslan Melnyk

Cliff Walking



The task

The task is to develop the Q-Learning algorithm and train an agent to find an optimal path from start to end. The gymnasium library provides the environment. In our task, the agent can walk in a 2D gridworld of size 4x12. A cliff runs along [3, 1..10], and the agent should avoid falling off the cliff.

Observation Space

In total, there are 12 * 3 + 1 possible states for observation (final state doesn't count, because the game ends). The observation represents the player's current position as

current_row * ncols + current_col

Both the row and column numbering start from 0. So, for example, at the start, the agent stands at column 0 and row 3 → the observation value is calculated as 3 * 12 + 0 = 36. It means that the starting state is state 36.

Action space

There are four possible actions that the agent can perform: move up (0), right (1), down (2), or left (3). If the agent takes an action that would move it off the grid,

- the environment does not move the agent,
- applies default reward (-1),
- returns new_state == state, not an error.

End of the episode

The episode finishes if the player reaches the goal (row 3, column 11, in other words, state 47). If the player falls off the cliff, it returns to the start location.

Reward

Each step gives an agent a -1 reward (this is also a default reward when moving off the grid). If the player falls off the cliff, they receive a negative reward of -100. Note that all rewards are negative: this makes the agent try to end the game faster (get to the end point) to maximize the total reward (minimize the number of steps in the episode).

Learning algorithm

Q-Learning

Q-Learning is an off-policy value-based method that uses a temporal difference approach to train its action-value function. Temporal difference means that the agent doesn't wait till the end of the episode to start learning; instead, it learns at each step by updating a state value dynamically. The agent carries a table with all possible states and actions to perform, and calculates the value for each cell: a higher value means a higher preference to perform the particular action from a particular state.

Hyperparameters

There are several hyperparameters that affect the learning:

- Number of episodes: how many times the agent starts and finishes the game.
- Learning rate (alpha): how fast the information is updated, how quickly new info overrides the old info
- Discount (gamma): how important are future rewards
- Exploration rate (epsilon): probability of choosing the random action instead of preferring the best already known action
- Exploration rate decay (epsilon decay): a mechanism that allows to reduce the exploration rate gradually over time, when more information is already discovered

The main learning process is performed according to the following formula:

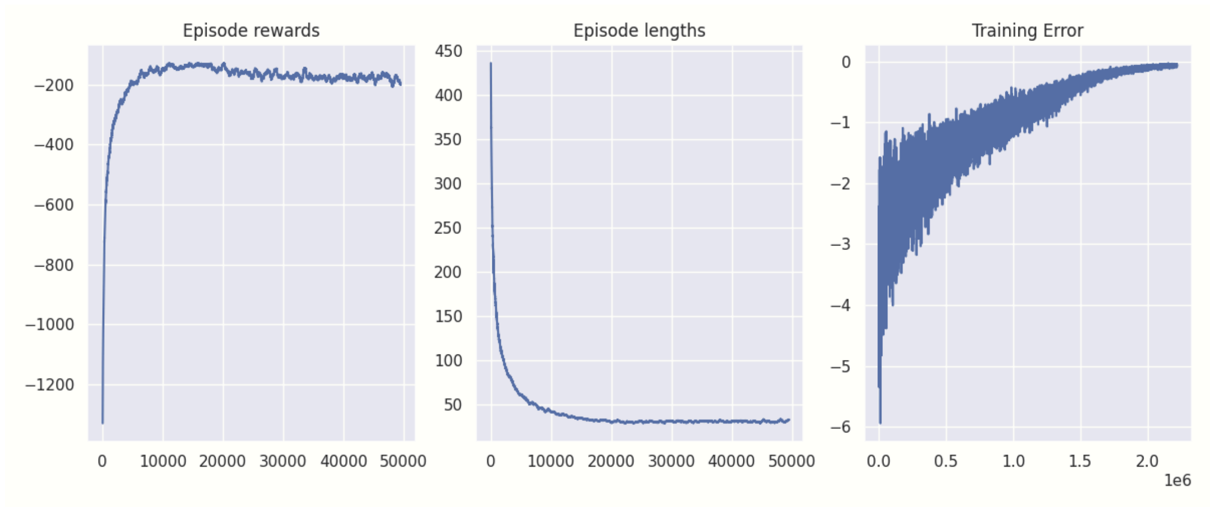
$$new\_q\_value = old\_q\_value + \alpha * (target - old\_q\_value)$$

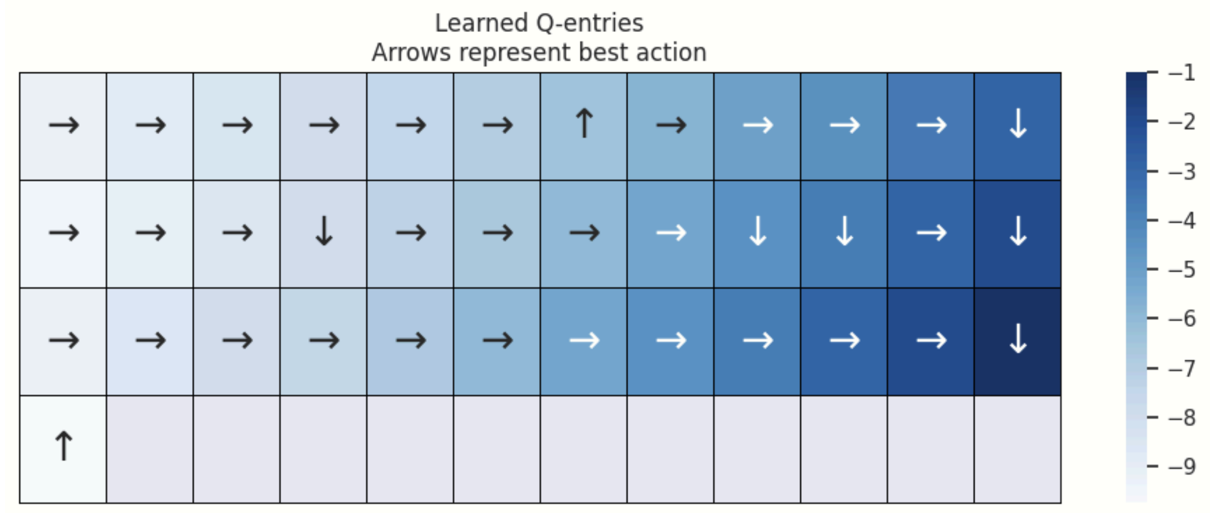
$$target = reward + \gamma * max\_reward\_in\_next\_state$$

Here, alpha is a learning rate, and gamma is a discount.

To have a reference, let's start with the following parameters:

n_episodes = 50000
learning_rate = 1e-3
discount = 0.95
epsilon = 0.3 (no decay)

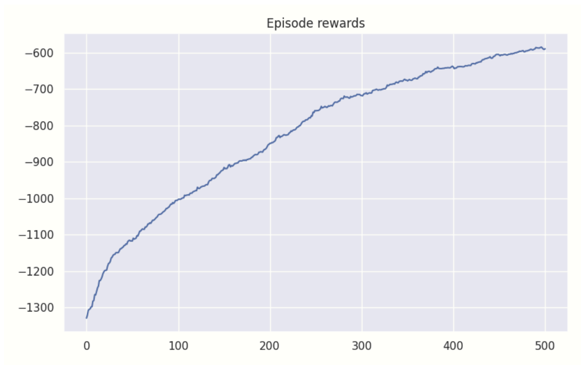
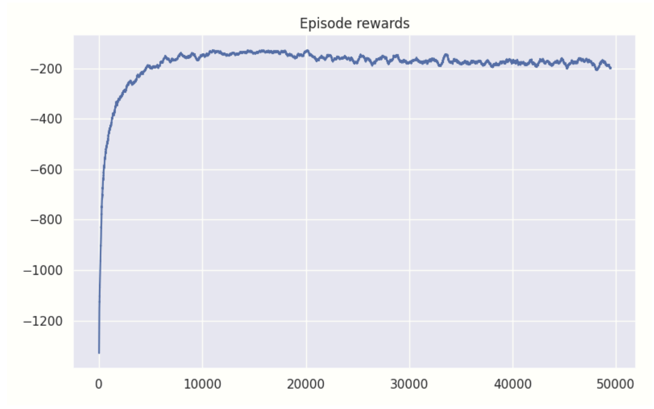
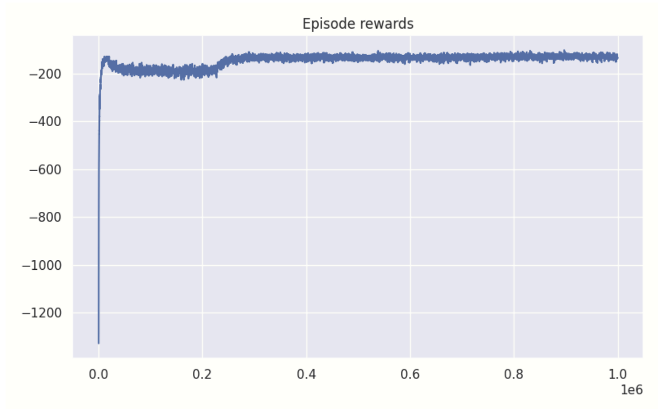
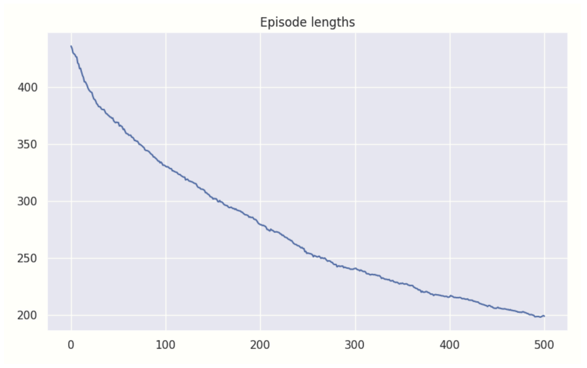
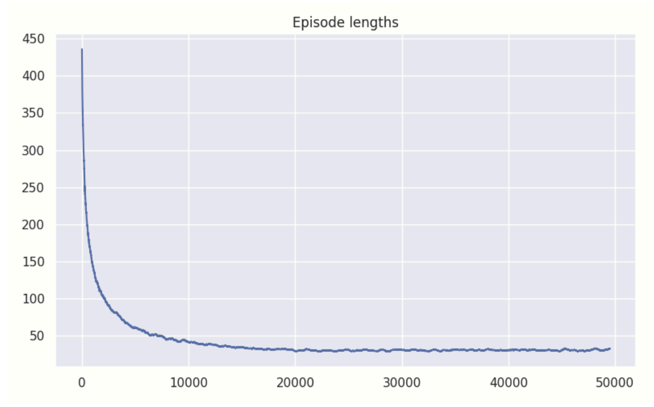
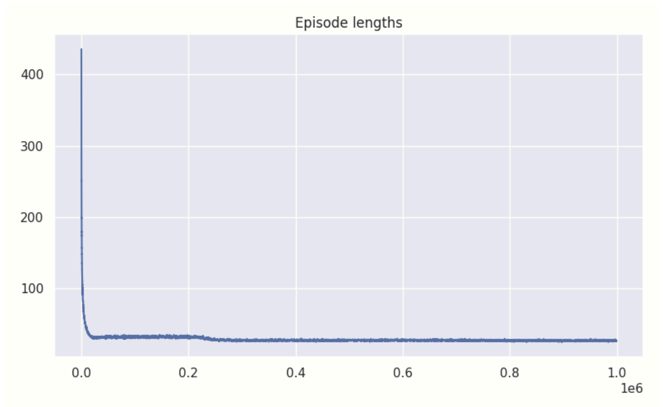


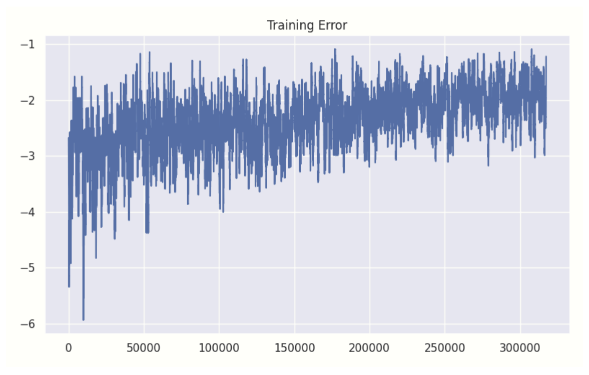
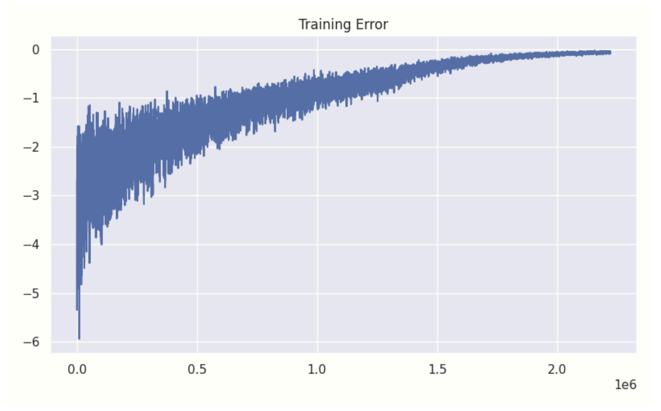
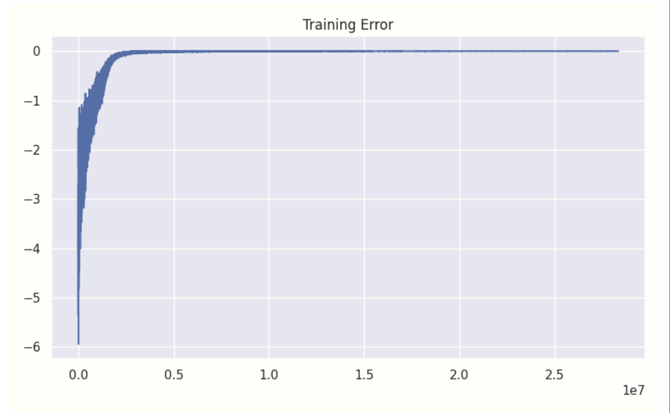
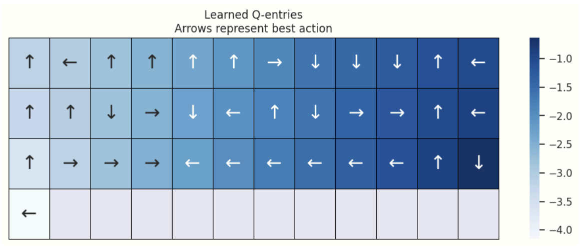
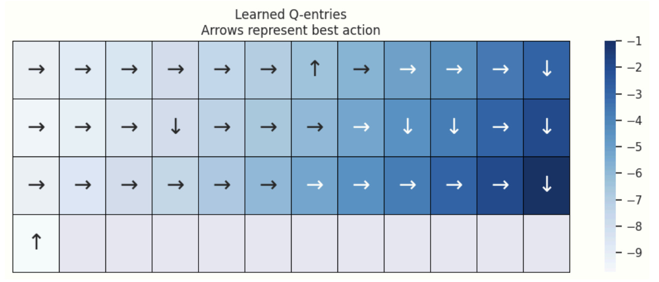
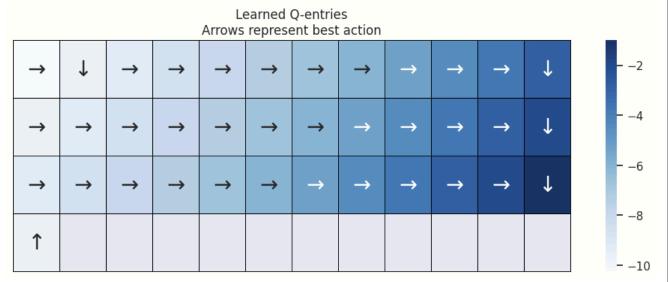


We can see that the learning was successful, and the agent was able to find a path to the goal cell with minimal falling off the cliff. The heat map shows the most preferred action for each cell. We can see that sometimes the agent can prefer to go off the grid, because this behavior is not punished extensively, but we can see that it never prefers to go off the grid.

Now let’s conduct some experiments with different values of hyperparameters. When changing one parameter, the rest of them will be set the same as in the previous example.

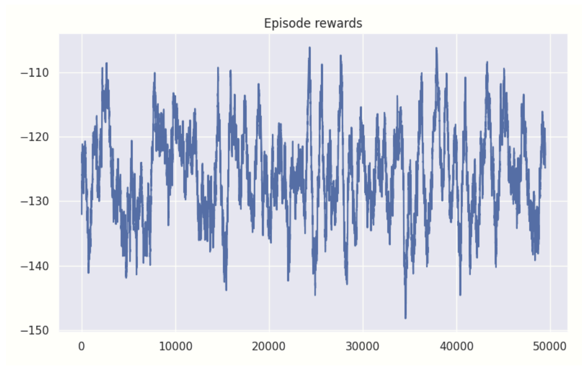
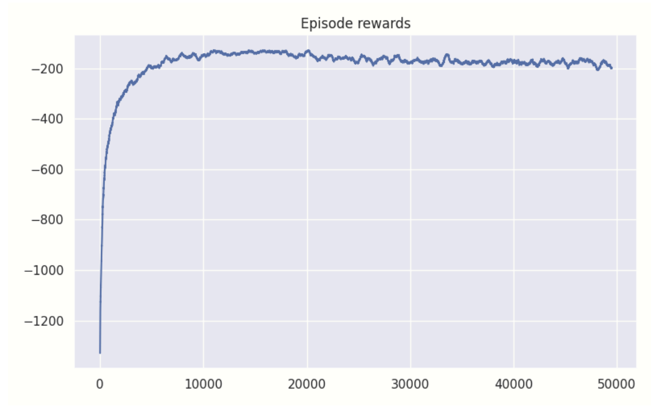
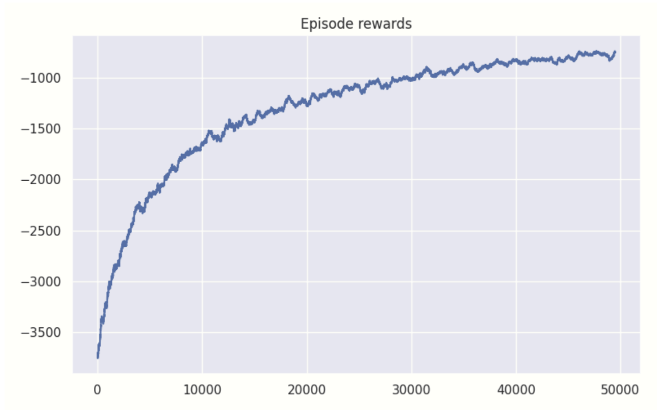
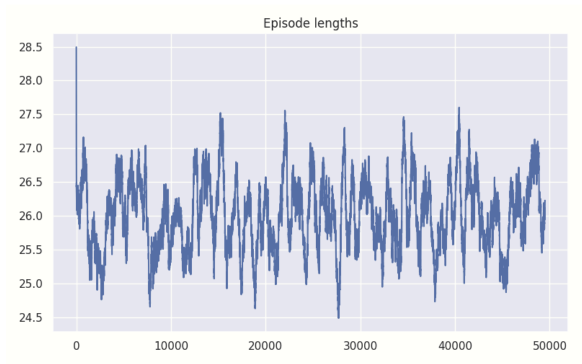
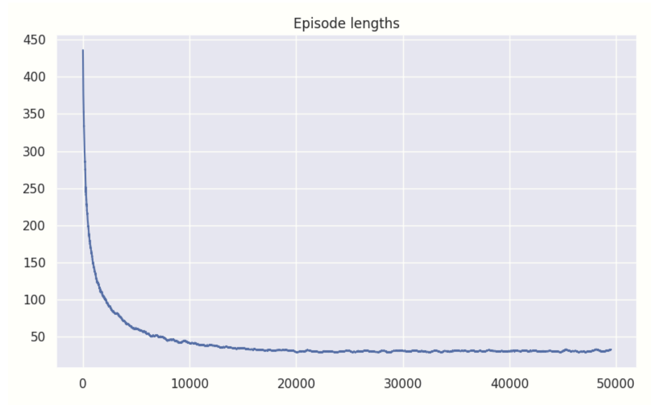
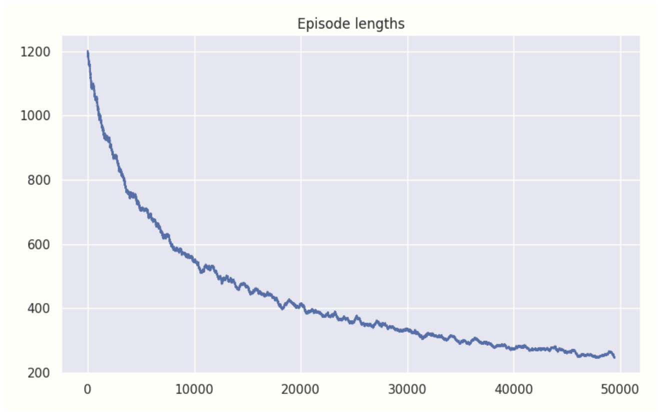
1. Number of Episodes

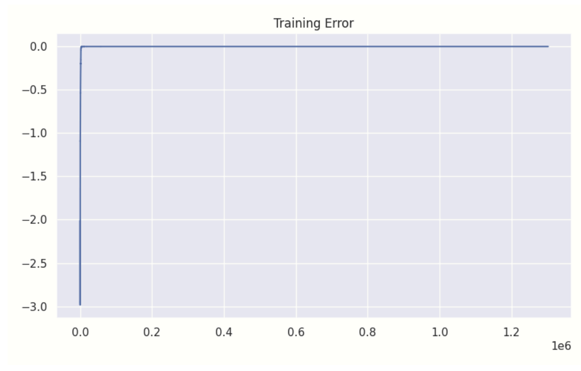
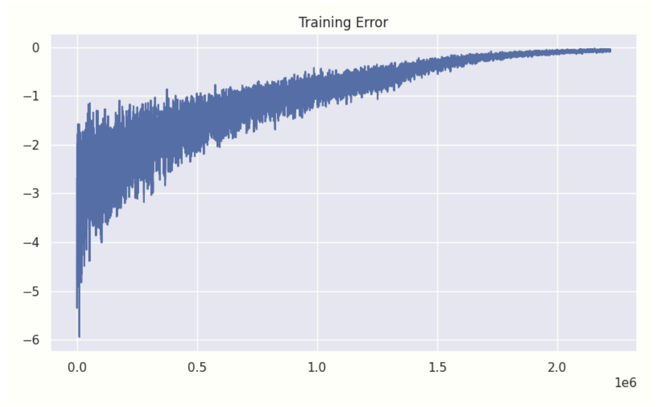
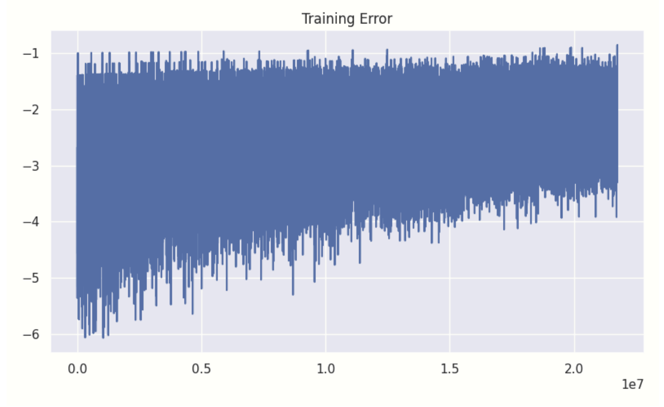
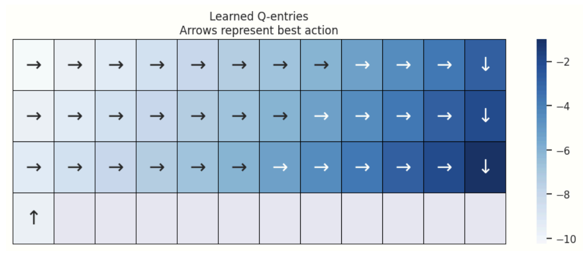
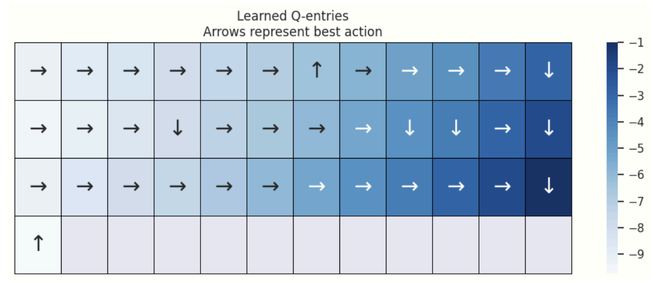
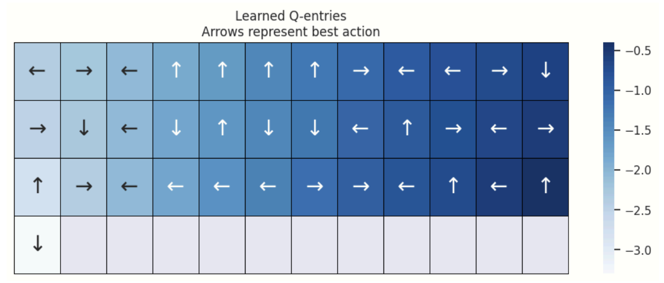
	1000	50000	1000000
rewards			
lengths			

errors over updates			
Q-table			
total training time	00:04	00:37	7:52

As we can see, most of the learning is performed in the first 10000 episodes; after that, the total reward doesn't grow and fluctuates over a more or less the same number. The only thing that improved in the experiment with a million episodes is a heat map: the algorithm perfected its understanding of the grid world, and goes straight to the final cell.

2. Learning Rate

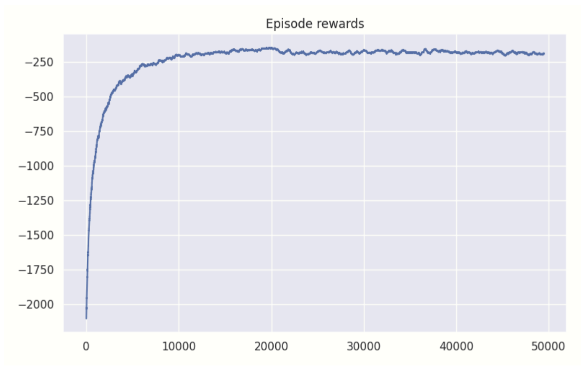
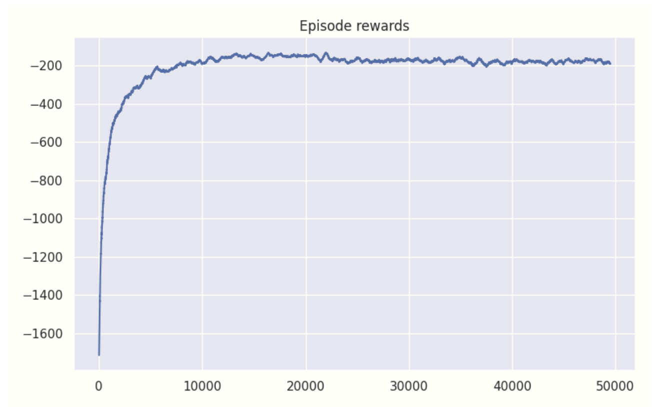
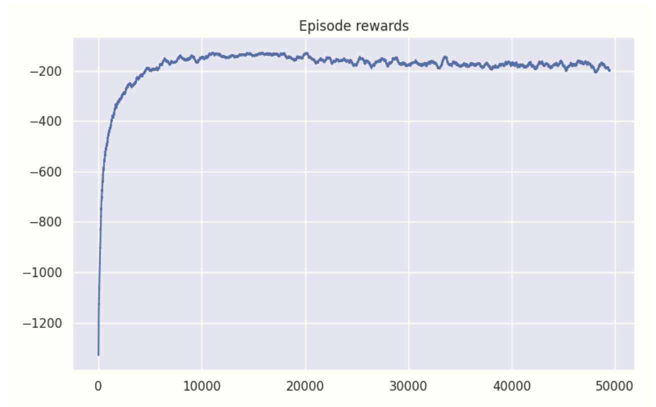
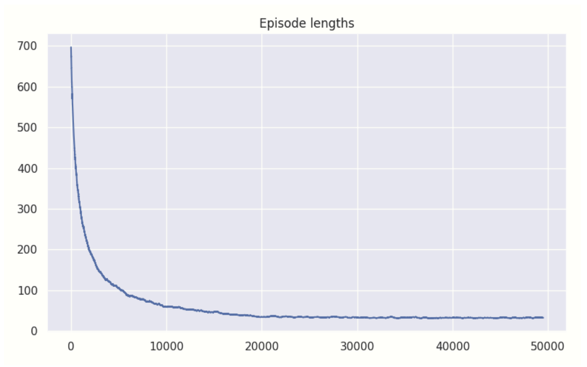
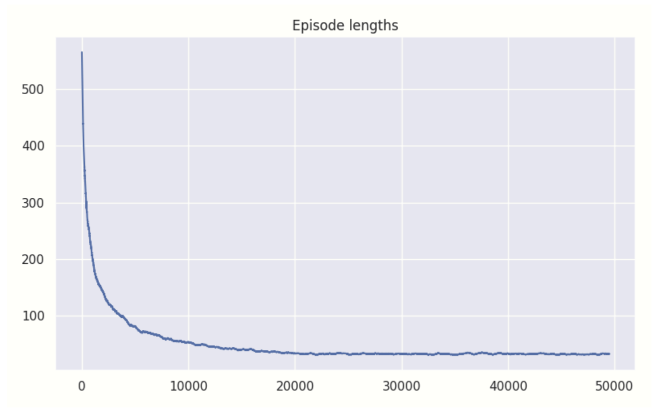
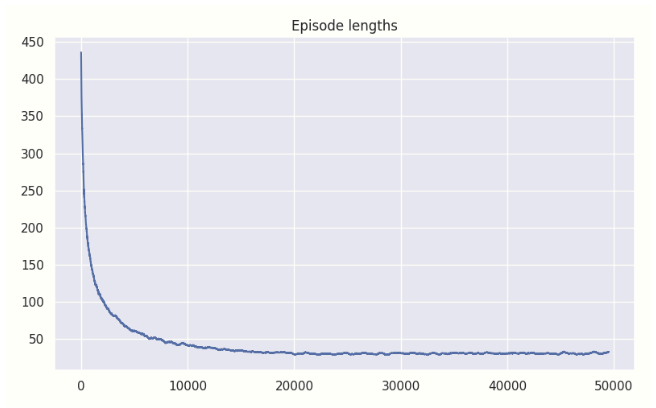
	1e-2	1e-3	1e-5
rewards			
lengths			

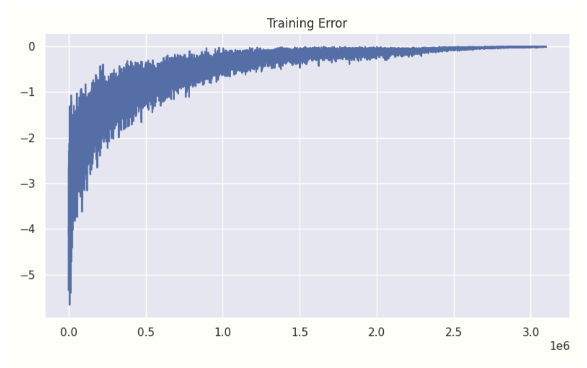
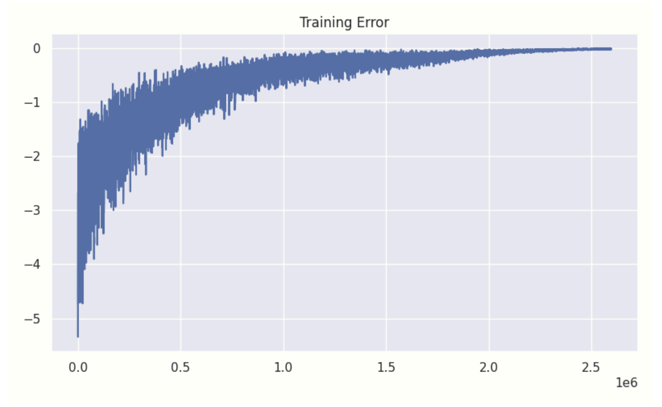
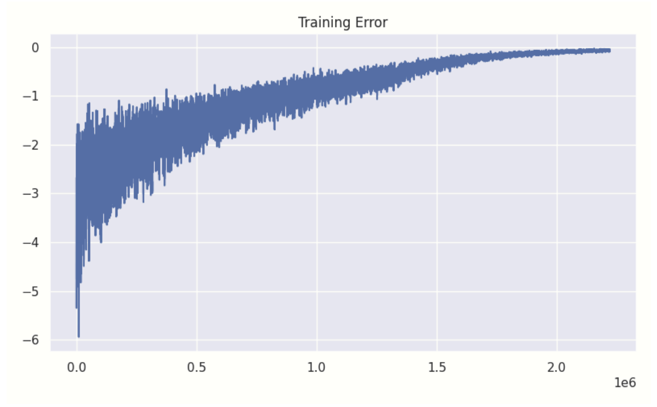
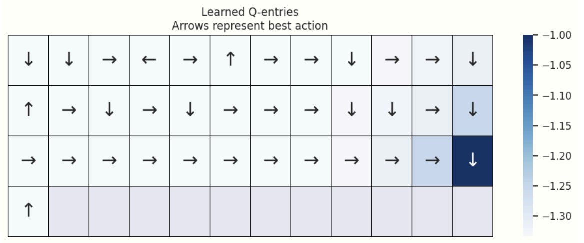
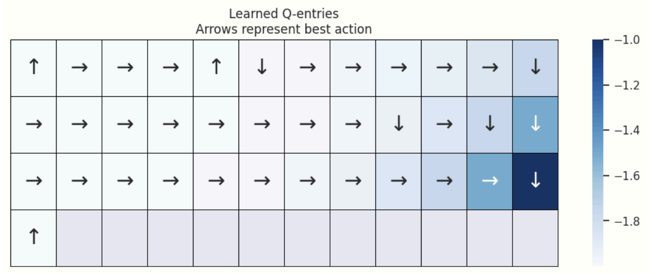
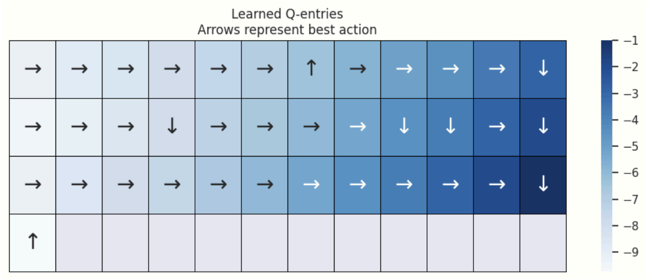
errors over updates			
Q-table			
total training time	00:19	00:37	05:54

The learning rate describes how important for the algorithm is the newly acquired information: if the learning rate is high, new information gets higher priority than old one; in turn, a low learning rate means that the old information is treated as more important, and updates happen slowly.

Learning rate of 1e-2 results in aggressive learning: model updates very fast, almost from the first episode; however, there is almost no improvement over time. Lower learning rate (1e-3) results in a slower, more stable learning trend. In practice, after 500000 episodes, both models achieve approximately the same reward value. Even a lower learning rate leads to the model training so slowly that there is very little improvement over a high number of episodes.

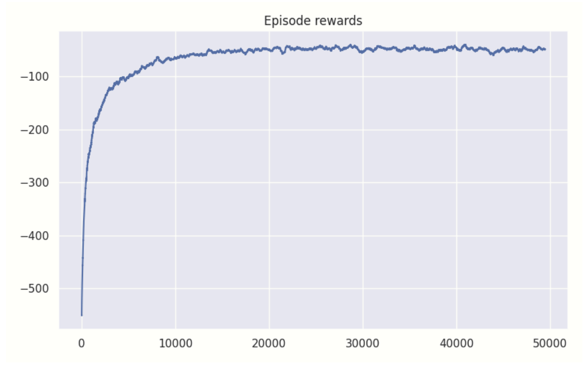
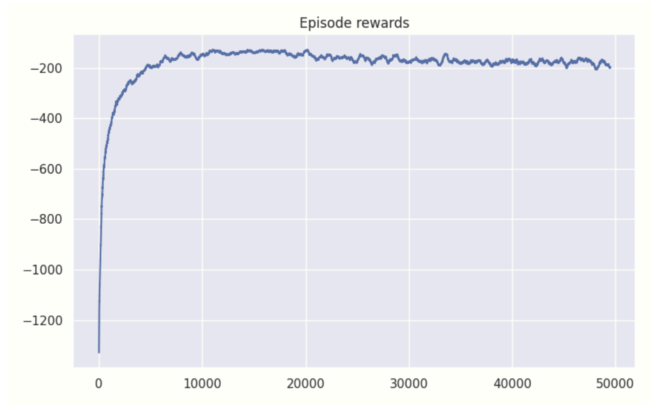
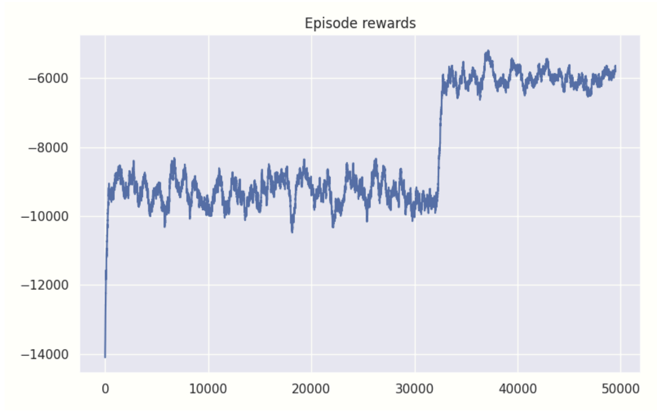
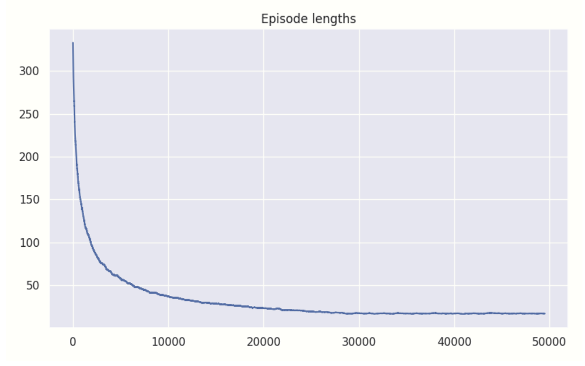
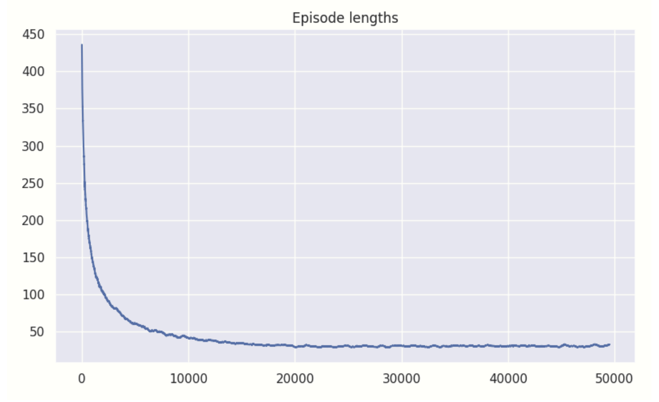
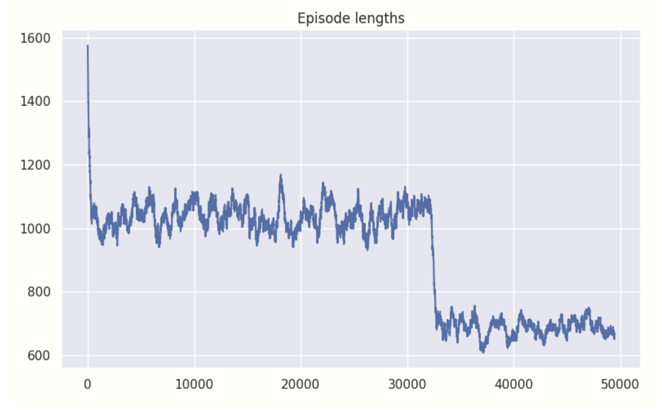
3. Discount

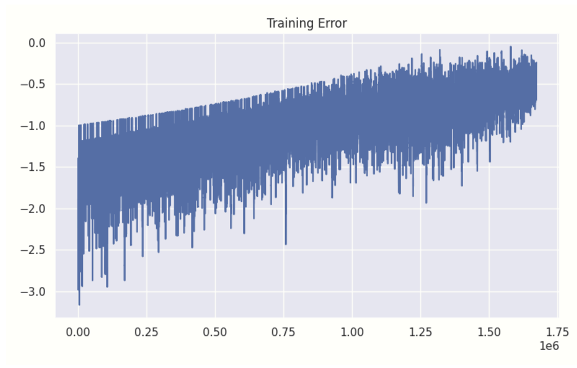
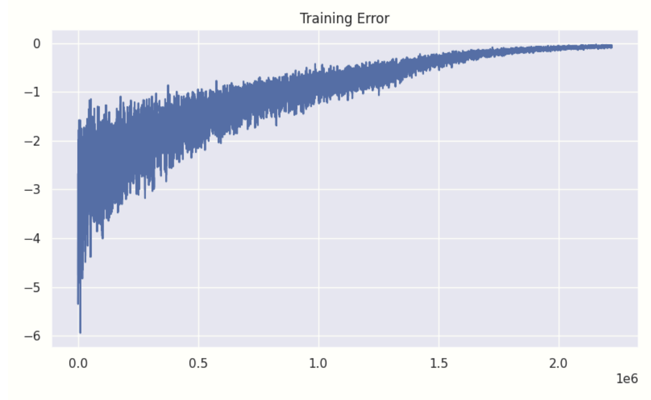
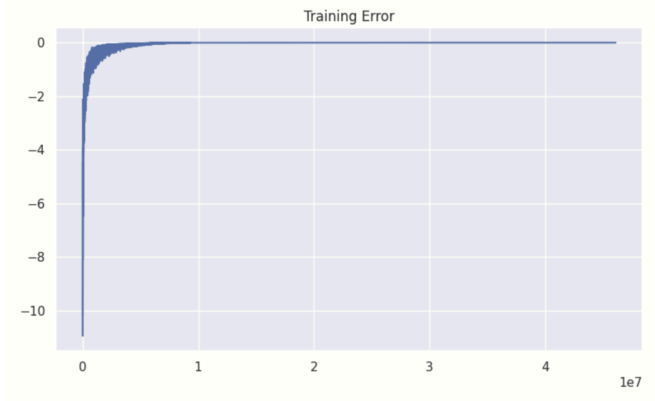
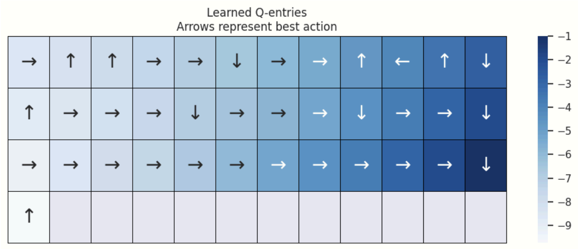
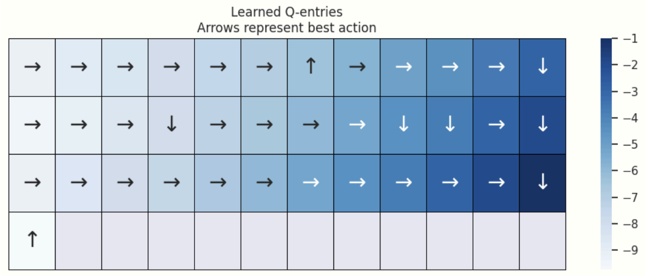
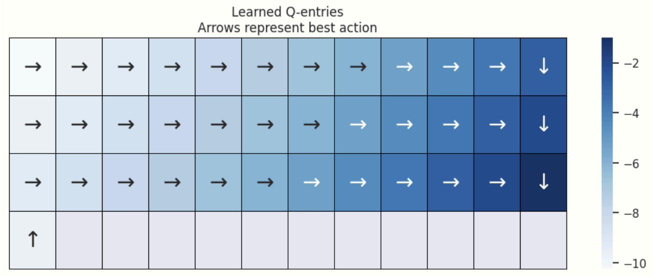
	0.25	0.5	0.95
rewards			
lengths			

errors over updates			
Q-table			
total training time	00:48	00:40	00:35

Intuitively, the discount factor describes whether the model cares about immediate reward or a long-term return. Low discount rate shows that the model doesn't care about long-term results: areas near the cliff are not indicated, and at the last step, it found an immediate reward. In contrast, a high discount value shows that the model values the long-term result and knows the optimal path to get to the end of the game.

4. Exploration/Exploitation Strategy

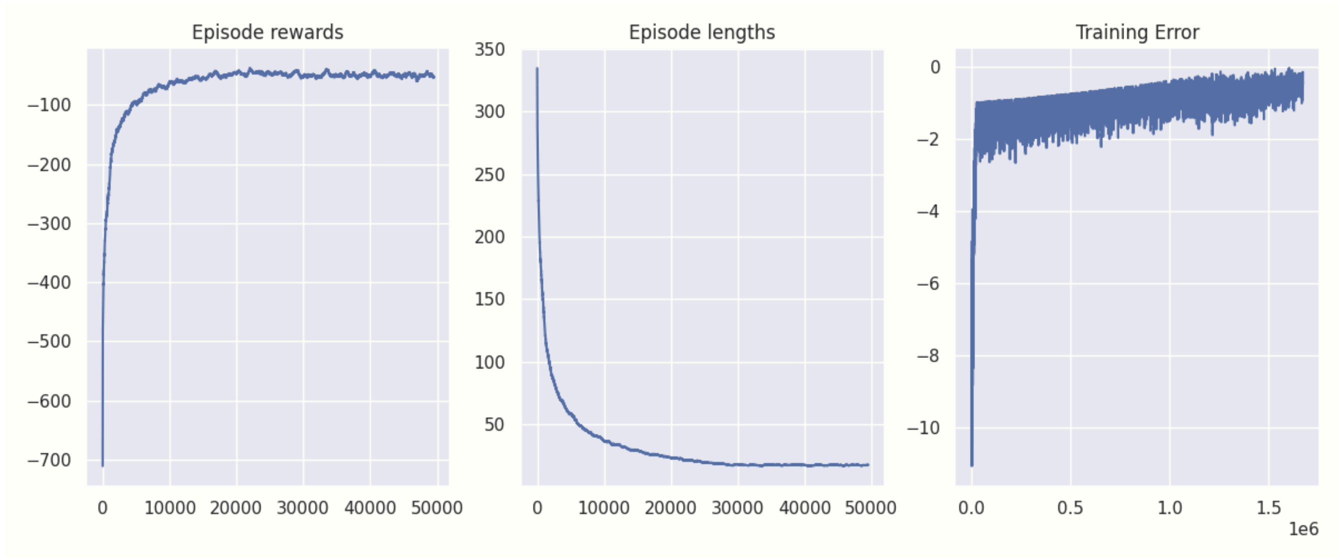
constant epsilon	0.1	0.3	0.9
rewards			
lengths			

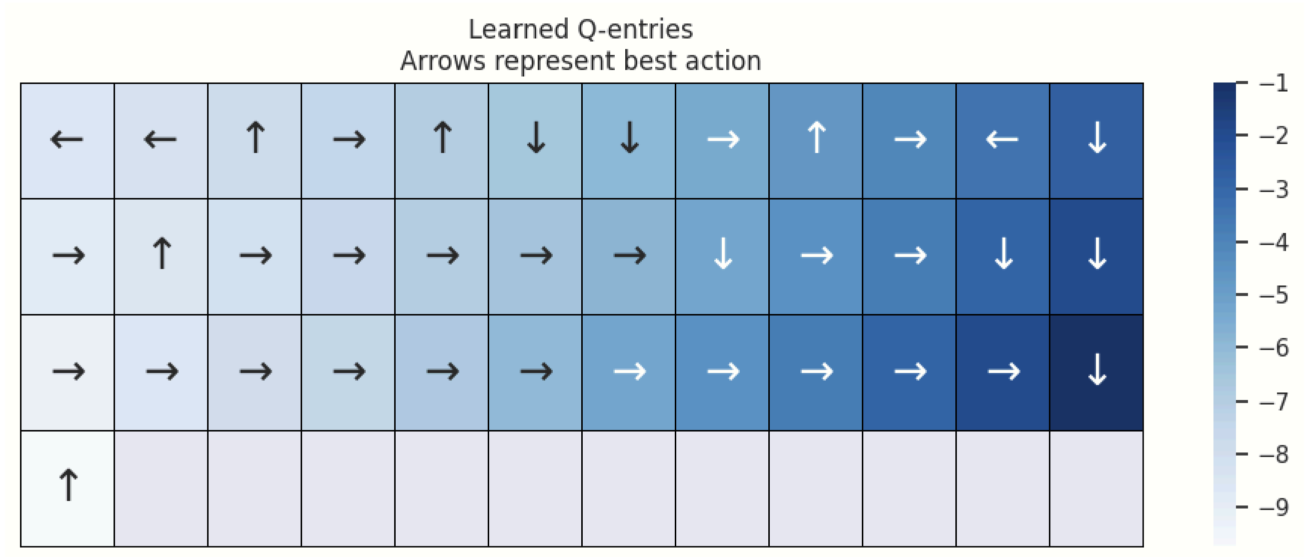
errors over updates			
Q-table			
total training time	00:24	00:34	10:56

The exploration rate describes the probability with which the model will try a random action instead of a previously found good one: a low exploration rate creates a conservative model, while a high rate creates an adventurous and risky model. The values of 0.1 and 0.3 produced good results: it is just enough to find a good path and prefer it mostly, while also occasionally exploring random actions and potentially finding a better way. When the value is too high, for example, 0.9, the model always prefers to make a random action; the reward does not improve. However, this small probability of choosing the best-known path is enough for creating a nice-looking heat map: we can say that the model *knows* the best path, but never actually *prefers* it.

Epsilon Decay (1.0 -> 0.1)

Epsilon decay allows for taking the best from high and low exploration rates: in the beginning, the model will prefer to explore and collect more data, and later on, it will become more conservative and prefer the best paths, providing stable results.





←	←	↑	→	↑	↓	↓	→	↑	→	←	↓
→	↑	→	→	→	→	→	↓	→	→	↓	↓
→	→	→	→	→	→	→	→	→	→	→	↓
↑											

Conclusions

We have explored Q-Learning method and different values of hypermarameters. In each experiment there were following best parameters found:

- number of episodes: 500000
- learning rate: 1e-3
- discount: 0.95
- exploration rate: epsilon decay from 1 to 0.1

More precisely identifying best combination of parameters would require more time, however, our results shows a decent model learning, and we were able to achieve a stable episode reward of less than -100 (meaning no stepping off the cliff).