RL Snake Variant Project (Decaying Fruits)

Vidun Jayakody, Muntaha Attef, Jeremy Tran Perez, Yuvraj Deol

Date: December 8

Professor: Junfeng Wen

1 Introduction

This project focuses on reinforcement learning in a closed, dynamic environment, as exemplified by a modified version of the classic Snake game. In this environment, defined by a 10x10 grid, the fruits consumed by the snake are subject to decay, introducing a unique mechanic where the consumption of rotten fruits results in a penalty instead of a reward. However, the environment has an additional element that rotten fruits eventually respawn as fresh fruits after a certain period of time. This dynamic forces the agent to make strategic decisions that balance short-term gains against potential losses. The agent must decide not only whether to consume a fruit, but also whether it is worth waiting for some fruit to turn fresh again.

This adaptation of the Snake game introduces this added challenge which is representative of real-world decision-making scenarios, where agents must optimize their behaviour under time-sensitive conditions and dynamic state transitions. Unlike the original Snake game, which rewards any food consumption, our environment incorporates a risk-reward trade-off that urges the agent to avoid decayed fruit, by running in loops without hitting any walls or its tail, to maximize positive reward. Furthermore, the smaller grid size constrains the agent's navigation skills and adds the component of complex space planning to the task.

In this project, our objective is to explore how reinforcement learning algorithms can be used effectively to address these challenges, with the goal of developing an agent that maximizes its growth while navigating the complexities of temporal constraints, risk-reward trade-offs, and dynamic decision-making. The project addresses fundamental challenges in reinforcement learning for dynamic environments. The problem of fruit decay and respawn introduces important elements of timing, decision-making, and risk assessment, all of which are essential concepts in many reinforcement learning applications.

2 Approaches

2.1 A brief overview of previous works

Autonomous Agents In Snake Game via Deep Reinforcement Learning [1]

This work refines the Deep Q-Network (DQN) model for the Snake Game by addressing the issue of sparse and delayed rewards using a custom reward mechanism. The mechanism consists of three key components: Distance Reward, Training Gap, and Timeout Strategy. The Distance Reward incentivizes the agent to move towards the target. The Training Gap prevents the agent from learning improper experiences right after eating an apple, conceptually similar to a physiological refractory period. Lastly, the timeout strategy penalizes the agent if it fails to eat an apple in a given number of steps. With the combination of these three components, the custom reward mechanism effectively addresses the limitations of the original DQL model, resulting in a refined model that outperforms it.

A Deep Q-Learning based approach applied to the Snake game [2]

This paper presents a Deep Q-Learning (DQL) approach for training an agent to play the Snake game. The proposed approach uses sensor measurements over image-based inputs, simplifying the neural network architecture and training process. The agent is built as a 5-layer fully connected network with ReLU activations and Dropout layers for regularization. Training involves the exploration-exploitation trade-off, backpropagation, the Adam optimizer, and the Mean Square Error (MSE) as the loss function. The study concludes that the proposed DQN agent effectively learns to play the Snake game using sensor data, resulting in simpler and faster training.

A Multi-Agent Actor-Critic Based Approach to the Snake Game [3]

This study investigates the performance of multi-agent Q-learning and multi-agent Actor Critic reinforcement learning methods in a competitive environment where multiple agents interact simultaneously on a shared platform. The research addresses challenges such as state representation, sparse rewards, and teamwork coordination, which are critical to interactive scenarios. The study concludes that while Q-learning can quickly converge in single agent scenarios, it struggles in multi-agent environments. In contrast, the Actor-Critic approach enables agents to make more informed decisions by better anticipating future risks, resulting in greater stability and better teamwork coordination.

Optimization Strategies for Atari Game Environments [4]

This paper presents the Energy Serpent Optimizer (ESO) algorithm, a hybrid combination of the Snake Optimization Algorithm (SOA) and Energy Valley Optimization (EVO) to improve hyperparameter tuning for RL agents. The main objective is to improve the agent's performance in a maze-based environment inspired by Ms. Pac-Man. When tested, ESO demonstrated superior performance compared to SOA and EVO, achieving the highest rewards and faster convergence rates. This innovative approach demonstrated the potential of combining various algorithms as a means of creating more efficient RL models in complex and dynamic environments.

2.2 Overview of Challenges

The first challenge was building the Snake game from scratch to ensure compatibility with gymnasium. This required careful design of mechanics like grid representation, dynamic fruit states (fresh and decayed), and the snake's movement and collision handling. Iterative adjustments to the reward structure, grid size, and fruit decay timing were needed to create a balanced environment. Additionally, designing an observation space that effectively represented the game state while remaining computationally efficient added complexity.

Choosing a learning algorithm was another challenge. The environment's dynamic rewards and states required balancing exploration and exploitation. Q-learning was selected for its simplicity, compatibility with discrete state-action spaces, and suitability for grid-based tasks. Although advanced algorithms like DQN could handle complex observation spaces, the project's discrete state space aligned better with Q-learning, making it a practical starting point.

Implementing Q-learning revealed key challenges. The agent initially exhibited suboptimal behaviors, such as exploiting static positions, avoiding walls regardless of fruit location, and consuming fruit without considering decay. The state representation inadvertently encouraged static strategies, with the agent favoring "safe zones" and avoiding edges to minimize risk. These behaviors detracted from timely fruit collection. To address this, we increased the reward for eating fruit and raised the discount factor γ to emphasize future rewards, encouraging the agent to prioritize long-term goals.

Implementing DQN introduced further challenges. Sparse and dynamic rewards led to stagnant or overly cautious behavior, with the agent avoiding edges and walls but failing to pursue fruit effectively. Adjustments to the reward structure and γ slightly improved performance, but navigation strategies remained inefficient. Due to the computational complexity, sensitivity to hyperparameters, and limited improvement, DQN was abandoned in favor of Q-learning, which better suited the Snake game's discrete state space.

2.3 Approach 1: Original Q-learning

The initial approach utilized a basic Q-learning algorithm with iterative adjustments to both the algorithm and the environment. Epsilon decay was accelerated to shorten the exploration phase, encouraging earlier exploitation of learned strategies and reducing the agent's reliance on static patterns. The reward structure was also modified: the reward for consuming fresh fruits was increased, and a small penalty was added for fruit respawning, fostering immediate fruit collection and discouraging passive behaviors.

Minor adjustments were made to fruit decay parameters, extending decay time to prevent accidental respawning. This was intended to address environmental inconsistencies rather than agent behavior. Algorithmic changes remained the primary focus, as the observed issues were strategy-related rather than environment-driven.

Alternative methods were explored, such as extending fruit lifetimes to reduce respawning frequency and removing grid walls to allow edge wrapping. These were excluded to maintain Snake's classic mechanics and due to the algorithmic nature of the issues.

The largest challenge with Q-learning was the inability to generalize across the vast state space of a 40x40 grid. The discrete state-action mapping struggled with convergence, particularly with decaying fruits. This limitation led to transitioning to Deep Q-learning, which leverages neural networks to approximate Q-values and efficiently handle large state spaces.

2.4 Approach 2: DQN

The second approach involved implementing Deep Q-Learning (DQN) to address the limitations of the initial Q-learning agent. DQN uses a neural network to approximate the Q-value function, allowing it to handle complex state representations. To start, we defined a state representation that could effectively capture the game's dynamics. As the Snake game has a grid-based representation, as well as the fruit's position (fresh and decayed), we fed this input into the neural network to allow for spatial and contextual

(for the fruit) features.

Initially, the agent was very cautious and prioritized survival by avoiding walls and stalling in the center without going after the rewards. To fix this, we introduced a slower epsilon decay, allowing for the agent to explore the environment before attempting to exploit. This adjustment was aimed to balance the tradeoff between exploration and exploitation which would allow the agent to develop more reward-focused strategies. We also observed that the original reward structure, that focused mainly on consuming the fruit and avoiding walls, was limiting the agent. We adjusted to increase the reward for fresh fruit collection by giving a higher reward for lower steps and penalties for fruits after a certain number of steps which assumed that more steps before collecting the fruit would be negative, rather than the previous goal of just collecting fruit. We also planned to add penalties for idling away for rewards to discourage overly passive behaviours.

We considered exploring a couple of alternative strategies as well but ultimately discarded the ideas. Maybe if the fruit's decay rate and total lifetime was slowed and delayed, it would allow for the algorithm to learn at a better rate, however, it does not address the underlying algorithmic challenges. We also thought to increase penalties for not collecting fruit at all (a penalty for when the fruit would despawn) but it would lead to unintended behaviours as opposed to more balanced reward adjustments.

While the DQN approach had a theoretically more powerful framework, the additional complexity, sensitivity to hyperparameters, as well as other issues made it less suitable for the relatively small and discrete state-action space of the Snake game. The final DQN agent before it was scrapped incorporated a slower epsilon decay, refined reward structure, and a more robust neural network architecture. However, performance plateaued and it failed to outperform a simple Q-learning agent in terms of reward maximization and computational efficiency. As a result, we deferred to the Q-learning approach, allowing for a more practical and reliable solution for this specific environment.

2.5 Approach 3: Final Q-Learning

Reverting to the Q-learning approach, we learned from our mistakes of making the agent work with the environment and instead went back to the drawing board to figure out how to balance efficiency and performance with the limited resources we had. The solution was simple, make the grid smaller. The original iteration of the Snake game was on a 40x40 square grid which we reduced in favour of a 10x10 square grid. This small change allowed us to get results across thousands of episodes within a reasonable time frame (hours rather than days for thousands of episodes). With the smaller grid came a larger emphasis on proper space planning from the agent's end. We faced the challenge of the agent moving in circles and hoping it gets a fruit and since the grid was small, the snake would eventually get to it within the time frame. This resulted in the final iteration of increasing the epsilon decay to place a large emphasis on the fast collection of the fruit.

The final iteration of the agent involved balancing exploration and decay parameters more carefully. Lower initial epsilon values, combined with slower decay rates for epsilon and fruit respawning, produced an agent that demonstrated improved strategies for collecting fruits and minimized passive behaviours. Through these experiments,

it became evident that the interplay between the algorithm's hyperparameters and the environment's reward structure was critical in shaping the agent's behaviour. These findings reinforced the importance of iterative refinement in reinforcement learning tasks and highlighted the need for thoughtful design when aligning algorithmic behaviour with task objectives which concludes the timelines for all approaches taken to solve the problem of the decaying Snake game.

3 Empirical Studies

The experiments conducted in this project aimed to evaluate the effectiveness of Q-learning in training an agent to play the modified Snake game. The agent was tested under different training runs of 5,000, 10,000, and 50,000 episodes. Each run was analyzed based on key performance metrics that showcased good gameplay. They were total reward, survival time, and the number of fruits collected. We begin with a definition of the experimental setup which includes the state space, action space, reward structure, and methodology for initializing the Q-table, followed by a discussion of the results and observations.

3.1 State Space

The state space represents the agent's current environment, which includes:

- The position of the snake's head is represented as (`snake_x_cell`, `snake_y_cell`) coordinate pair on the grid.
- The position of the fruit is represented as (`fruit_x_cell`, `fruit_y_cell`).
- The direction that the snake is moving is represented as `dir_idx`.
- Together these elements form a tuple that uniquely identified each state (`snake_x_cell`, `snake_y_cell`, `fruit_x_cell`, `fruit_y_cell`, `dir_idx`). For a 10×10 grid, there results in a total of $10 \times 10 \times 10 \times 10 \times 4 = 40,000$ possible states.

3.2 Action Space

The action space is simple, it consists of the four discrete actions the agent can take which are move up, move down, move left, or move right. These actions allow the agent to navigate the grid while aiming to collect fruits and avoid collisions with the walls and itself.

3.3 Reward Structure

The reward structure was designed to encourage the agent to eat fresh fruits and discourage consuming decayed fruits or colliding with the walls or itself. The reward structure was designed as such:

- The reward for eating fresh fruit is +100
- The penalty for eating decayed fruit is -5

- The small penalty for each step taken to increase efficiency is -0.1
- The penalty for collision or game termination is -10

This structure was built on a balance between exploration, survival, and fruit collection.

3.4 Initialization of the Q-table

The Q-table is a two-dimensional matrix whose rows represent all possible states and columns represent the four possible actions. The table was initialized such that there were 40,000 rows for each state, 4 columns for each action and its initial values were 0 since it allowed the agent to learn the optimal Q values during training.

3.5 Results

3.5.1 5,000 episodes

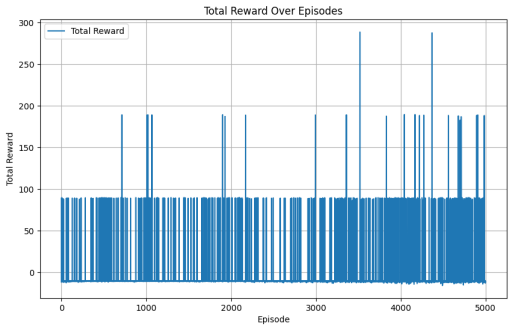

Figure 1: Graph of total reward earned in each of the 5,000 episodes

The total reward during this run showed significant variability, as seen in the graph. The agent struggled to develop a consistent strategy, frequently incurring penalties for eating decayed fruits or collisions. This was primarily due to the short training duration, which limited the exploration of the state space. Episodes with higher rewards were sporadic and likely due to random exploration rather than learned behaviour.

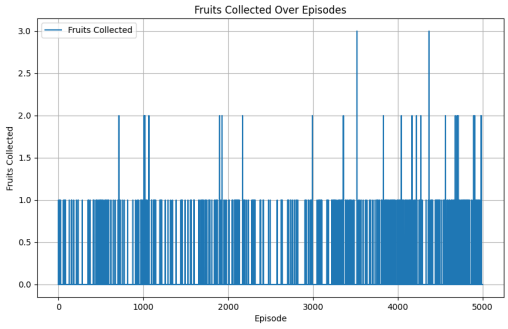

Figure 2: Graph of total fruits collected in each of the 5,000 episodes

The fruits collected remained modest throughout, rarely exceeding one fruit per episode. The agent often failed to navigate efficiently towards fresh fruits, instead opting to avoid risks by staying in safer areas of the grid. This behaviour reflects the agent's incomplete understanding of the reward structure at this stage.

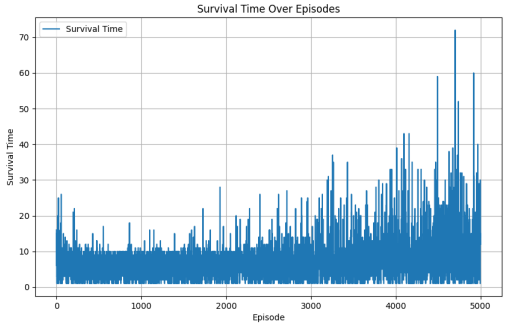

Figure 3: Graph of survival time in seconds of the agent in each of the 5,000 episodes

Survival times were already relatively short, with the agent frequently colliding with walls or itself. The graph demonstrates that the agent often prioritized staying in the center of the grid resulting in suboptimal movement and early termination of episodes.

3.5.2 10,000 episodes

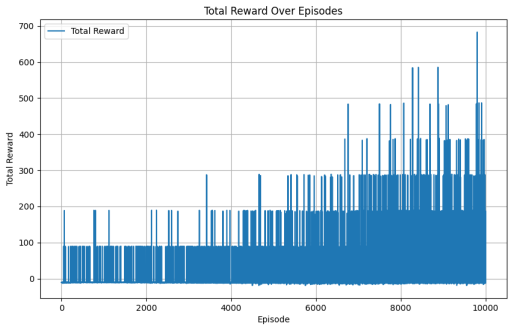

Figure 4: Graph of total reward earned in each of the 10,000 episodes

The reward curve displayed a steady upward trend, indicating the agent's increasing ability to balance exploration and exploitation. By this stage, the agent had started to converge on a policy that better aligned with the reward structure. Episodes with higher rewards became more frequent, showcasing an improved ability to prioritize fresh fruits.

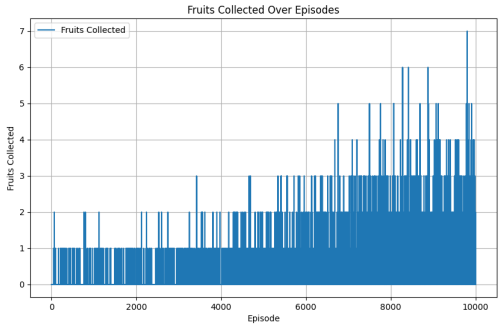

Figure 5: Graph of total fruits collected in each of the 5,000 episodes

The fruits collected metric improved noticeably compared to the 5,000-episode run. The agent was now able to collect multiple fruits in some episodes, as it learned to navigate toward fruits more effectively. Adjustments to the epsilon decay rate were impactful, allowing the agent to transition from exploring to exploitation more effectively.

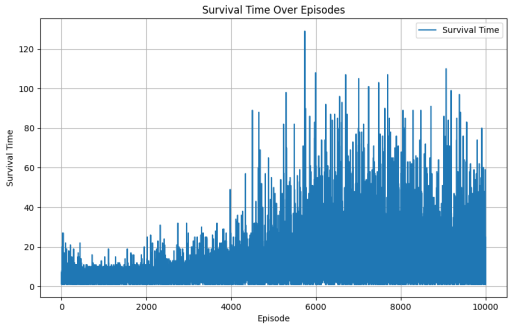

Figure 6: Graph of survival time in seconds of the agent in each of the 10,000 episodes

Average survival time increased significantly during this run. The agent demonstrated a better understanding of avoiding collisions, with fewer premature terminations. However, the survival time was still limited by occasional lapses in strategy when navigating the grid.

3.5.3 50,000 Episodes

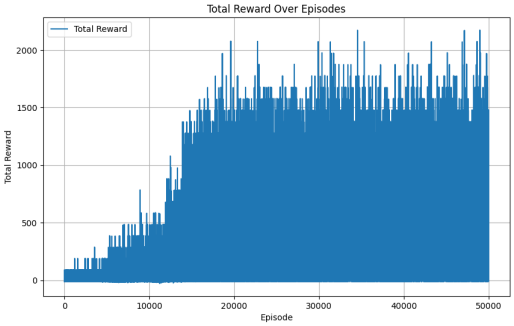

Figure 7: Graph of total reward earned in each of the 50,000 episodes

By the end of 50,000 episodes, the total reward plateaued, indicating convergence in the Q-table. The agent consistently achieved higher rewards by efficiently collecting fresh fruits and avoiding decayed ones. The stability in the reward graph suggests that the agent had optimized its strategy within the given environment.

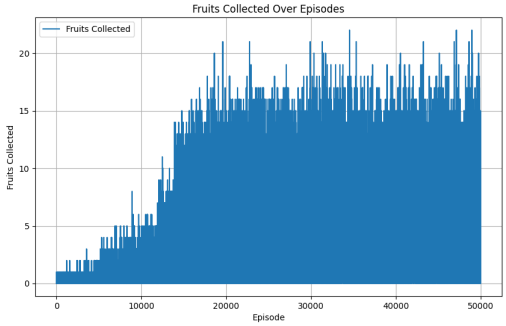

Figure 8: Graph of total fruits collected in each of the 50,000 episodes

Fruit collection rates peaked during this iteration, with the agent frequently collecting 10 or more fruits in a single episode. This demonstrates the agent's mastery of navigating the grid and prioritizing fresh fruits over avoiding risks.

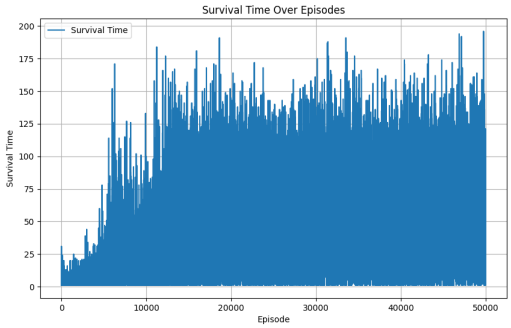

Figure 9: Graph of survival time in seconds of the agent in each of the 50,000 episodes

Survival times became highly stable and often reached the upper limit of 1,000 steps per episode. The agent successfully avoided collisions and adapted its movement to maximize survival while pursuing rewards.

3.5.4 Discussion

The results illustrate a clear learning curve across the three iterations, with steady improvements in all three metrics. The learning behaviour of the agent during the original 5,000 episodes was that it relied heavily on exploration, which resulted in erratic behaviour and inconsistent rewards. This phase highlighted the importance of a sufficiently long training duration to allow the Q-table to converge. During the 10,000-episode run, a turning point occurred as the agent began exploiting its learned policy which led to measurable improvements in all metrics mentioned above. Once we ran it for 50,000 episodes, the agent demonstrated near-optimal performance. Of course, it will never be perfect to train it on an infinitely ongoing game with the limited resources we have, however, it

successfully navigates the environment and maximizes its rewards.

Adjustments to the epsilon decay rate played a critical role in transitioning the agent from exploration to exploitation. Faster decay rates in earlier iterations led to premature convergence, while slower rates allowed for better policy refinement during the later runs. The learning rate (α=5) was well suited to our problem, which enabled the agent to adapt quickly while maintaining stability in Q-value updates. Furthermore, the reward structure effectively guided the agent's behaviour and the discrete state-action space.

4 Conclusion

This project explored reinforcement learning in a dynamic environment through a modified Snake game. The iterative development highlighted the importance of state representation, reward structure, and hyperparameter tuning in shaping agent behaviour. Adjustments to the reward structure and exploration parameters signification improved the agent's performance guiding it toward more optimal strategies

While Deep Q-learning offered a more powerful framework, its computational complexity and sensitivity to hyperparameters make Q-learning the better choice for this discrete environment. Simplifications, such as reducing the grid size enabled effective training and highlighted the trade-offs between computational efficiency and scalability.

Key lessons include the importance of aligning environment design with takes objectives and iteratively refining parameters to encourage desired behaviours. Future improvements could focus on using relative state representations, implementing DQN for larger grids, and introducing complex reward structures such as increased rewards for eating fruits and combination points for eating consecutive fruits. This project demonstrates the effectiveness of Q-learning for small-scale tasks while identifying its limitations in scaling to more complex scenarios.

References

[1] "Autonomous Agents in Snake Game via Deep Reinforcement Learning." IEEE Conference Publication — IEEE Xplore, 1 July 2018. Available at: https://ieeexplore.ieee.org/document/8460004.

[2] "A Deep Q-Learning Based Approach Applied to the Snake Game." IEEE Conference Publication — IEEE Xplore, 22 June 2021. Available at: https://ieeexplore.ieee.org/document/9480232.

[3] "A Multi-Agent Actor-Critic Based Approach Applied to the Snake Game." IEEE Conference Publication — IEEE Xplore, 24 July 2023. Available at: https://ieeexplore.ieee.org/document/10241182.

[4] Sarkhi, Sadeq Mohammed Kadhm, and Hakan Koyuncu. "Optimization Strategies for Atari Game Environments: Integrating Snake Optimization Algorithm and Energy Valley Optimization in Reinforcement Learning Models." AI, vol. 5, no. 3, July 2024, pp. 1172–91. Available at: https://doi.org/10.3390/ai5030057.