CS F317 Assignment 2 Report

Pratyush Bindal 2022A7PS0119H
Uday Sudani 2022A7PS0136H

Abstract

We investigate off-policy Q-Learning-based Reinforcement Learning (RL) approaches for agent navigation in a CLIF-style environment. We emphasise on the methodologies utilised and evaluated performance of the Q, Double-Q, Triple-Q, and Quadruple-Q Learning algorithms. Our basis of this report is based on the ideas of popular games such as Super Mario, Flappy Bird and Hill Climb Racing, and we tried to mimic and modify existing game formulations, and implemented a custom game-style environment.

We utilised variations of Q-learning algorithms so that the agent moves through a dynamic environment filled with static and dynamic dangers, platforms, adversaries, coins, and boosts, while experiencing other stochastic aspects. The agent chooses optimal action to reach the goal by collecting maximum amount of coins and also avoiding to incur penalty by falling into hazards or other pitfalls.

Traditional Q-Learning updates a single Q-table, which can lead to overestimation bias, but Double Q-Learning addresses this issue by spreading the Q-value updates across two Q-tables. Triple and quadruple Q-Learning enhance this by include more Q-tables, which reduces overestimation while increasing computing cost.

Various assessment measures, such as cumulative rewards, average rewards, and training time were used to examine the effectiveness of each technique in complicated environments. The code developed for formulating the problem, implementing variations of Q-Learning methods along with obtained results mentioned in the report can be found at bPratyush/CS-F317-Reinforcement-Learning-Assignments: Coursework Assignments for CS F317 Reinforcement Learning.

Problem Formulation: Q-Quest

We focused on developing a reinforcement learning (RL) agent capable of operating within a dynamic and complex CLIF-based environment. The agent's objective is to reach the goal while collecting maximum number of coins while avoiding any hazards, dangers or pitfalls it faces while moving on the platform from start point to ending point, that is the goal.

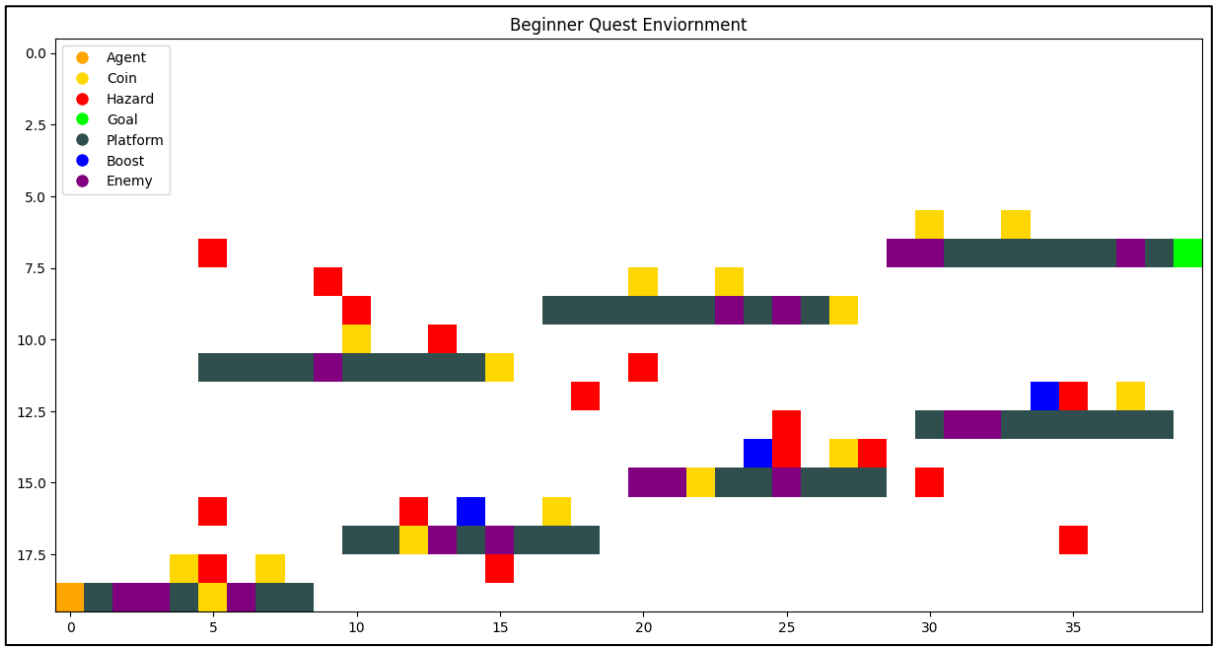
Environment Setup

The environment is represented as two-dimensional grid with each representing a state where agent can interact. There is a single agent which we named Pixel Piero, whose state is represented as a tuple of x-coordinate, y-coordinate, velocity. We implemented two versions of the game environment – the primitive version called Beginner Quest and an upgraded version called Superior Quest.

Beginner Quest Environment

Pixel Piero can take **three actions** in the Beginner Quest Environment which are moving towards left, moving towards right, and jumping up if he is on the platform. Different elements populate the grid, each introducing unique interactions and consequences for him:

1. **Navi Spaces:** Basic navigable spaces with no specific reward or penalty
2. **Skywalks:** Pixel Piero can only move on various platforms called skywalks of varying lengths scattered across the environment at different heights
3. **Genesis and Horizon:** Pixel Piero starts from the Genesis and the aim is to move till the Horizon, that is the end position, while maximizing rewards accumulation. Reaching Horizon gives high rewards. (Reward = +50)
4. **Pixie Tokens:** These tokens are scattered throughout the environment which Pixel Piero should collect for getting positive reward points (Reward = +5)
5. **Turbo Tokens:** These are the boosts which give higher reward than accumulating coins. (Reward = +10)
6. **Meany Beasts:** These are enemies placed randomly on the platform which walk on skywalks in predefined pattern which Pixel Piero does not know but should learn to minimise encounter. There is a penalty for encountering an enemy (Penalty = -15)
7. **Fixie Frights:** These are static hazards which are kept at fixed positions and do not move. Encountering them yields a penalty for Pixel Piero. (Penalty = -10)
8. **Tricky Traps:** These are dynamic hazards which are initially placed at fixed positions and are the moved after each episode in a pattern. Encountering them yields a penalty for Pixel Piero. (Penalty = -10)
9. **Drop Zone:** Pixel Piero will face penalty if it falls out of skywalk towards the ground. (Penalty = -20)
10. **Gravity and Jump Strength:** These are additional elements which will alter Pixel Piero's vertical velocity. (Jump Strength = 2 units and Gravity = 1 units)

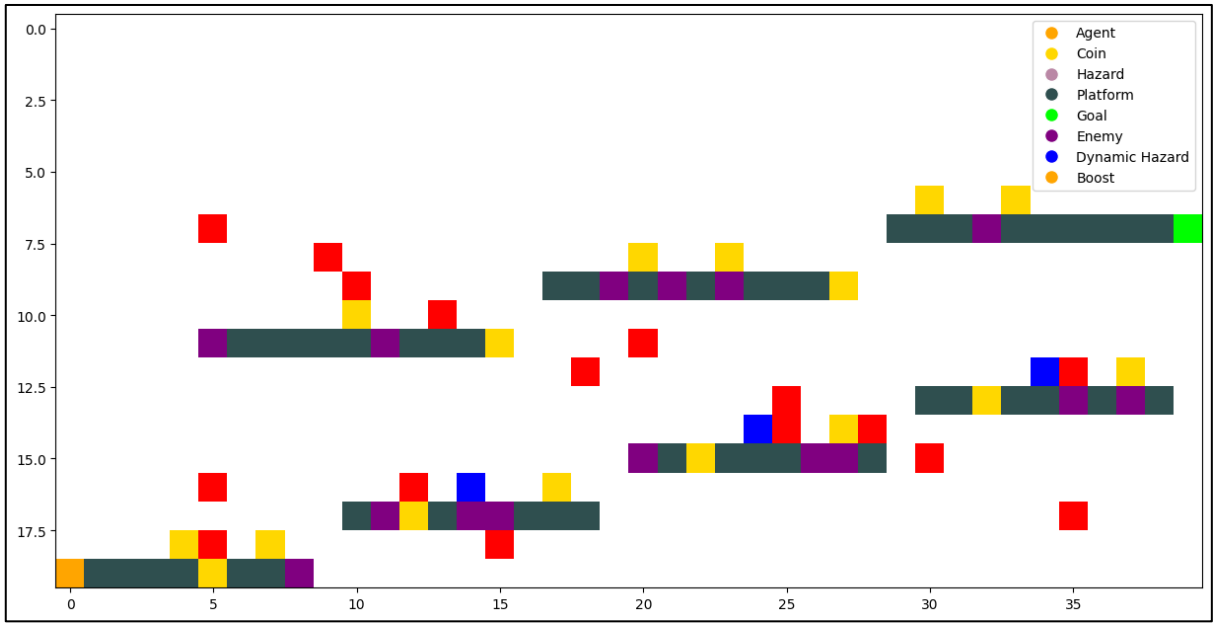


Superior Quest Environment

It is an upgraded version to the Basic Quest Environment, where Pixel Piero can take **four actions** instead of three which are moving towards left, moving towards right, jumping up if he is on the platform and an additional action to kill Meany Beasts. The modifications of different elements populating the grid with respect to Beginner Quest Environment, each introducing unique interactions and consequences for Pixel Piero has been listed below:

1. **Horizon:** Reaching Horizon gives very high reward (Reward = +500)
2. **Danger Zone:** Falling out of skywalk attracts higher penalty (Penalty = -30)
3. **Meany Beasts:** Pixel Piero can now kill enemies and penalty for killing now incorporates the count of Meany Beasts killed multiplied by the original penalty.
4. **Survival Quest:** There is an additional penalty for each step taken to reach the goal which will encourage Pixel Piero to reach the goal in minimum count of steps. (Penalty = -1)
5. **Gravity and Jump Strength:** Jump Strength has been enhanced to 4 units whereas the logic for incorporating gravity has been modified. If Pixel Piero is not on a platform and can fall, it enables gravity, hence, increasing its vertical velocity and checking how far it can fall before hitting a platform. If Pixel Piero reaches a platform, it stops falling instead of dropping to the ground. If Pixel Piero lands or is on a platform, his downward speed is reset to zero.

Remaining elements incorporated in Q-Quest remains same for both environments.



Pixel Piero's Objectives and Constraints

The RL agent must navigate through the environment to achieve the following:

1. **Maximise Cumulative Rewards:** Pixel Piero should maximise cumulative reward accumulation by collecting pixie and turbo tokens and reaching the Horizon.
2. **Avoiding Penalties:** Pixel Piero should avoid encounters with Meany Beasts, Fixie Frights and Tricky Traps by learning their movement pattern or remembering their fixed positions. Furthermore, Pixel Piero should avoid going into Drop Zones, as there is no recovery point from that zone, hence, the current episode would terminate. In Superior Quest Environment, Pixel Piero should also take care of count of steps to minimise the penalty for each step taken.

Implemented Reinforcement Learning Algorithms

We implemented the variations of the Q-Learning algorithm to analyse comparatively the difference and similarities between the variations while also selecting the best algorithm for the two environments listed above. The parameters used for the two environments have been listed in the following table:

Hyperparameters	**Beginner Quest Environment**	**Superior Quest Environment**
Learning Rate	0.02	0.08
Discount Factor	0.95	0.90
Exploration Rate	1.00	1.00
Exploration Decay	0.9994	0.9999
Minimum Exploration Rate	0.001	0.01

Q-Learning Algorithm

We set up and initialise Q-table to zero for the possible actions, three or four depending on the environment. During training, it chooses actions based on an exploration strategy that balances random exploration and exploiting known rewards. For each episode, Pixel Piero receives feedback in the form of rewards, which he uses to update the Q-table and improve its decision-making for future.

Double Q-Learning Algorithm

We enhance the standard Q-learning algorithm by aiming to reduce overestimation bias in action value estimates encountered during standard Q-learning algorithm. We set up and initialise two separate Q-tables which is updated alternately during training. When choosing an action, Pixel Piero combines the values from both Q-tables to determine the best action based on the sum of their estimates. During the update phase, he randomly selects any Q-table for updating while utilising the other table to calculate the target value.

Triple Q-Learning Algorithm

Similarly, we implement Triple Q-Learning algorithm which builds upon the principles of standard Q-learning and aims to improve action value estimates by maintaining three separate Q-tables. This setup might mitigate overestimation bias during the learning process but may lead to significant computational costs. Similar to Double Q-Learning algorithm, Pixel Piero selects actions based on the combined values from all three Q-tables. During the update phase, he randomly chooses one of the three tables for updating while utilising the minimum value from the other two tables to calculate the target for the chosen table.

Quadruple Q-Learning Algorithm

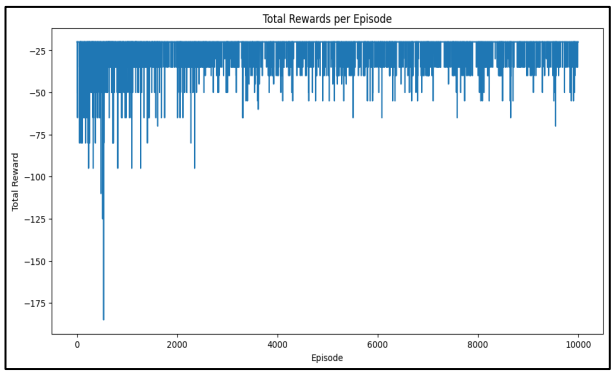
Similarly, we implement Quadruple Q-Learning algorithm, which enhances standard Q-learning algorithm by utilizing four separate Q-tables. This approach aims to reduce overestimation bias in action value estimates by updating each Q-table based on the minimum estimated values from the other three tables but might lead to significant computational costs. Pixel Piero chooses actions based on the combined Q-values of all four tables, and during the update phase, it randomly selects one of the tables to update.

Results

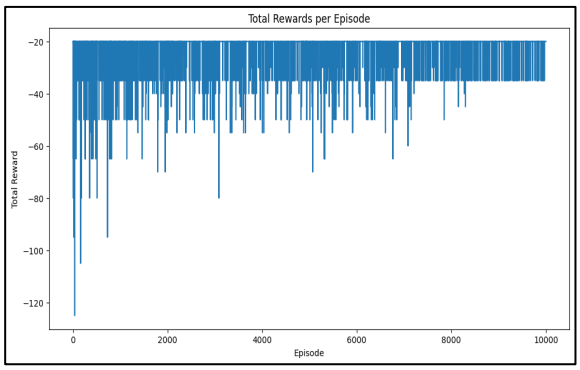
Beginner Quest Environment

We ran the variations of Q-Learning algorithm for 10000 episodes in Beginner Quest Environment and obtained the following graphs and results:

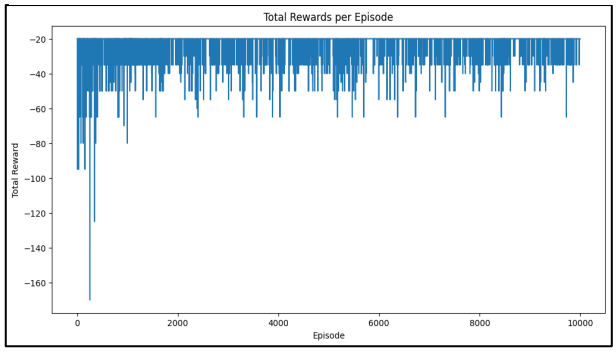
Total Rewards per Episode



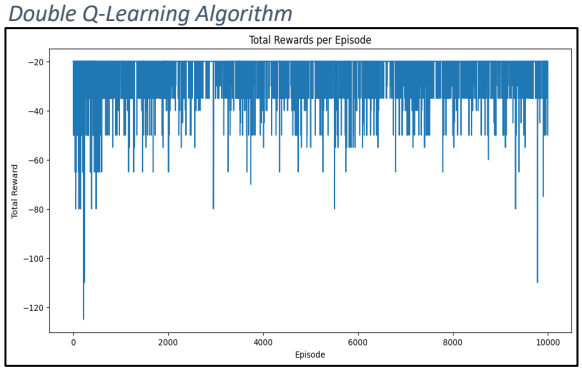
Standard Q-Learning Algorithm



Double Q-Learning Algorithm

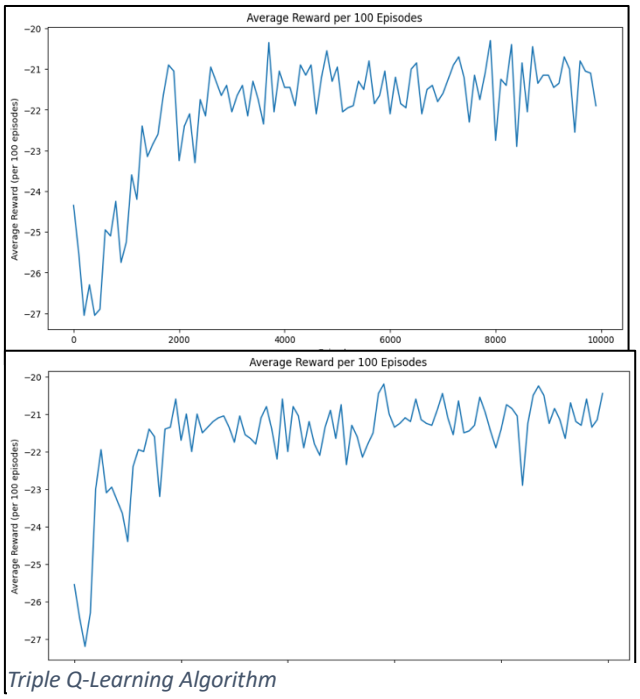


Triple Q-Learning Algorithm

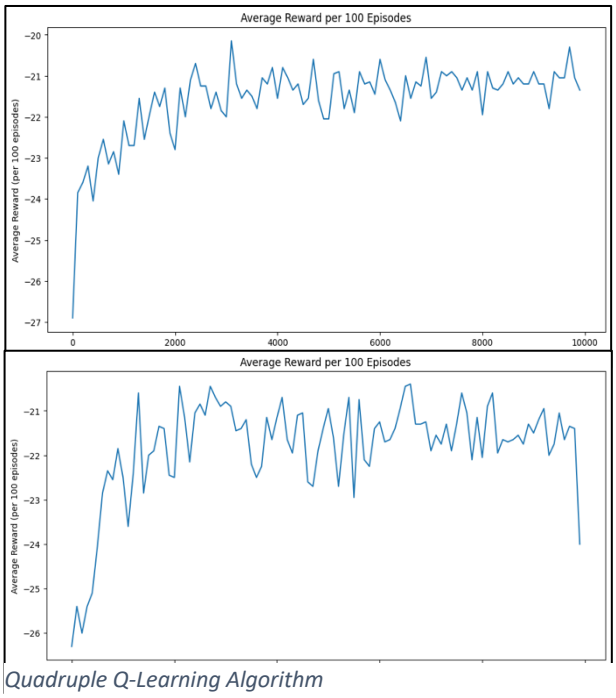


Quadruple Q-Learning Algorithm

Average Rewards per Episode



Triple Q-Learning Algorithm



Quadruple Q-Learning Algorithm

Algorithm	Final Total Reward	Average Reward
Standard Q-Learning	-20	-22.0475
Double Q-Learning	-20	-21.5890
Triple Q-Learning	-20	-21.6280
Quadruple Q-Learning	-20	-21.8105

Superior Quest Environment

We ran the variations of Q-Learning algorithm for 100000, 200000 and 1000000 episodes for Superior Quest Environment and obtained the following results:

100000 episodes

Algorithm	Final Total Reward	Average Reward	Training Time (s)
Standard Q-Learning	475	343.43073	103.51
Double Q-Learning	473	340.68952	120.77
Triple Q-Learning	476	333.24044	146.91
Quadruple Q-Learning	475	330.67928	157.58

200000 episodes

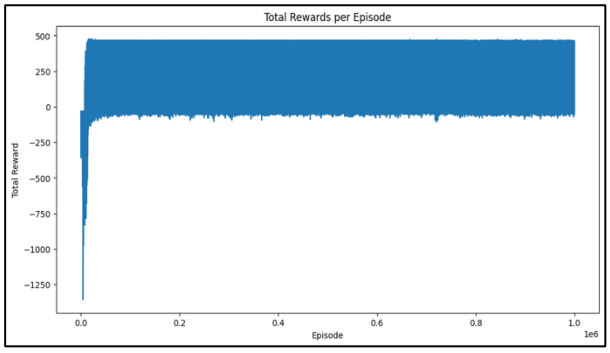
Algorithm	Final Total Reward	Average Reward	Training Time (s)
Standard Q-Learning	469	396.011925	237.06
Double Q-Learning	476	397.865155	260.47
Triple Q-Learning	476	395.252725	303.45
Quadruple Q-Learning	471	391.14675	345.36

1000000 episodes

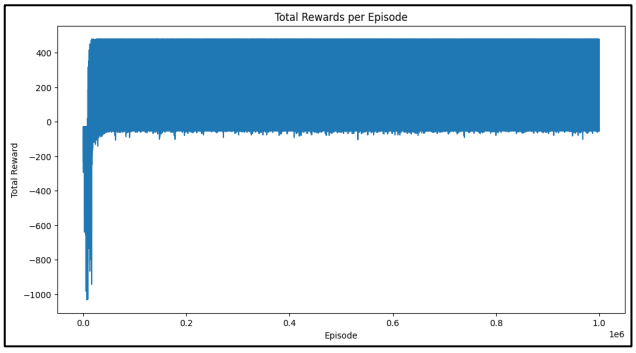
Algorithm	Final Total Reward	Average Reward	Training Time (s)
Standard Q-Learning	455	427.987424	1221.36
Double Q-Learning	477	444.867258	1437.92
Triple Q-Learning	462	429.116813	1669.12
Quadruple Q-Learning	470	436.996807	1810.57

The graphs obtained for 1000000 episodes have been show below. Similar graphs were plotted for 100000 and 200000 episodes, which are in their respective python notebooks.

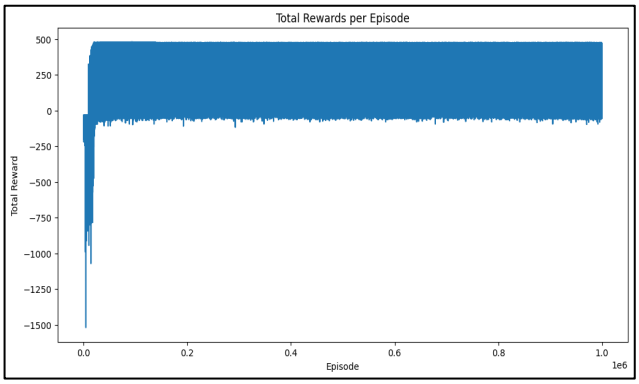
Total Rewards per Episode (1000000 episodes)



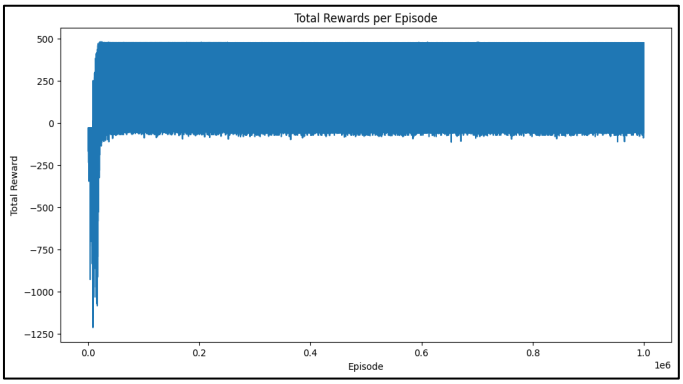
Standard Q-Learning Algorithm



Double Q-Learning Algorithm

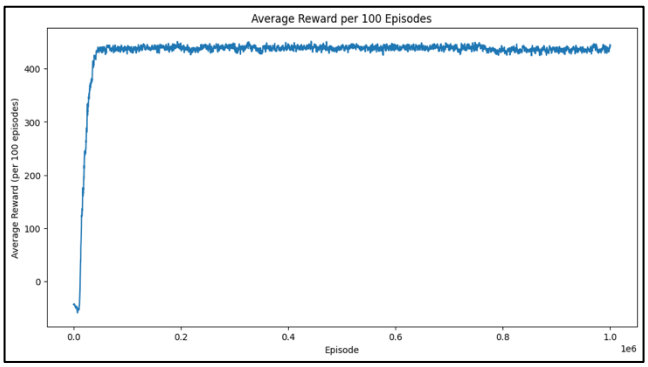


Triple Q-Learning Algorithm

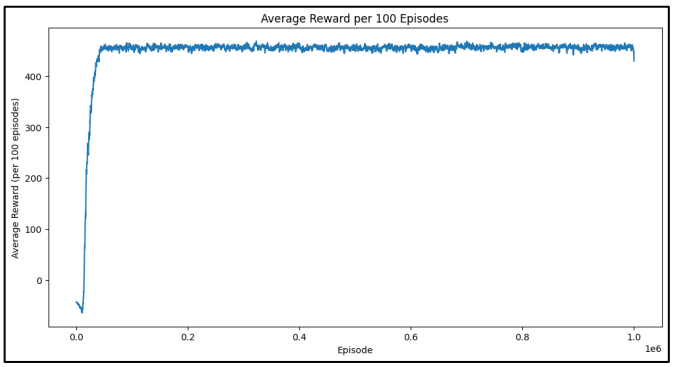


Quadruple Q-Learning Algorithm

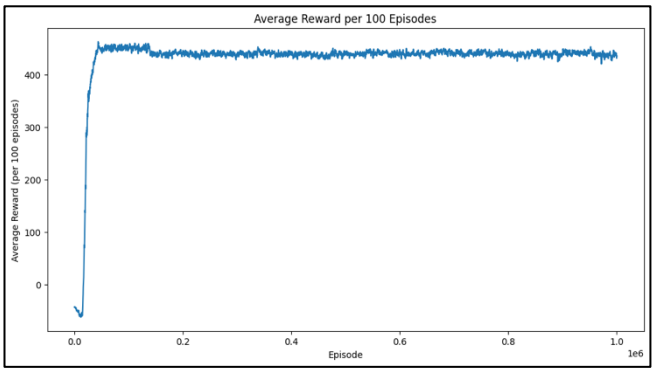
Average Rewards per Episode (1000000 episodes)



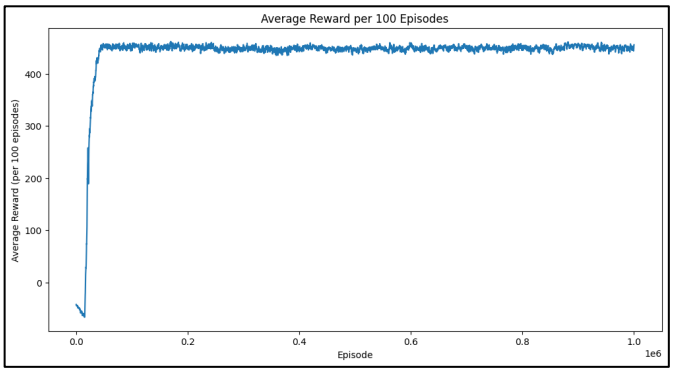
Standard Q-Learning Algorithm



Double Q-Learning Algorithm



Triple Q-Learning Algorithm



Quadruple Q-Learning Algorithm

Comparative Analysis

For Beginner Quest Environment, the results demonstrated that all four algorithms achieved same final total reward, indicating that none were able to secure a positive outcome within the environment given. The average rewards varied slightly across the approaches, with Double Q-Learning exhibiting the highest average rewards, but, the overall difference in average reward accumulation remained minimal across all algorithms. These results suggest that for a simplistic environment, advanced variations of Q learning algorithm may not provide significant advantage over standard Q-learning algorithm.

Hence, we move on to upgraded Superior Quest Environment for better analysis of the implemented algorithms. We analysed the performance of variations of Q-Learning algorithms across three distinct training durations within the Superior Quest Environment.

The results mentioned above indicate that all algorithms exhibit improved performance metrics with an increase in the number of training episodes. Notably, Double Q-Learning consistently achieved the highest average reward across all episode counts. Also, the training times increased significantly as we increased the episode count and moved from standard Q-Learning to Quadruple Q-Learning algorithms.

Conclusion

A comparison of multiple Q-learning algorithms found that Double Q-Learning regularly outperforms other techniques. This is due to Double Q-Learning algorithm's ability to reduce overestimation bias by keeping two different Q-tables for action-value pairings. The Double Q-Learning method selects actions based on one value function and updates them with the other, hence decreasing the overestimation bias inherent in the regular Q-Learning algorithm, resulting in more accurate and consistent learning results. Furthermore, Double Q-Learning achieves an ideal mix of exploration and exploitation, which is essential for effective learning. Its dual value estimations enable sufficient exploration without excessively favouring known values, hence improving overall performance by selecting optimum actions.

In contrast, we expected the Triple and Quadruple Q-Learning algorithms to outperform the regular Q-Learning and Double Q-Learning algorithms; instead they underperformed. This might be due to the increasing complexity of these algorithms, which may have disturbed the balance between exploration and exploitation. As the number of value estimates grows, additional complexity might lead to lower performance returns. The increased complexity and quantity of different Q-tables may have caused instability during training, resulting in unsatisfactory results. The extra computational burden associated with Triple and Quadruple Q-Learning may further reduce their efficiency. The time and resources necessary to maintain various value estimates could limit their ability to learn effectively in a certain amount of time.

In conclusion, Double Q-Learning remains the best choice among the four algorithms for the environment because of its ability to reduce overestimation bias, balance exploration and exploitation, and maintain training stability. The underperformance of Triple and Quadruple Q-Learning emphasises the significance of choosing algorithms that are suitable for the environment's specific challenges, as well as the fact that increased complexity does not always translate to higher performance.

Choosing between versions of Q-Learning algorithms in a CLIF-style environment, such as Q-Quest, requires a better knowledge of aspects such as the exploration-exploitation trade-off and the need to improve learning efficiency without incurring excessive computational costs. By carefully choosing the most suitable technique, we may optimise Pixel Piero's gaming, allowing him to choose optimal actions at each state, leading to overall higher rewards.