CS 329 - Foundations of AI - Multiagent Systems

Project

Aeshaa Nehal Shah - 23110018
Harinarayan J - 23110128
Vyomika Vasireddy - 23110363

Snake Game

Introduction:

In this project, we develop a Reinforcement Learning (RL) agent capable of playing the classic Snake game, with the goal of evaluating how different RL methods perform under varying game dynamics and reward structures. We selected this game due to its simplicity and the variability for different changes.

To explore these effects, we compared multiple RL algorithms with two different reward schemes.

Reward Schemes:

- Simple Reward Scheme:

  In this baseline approach, the agent receives a reward of +1 for successfully eating food and –1 upon game termination (e.g., collision). No intermediate rewards or penalties are applied. This scheme is commonly used for its simplicity and for reducing noise in the reward signal.

- Scaled Reward Scheme (Length-Dependent Rewards):

  In the second approach, the same positive and negative rewards are used, but they are scaled by a factor proportional to the snake's current progress. Specifically, the reward is:

  Reward: ±1×(length of snake)

  As the snake grows, the magnitude of both positive and negative rewards increases. This amplifies the incentive for food collection while also penalizing late-game mistakes more heavily.

RL Methods Used:

A. Q - Learning:

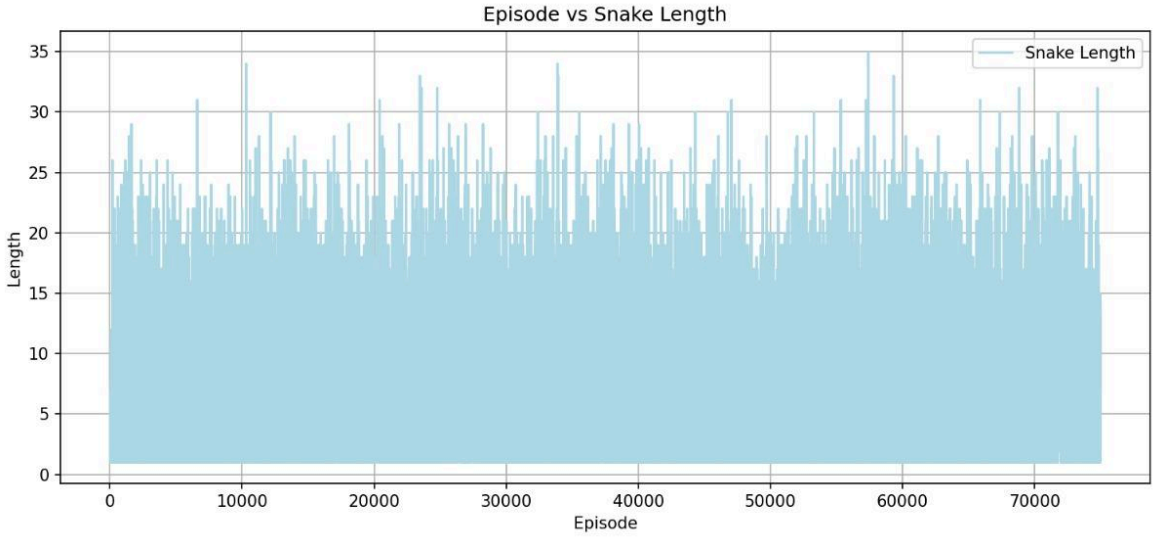
Q-Learning is an off-policy, value-based reinforcement learning algorithm that estimates the optimal action-value function Q*(s, a) using temporal-difference (TD) learning. The action-value function represents the expected discounted return obtained by taking action a in state s and thereafter following the optimal policy:

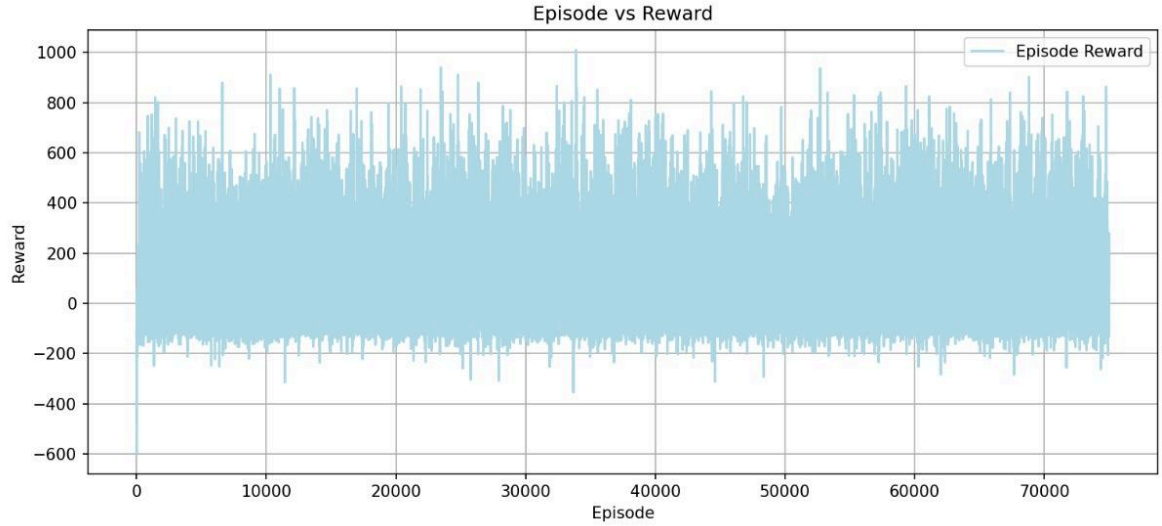
$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, s_0 = s,\ a_0 = a\right]$$

where rt is the reward at time t and $\gamma \in [0, 1]$ is the discount factor.

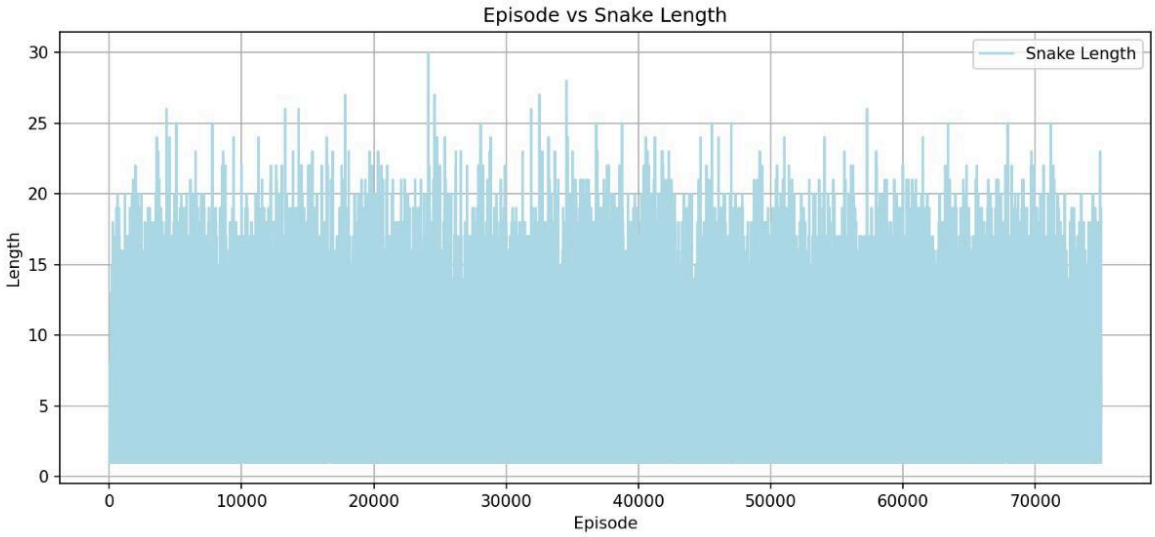
Results:

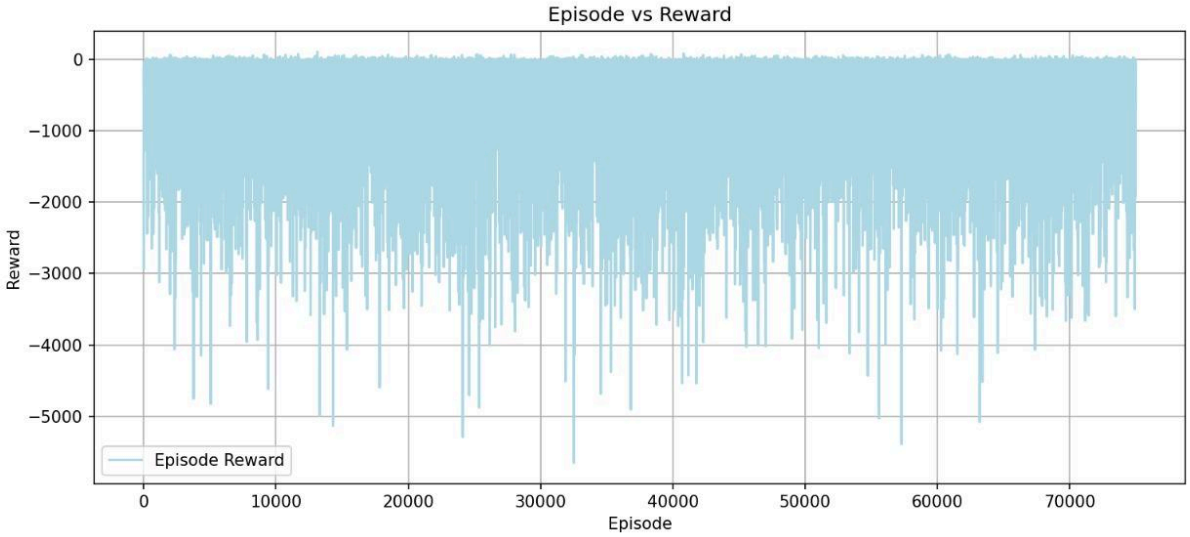
Reward Scheme 1:





Reward Scheme 2:


Episode vs Snake Length
Snake Length
Length
Episode
0
5
10
15
20
25
30
0
10000
20000
30000
40000
50000
60000
70000


Episode vs Reward
Episode Reward
Reward
Episode
0
−1000
−2000
−3000
−4000
−5000
0
10000
20000
30000
40000
50000
60000
70000

B. SARSA

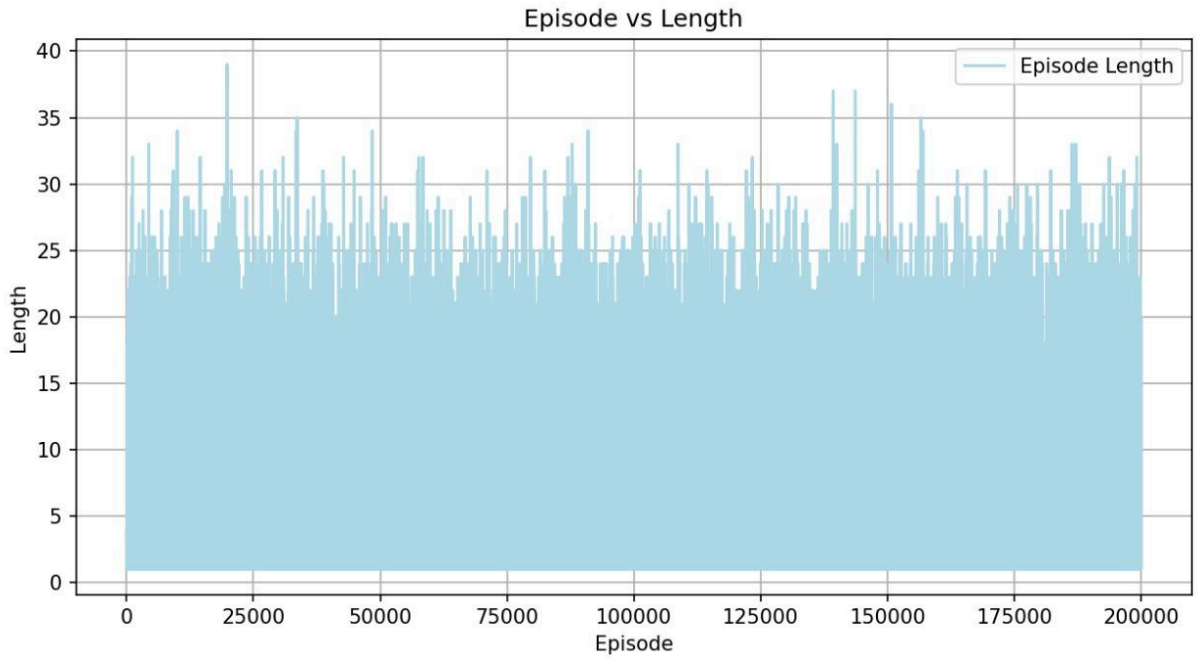
SARSA (State–Action–Reward–State–Action) is an on-policy temporal-difference (TD) control algorithm used to learn an action-value function Q(s, a) for a given Markov decision process. Unlike Q-Learning, SARSA evaluates and improves the policy that is actually being executed, making it an on-policy method.

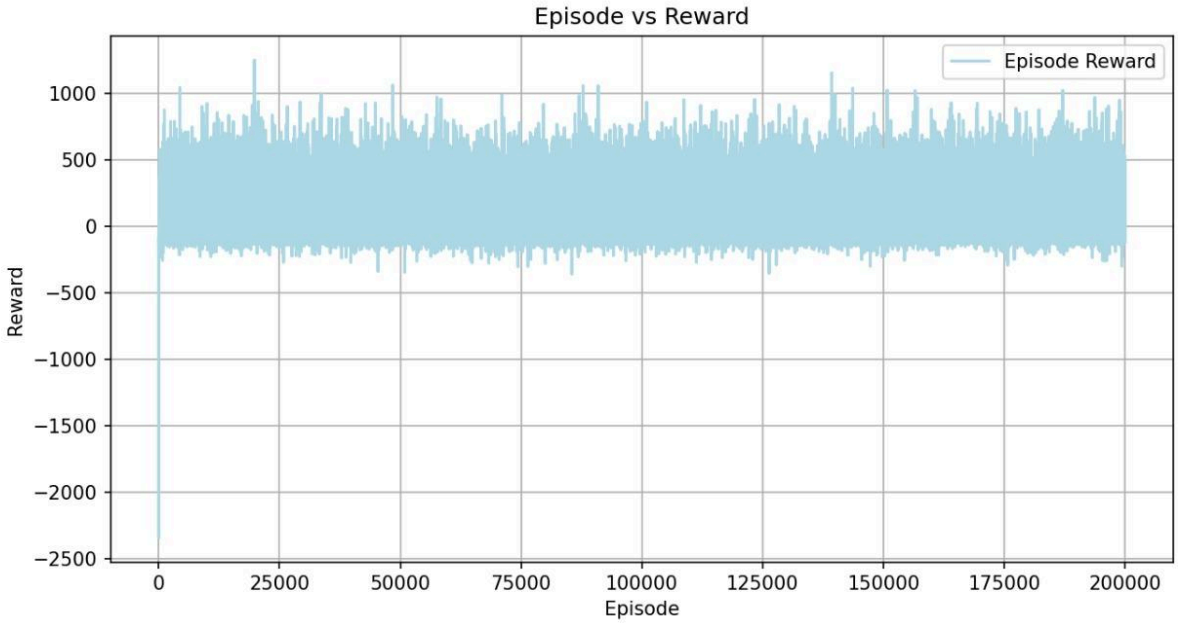
SARSA estimates the expected discounted return when following the current behaviour policy π:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \;\middle|\; s_0 = s,\ a_0 = a,\ a_t \sim \pi\right].$$

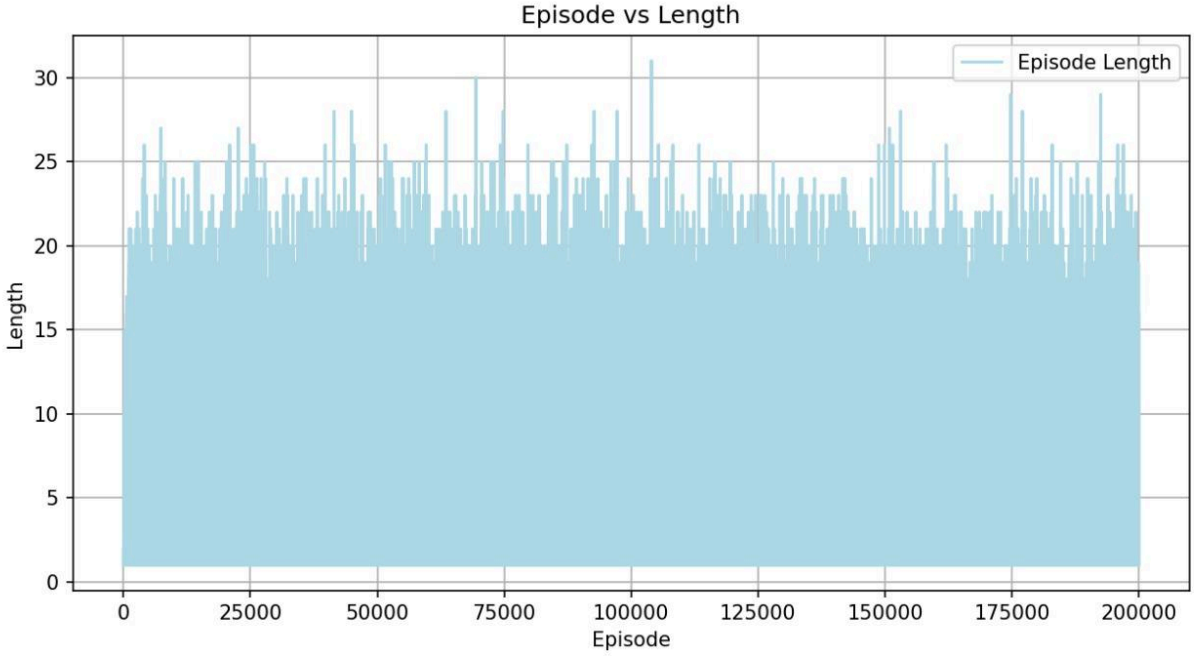
Results:

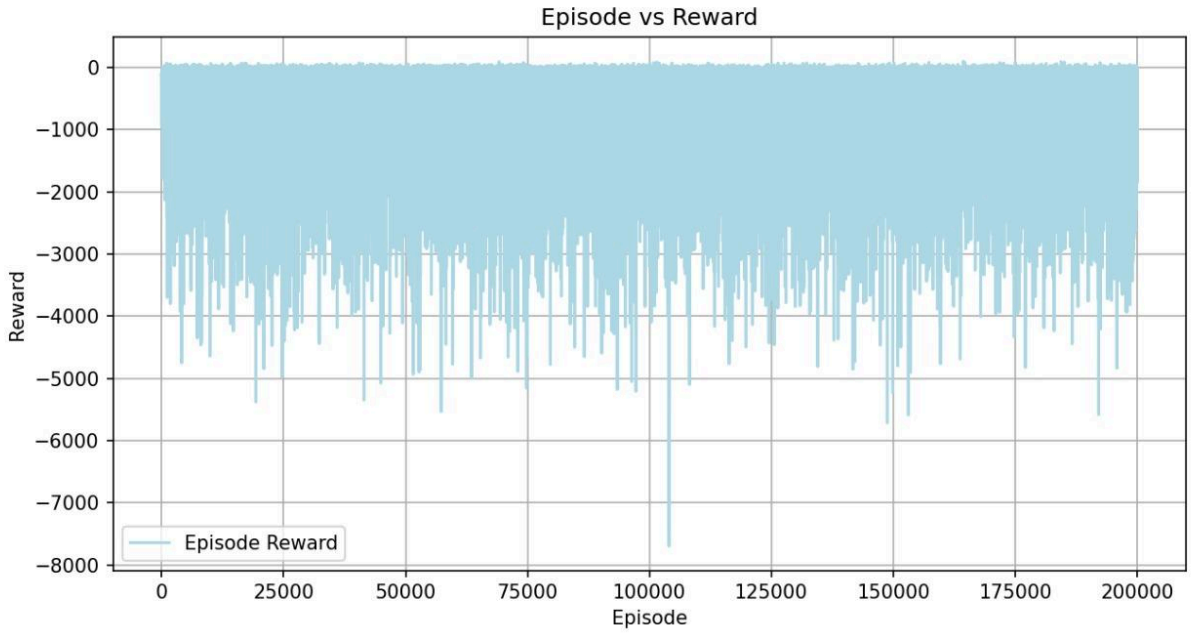
Reward Scheme 1:





Reward Scheme 2:





C. Deep Q-Network

Deep Q-Networks extend Q-learning by replacing the tabular Q-function with a neural network Qθ (s, a) that approximates action values directly from high-dimensional inputs.

DQN maintains two networks: an online network Qθ used for learning and action selection, and a target network Qθ− , which is periodically updated to stabilize training.

Action selection is performed using an ϵ-greedy strategy, where the agent chooses a random action with probability ϵ and otherwise selects the action with the highest predicted Q-value. The target for each transition (s, a, r, s′, d) is computed as

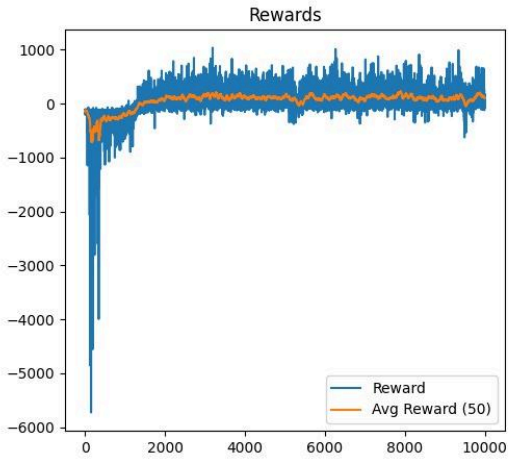
$$y = r + (1 - d)\,\gamma \max_{a'} Q_{\theta^-}(s', a'),$$

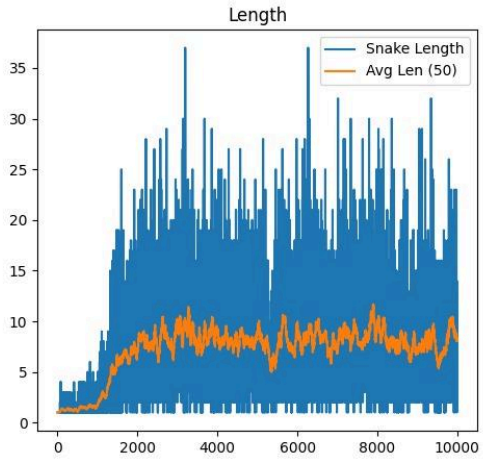
where γ is the discount factor and d indicates episode termination.

The online network is trained by minimizing the squared error between its prediction Qθ (s, a) and the target y. Gradient descent updates the network parameters to improve value estimates over time, while periodic updates of the target network θ− ← θ prevent divergence and improve stability.

Results:

Reward Scheme 1:





D. Actor Critic

The Actor–Critic algorithm combines policy-based and value-based learning into a single framework, where two separate components are learned simultaneously:

- Actor — a parameterized policy πθ (a|s) that selects actions.
- Critic — a value function Vw(s) that evaluates states and provides feedback to the actor.

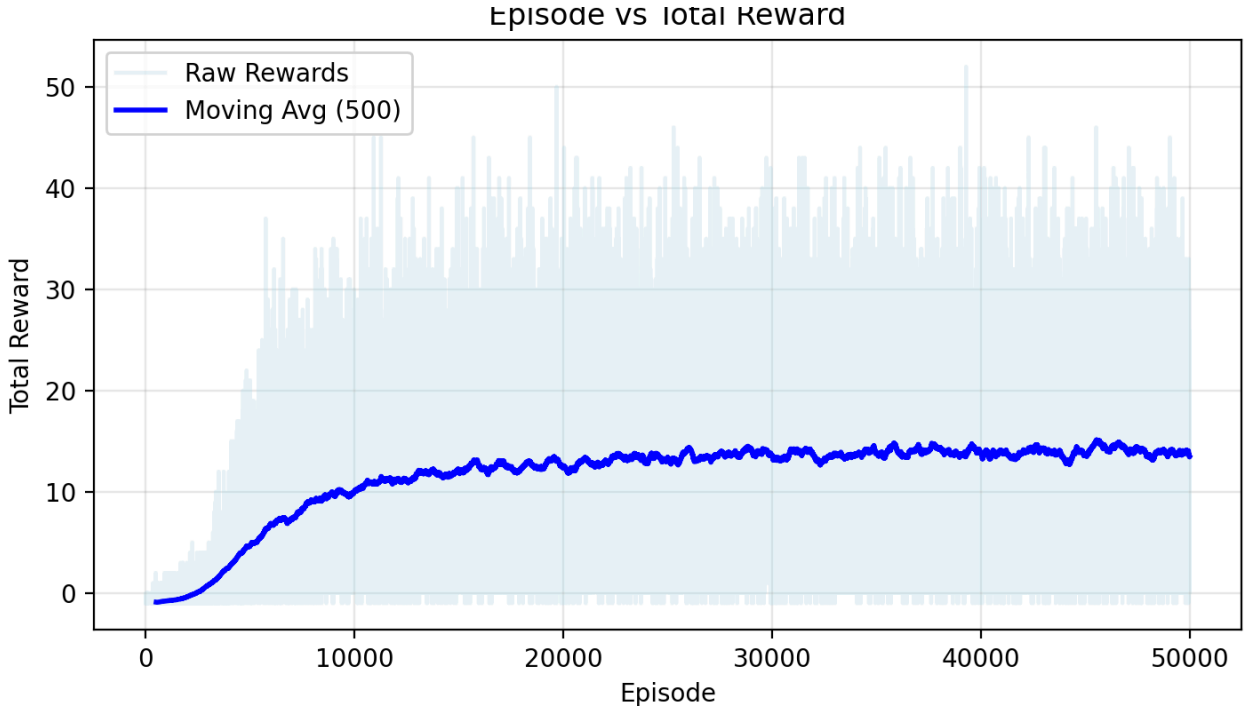
The actor outputs a probability distribution over actions using a softmax function. The critic learns to estimate the expected return of a state and provides the actor with a learning signal known as the temporal-difference (TD) error:

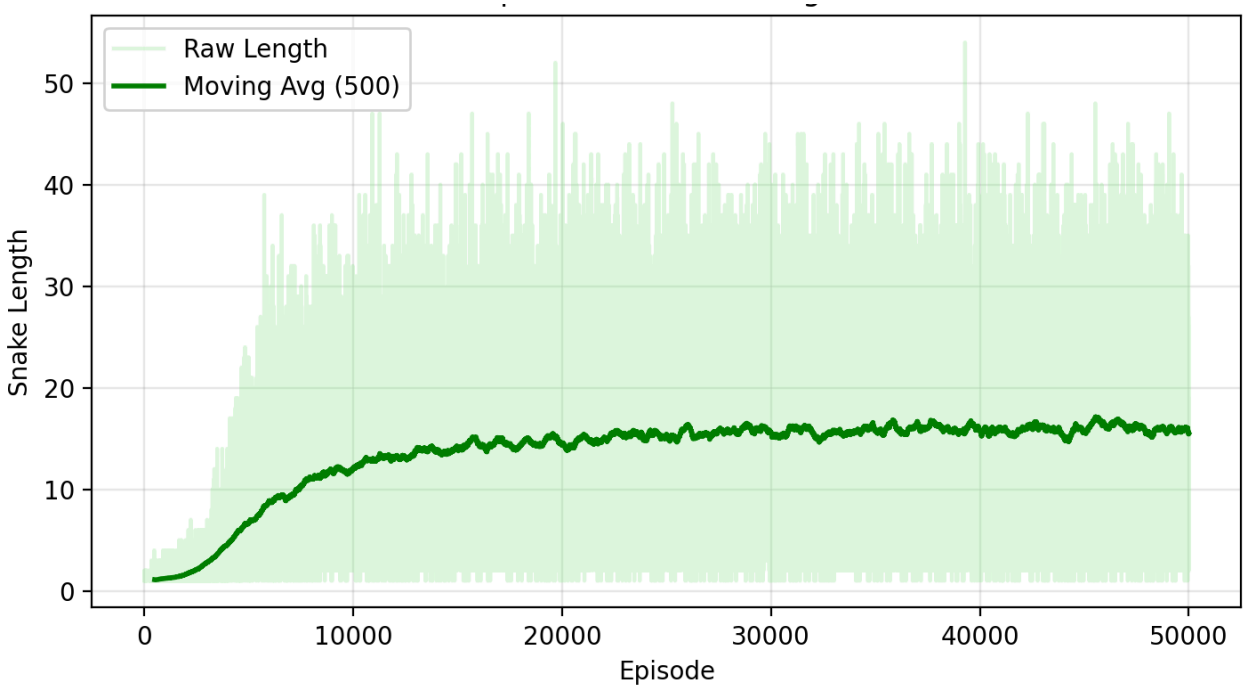
$$\delta = r + \gamma V_w(s') - V_w(s),$$

which measures how much better or worse the outcome was compared to the critic's expectation. The critic updates its value function parameters using this TD error so that future estimates become more accurate. The actor uses the same TD error to adjust the policy in the direction that increases the likelihood of actions that lead to positive outcomes and decreases the likelihood of actions that lead to negative outcomes. This is achieved by applying the policy gradient to log πθ (a|s) weighted by the TD error.

Results:

Reward Scheme 1:





Training Statistics:

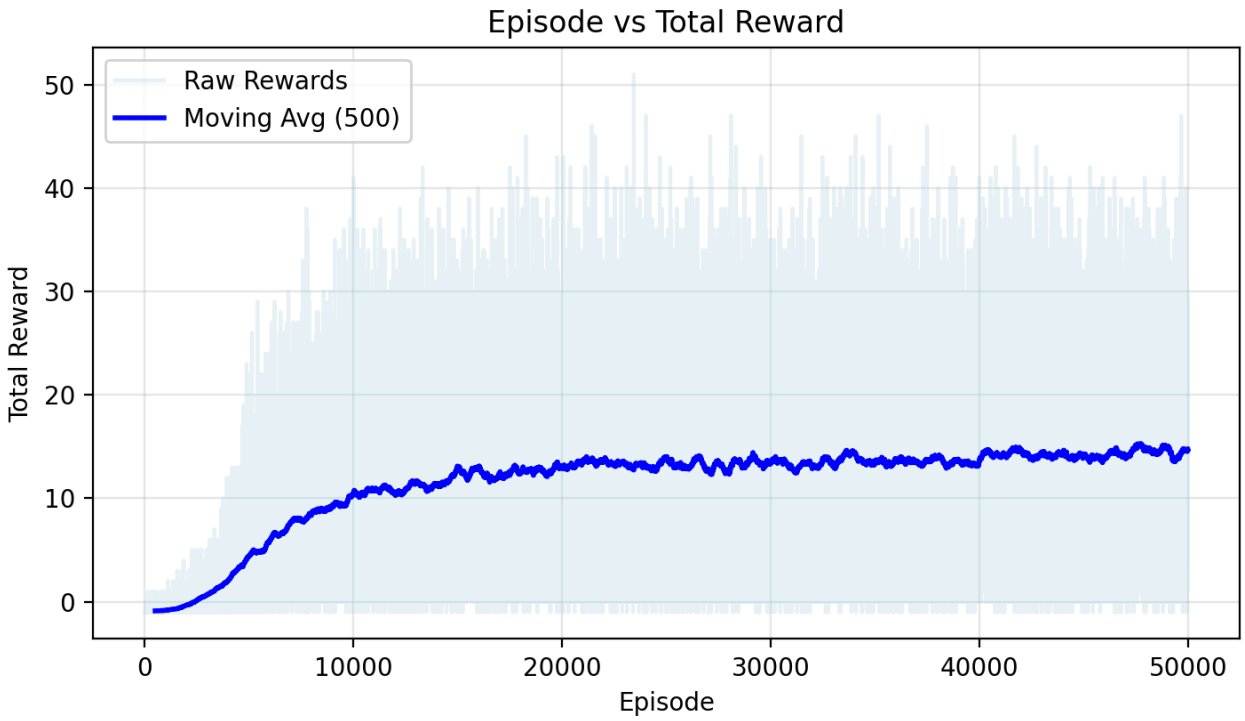
Total Episodes: 50000

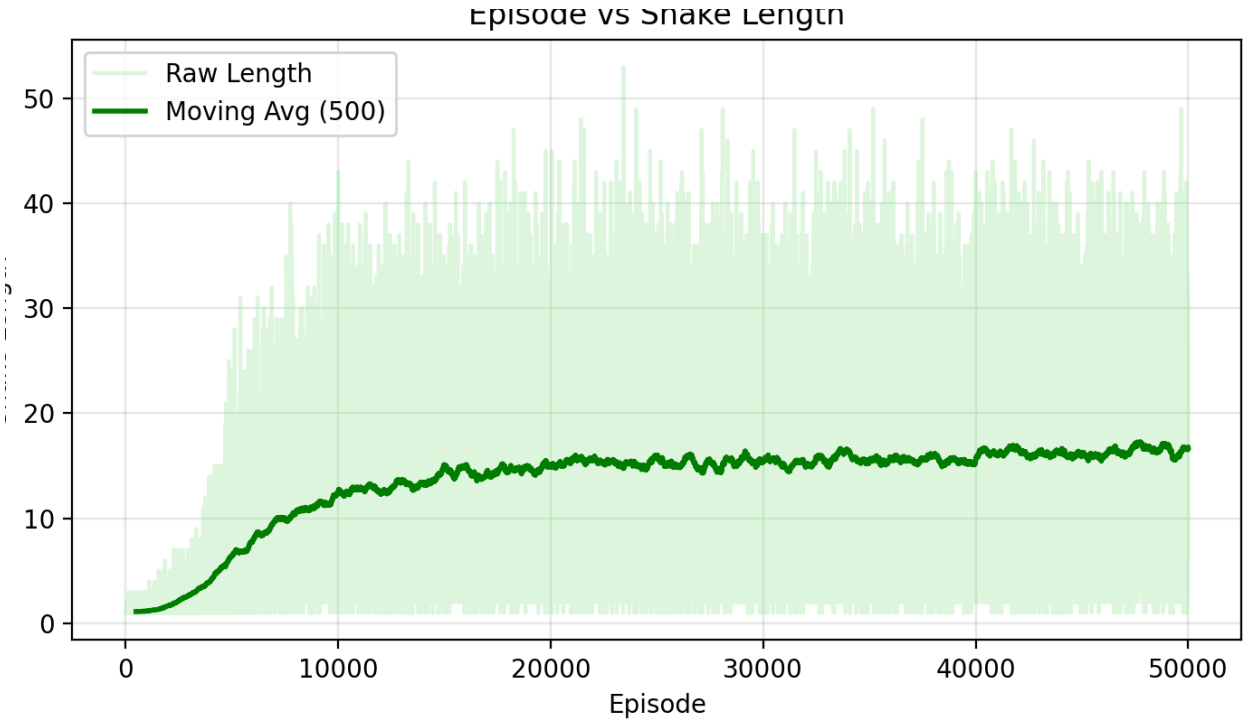
Average Reward: 11.57
Max Reward: 52.00
Min Reward: -1.00

Average Length: 13.57
Max Length: 54

Average Steps: 223.59
Max Steps: 966

Reward Scheme 2:





Training Statistics:

Total Episodes: 50000

Average Reward: 11.49
Max Reward: 51.00
Min Reward: -1.00

Average Length: 13.49
Max Length: 53

Average Steps: 222.54
Max Steps: 990

Overall Results:

We conducted an evaluation of four reinforcement learning methods: Q-Learning, SARSA, Deep Q-Networks (DQN), and Actor–Critic; across two different reward schemes.

Method	Reward Scheme	Max Length
Q-Learning	Simple (+1/-1)	34
Q-Learning	Scaled (length-based)	30
SARSA	Simple (+1/-1)	39
SARSA	Scaled (length-based)	32
DQN	Simple (+1/-1)	38
Actor–Critic	Simple (+1/-1)	54
Actor–Critic	Scaled (length-based)	53

We noticed that the simple reward scheme worked better for all methods, with Actor-Critic having similar results.

Based on our results, Q-Learning seemed to perform the worst with SARSA and DQN being slightly better than it and Actor-Critic performing the best.

However, the performance depends on the placement of the food considering the agents crashed into themself when the snake is longer and the food is on the other side.

We have linked some gameplays of the agent here.
Link to Agent Gameplay: CS 329 - Agent Gameplay

Work Distribution:

Aeshaa Nehal Shah - DQN
Harinarayan J - Actor Critic
Vyomika Vasireddy - Q-Learning, SARSA