REPORT - FINAL COURSE PROJECT

CSE 4/546: Reinforcement Learning, Fall 2024

SNAKE GAME AI | Team 10

Member 1 : Surya Venkata Rohit Moganti

Member 2 : Bhavani Kiran Kukunoor

ENVIRONMENT

Description of the Game:

Objective:
The goal is for the agent (snake) to consume food by navigating a grid and growing longer while avoiding collisions with itself and walls.

Grid Layout:
The game is played on a rectangular grid, with food and the snake.

Game rules -
The snake can travel in four different directions: up, down, left, and right.
When it consumes food, the agent is rewarded, which causes snake to grow
The game ends if it collides with barriers or itself.

State Representation: The current position of the snake (head and body segments)

- Position of the food.
- Direction of movement
- The distance to the nearest obstacles such (boundaries, body segments)

Action Space:

- The agent has four actions:
    - Move Up
    - Move Down
    - Move Left
    - Move Right

Reward System:

- **Positive Rewards**:
  - +10 points for eating food.
- **Negative Rewards:**
  - -10 points for colliding into wall or itself

Environment Dynamics: The agent's position on the dynamic grid is always shifting in response to its actions. Predicting and adjusting to the outcomes of every step is part of the challenge, which gradually strengthens the learning process.

BASELINE ALGORITHMS

→ Deep Q-Network (DQN)

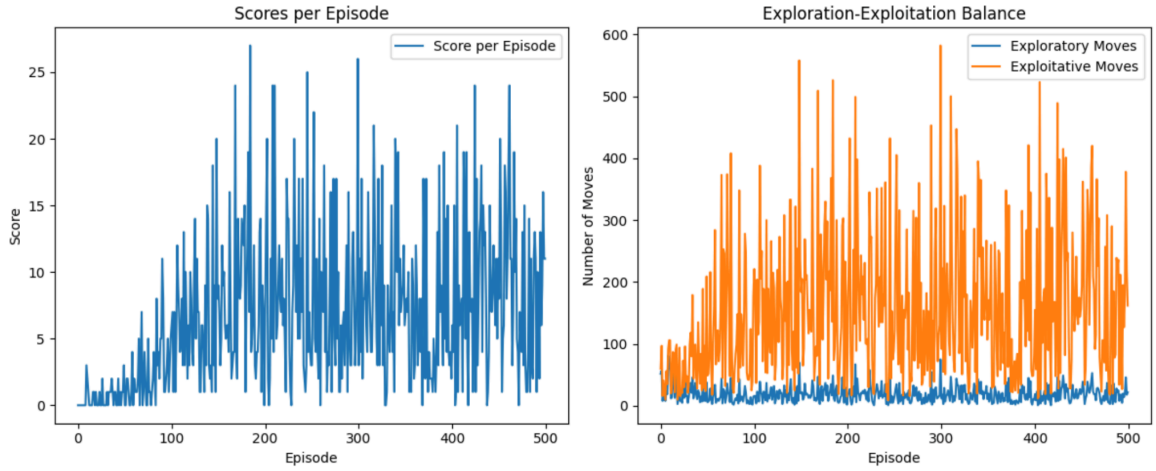
DQN is a reinforcement learning (RL) algorithm that combines Q-learning with deep neural networks. In DQN, the agent makes decisions based on observed states to maximize cumulative rewards by approximating the Q-value function using a neural network.

Why DQN for Snake Game:

The reason DQN works well for snake games is because they have discrete action spaces, thereby the agent only has to choose between moves like "left," "right," "up," and "down." It's a simple and well-liked choice for games, particularly those that allow for the representation of states as visual input (such as Snake's grid positions). Through a balance between exploration and exploitation, DQN improves in the agent's learning, progressively enhancing performance as it observes more gameplay.

```
ANALYSIS OF DQN ALGORITHM

Training completed. Average Reward: -10.00
Highest Score: 27
Lowest Score: 0
Average Score: 6.624
Average Lifespan: 173.76 steps per episode
```



1. Episode scores

Highest Score: The DQN agent's highest score is 27, which is marginally higher than the Double DQN results. This means that the agent reached a maximum score of 27 in at least one episode.

Lowest Score: 0: Because of early encounters with walls or with itself, the agent still had several events without a score.

Trend: The agent is learning to get better with time, as evidenced by the overall rising trend in scores, which is especially apparent in the early episodes (up to about 300). The scores, however, seem to vary more after episode 300, suggesting performance variability.

Fluctuations: Despite some episodes having greater scores, the DQN agent's results are still unpredictable, with some episodes displaying high scores and others falling. This is probably because DQN is an exploratory game and the dynamics of the Snake game are challenging.

2. Exploration and Exploitation

Exploratory Actions: In line with DQN's epsilon-greedy policy, which states that epsilon decays over time, the quantity of exploratory moves random actions performed to investigate the environment decreases as training goes on, particularly after the initial episodes.

Exploitative Actions: The quantity of exploitative moves rises as the agent gains knowledge. By the middle of training, the agent is mostly using the learned policy to take advantage of others.

Trend: The agent is progressively gaining confidence in its learnt Q-values, as evidenced by the progression from exploration to exploitation over time. This should hopefully assist the agent attain higher scores. The score swings, however, suggest that DQN may still have trouble staying consistent in a game like Snake, where some unexpected actions can lead to collisions that terminate the game.

Overall Performance:

Average Lifespan: The DQN agent survives longer on average in each episode, as seen by the average lifespan of 173.76 steps each episode, which is greater than the prior Double DQN result of 150.71 steps.

Learning Observations: There is a visible learning pattern in the DQN agent, which eventually favors exploitative actions. Although its peak score is somewhat higher than Double DQN's, the performance variability indicates that it might occasionally have trouble with generalization and stability in the gaming environment.

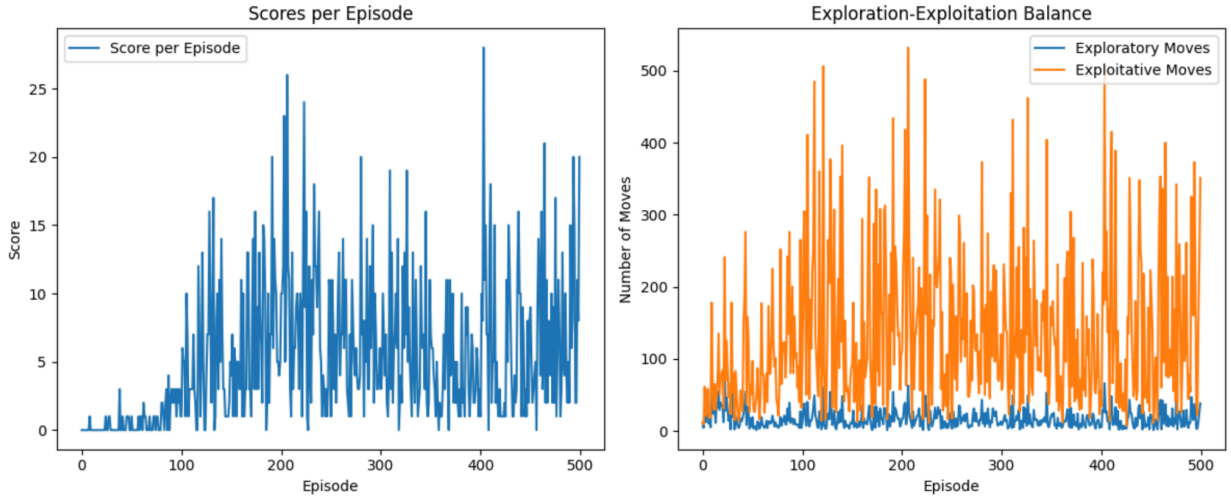
→Double Deep Q-Network (Double DQN):

A DQN variation called double DQN tackles the problem of overestimation in Q-value estimates. The agent may overestimate action values in standard DQN, which could result in less-than-ideal policies. Double DQN reduces overestimation bias by employing two networks: an action selection network to choose the optimal course of action and a target network to assess the action's Q-value.

Why Double DQN for Snake Game:
In a game like Snake game AI, where overestimating the advantages of particular actions (such continuously moving into a particular location) can lead to bad strategies, like trapping itself, Double DQN's capacity to generate more reliable and accurate Q-value estimations is advantageous. The enhanced Q-value stability of Double DQN can aid the agent in learning more dependable survival and food-gathering techniques.

```
ANALYSIS OF DOUBLE DQN ALGORITHM

Training completed. Average Reward: -10.00
Highest Score: 28
Lowest Score: 0
Average Score: 5.23
Average Lifespan: 150.71 steps per episode
```



1. Episode scores:
The snake's score for every episode is displayed in this plot.
Highest Score: 28: This means that the snake managed to reach a score of 28 in at least one episode, which means it successfully navigated and ate 28 food pieces before the program ended.
Lowest Score: 0: The snake may have lost points in some episodes because it collided with walls or its own body too soon.
Trend: Scores show a noticeable increase over time, particularly until about the 200th episode, indicating that the Double DQN model is progressively picking up a better strategy to live longer and gather more food.
Fluctuations: Scores continue to fluctuate a little even after the learning phase. This is to be expected as the snake game can be difficult with its many traps (walls, self-collisions), and the agent can at times engage in exploratory behavior that lowers points.

2. Exploration and Exploitation:

This figure illustrates how many exploitative and exploratory actions the agent takes in each episode.

Exploratory Actions: These are actions made at random to investigate the surroundings. The agent is choosing learned actions (exploitation) above random activities, as evidenced by the low frequency of exploratory moves (in contrast to exploitative moves).

Exploitative Actions: In these moves, the agent makes judgments based on the Q-values it has learned. Instead of exploring at random, the agent may be gradually growing more confident in its learned policy and depending on it to score higher, as evidenced by the growing amount of exploitative moves over episodes.

Trend: The plot exhibits more exploratory moves at first, but exploitative moves take the lead as training goes on. This pattern is common as Double DQN employs an epsilon-greedy policy, in which the agent increases its exploitation of its learned policy as epsilon (the likelihood of performing a random action) decreases over time.

Overall Performance:

Average lifetime: The agent often lives for a respectable amount of time, as evidenced by the average lifetime of 150.71 steps each episode. Since the objective of Snake is to optimize survivability in addition to achieving a high score, this metric is helpful to monitor.

Learning Observations: The score trends and the rise in exploitative moves show that the agent's performance gets better with time. Variability in the scores, however, suggests that the agent occasionally still fails to avoid traps, which is a common problem in Snake because of the dynamic nature of the game.

→Advantage Actor-Critic (A2C):

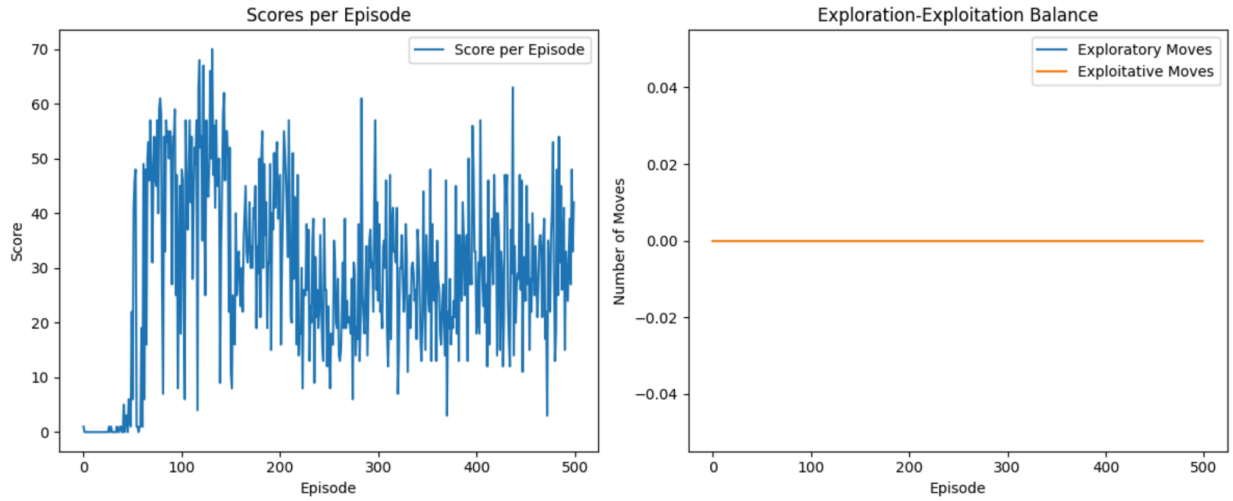
A2C is a policy gradient approach that uses two neural networks: the critic evaluates the actions by assessing their advantage over other options, and the actor chooses the actions. Since it is an on-policy algorithm, it learns from decisions made in accordance with its present policy and computes the advantage function to provide more stable learning guidance.

Why A2C for Snake Game:

By separating value estimation (critic) from action selection (actor), A2C may be able to explore more effectively since it learns not only what action to take but also why it is advantageous. This is particularly helpful in Snake game AI, where accuracy and timing are essential. Because A2C contains both an immediate action evaluation and a policy that evolves over time based on these assessments, it can assist the snake in avoiding bad decisions like abrupt twists into its own body or walls.

```
ANALYSIS OF ACTOR CRITIC ALGORITHM

Training completed. Average Reward: -10.00
Highest Score: 70
Lowest Score: 0
Average Score: 28.798
Average Lifespan: 970.43 steps per episode
```



1. Episode scores :

Highest Score: The A2C agent was able to attain a significantly higher score in certain episodes, as a result of improved policy stability, as evidenced by the highest score of 70, which is a notable improvement over the DQN and Double DQN outcomes.

Lowest Score: 0: Like other models, early game-ending movements are likely the reason why some episodes concluded with a score of 0.

Trend: In the early episodes, the scores rise quickly and attain comparatively high numbers. Although they still vary, the scores are continuously greater than those in earlier models starting with episode 100. This implies that although the A2C agent still exhibits some variability, it has learned useful strategies.

Fluctuations: Compared to DQN and Double DQN, there is less score fluctuation, suggesting that A2C may offer a more steady learning experience for the Snake game.

2. Exploration-Exploitation :

Exploitation vs Exploration: A flat line at 0 on the graph indicates that this tracking of exploratory and exploitative motions might not have been accurate. Through the actor-critic structure, where the actor makes decisions based on the policy and the critic assesses them, the policy is often constantly learnt in A2C. In DQN and Double DQN, where epsilon-greedy exploration is employed, the line separating exploration and exploitation steps may be less clear.

Trends: Although the exploration-exploitation balance isn't evident in this case, A2C usually depends on the actor-critic configuration, in which the agent is always updating its policy and tends to have a more seamless transition between exploration and exploitation without rigid separation.

Overall Performance:

Average lifespan: Compared to the other models, the average longevity is significantly higher at 970.43 steps per episode. This indicates that the A2C agent managed to live longer, indicating that it improved its ability to dodge plays that would finish the game.

Learning Observations: Of the three algorithms, A2C may be the most appropriate for this Snake game setting because it exhibits the highest average longevity, the highest peak score, and more consistent scoring across episodes. This performance, which improves stability and avoids overestimation, is probably the result of ongoing learning of the value function (via the critic) and policy (through the actor).

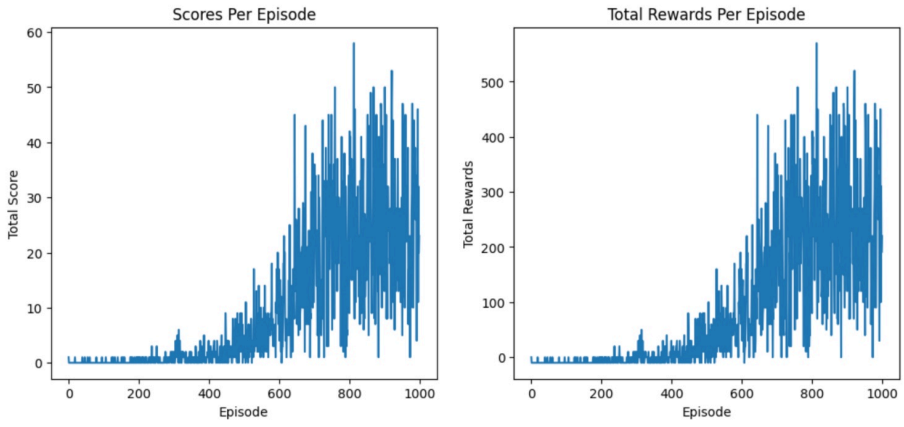
IMPROVED ALGORITHMS

→Dueling DQN:

Dueling DQN is an improved version of the standard Deep Q-Network (DQN), where the Q-value function is split into two parts: one representing the value of the state (how beneficial the state is) and the other representing the advantage of specific actions (how much better an action is compared to the others in that state). These two components are then combined to calculate the final Q-value. This approach enables the agent to more efficiently focus on the most important aspects of the environment.

Why Dueling DQN for Snake Game:

For the Snake game, Dueling DQN helps the model differentiate between the overall value of a state and the advantage of particular actions. This is especially useful in Snake, where not every action leads to major changes in the state, particularly when the snake is far from obstacles or food. By prioritizing important features of the state (like proximity to food or walls), Dueling DQN allows the snake to make smarter decisions, avoid collisions, and navigate more effectively in a complex environment.



Average Number of steps taken by the model in each episode is: 264.073

1. Episode Scores:

Overall, the scores steadily increase after the initial phase of learning. From around episode 200, there is a noticeable improvement in the agent's performance, with more consistent scores starting around episode 500. While there are still some fluctuations, they are less dramatic compared to earlier episodes, showing that the agent is stabilizing its learning.

Though the scores continue to vary, the fluctuations are less severe as training progresses, suggesting that the agent is becoming more consistent in its gameplay as it learns.

2. Exploration-Exploitation:

Exploitation vs Exploration: Although the graph doesn't directly display exploration versus exploitation, we can infer that the agent transitions from exploration to exploitation as training advances. Early episodes show more variation in scores, indicating that the agent is exploring different strategies, while later episodes demonstrate more consistent performance, suggesting a shift to exploiting learned strategies.

Trends: While the balance between exploration and exploitation isn't explicitly shown, the agent likely explores more in the beginning (indicated by the wider score range) and gradually focuses on exploiting successful strategies as it improves, as seen in the gradual rise in scores and rewards.

3. Overall Performance:

Average Lifespan: The agent lasts an average of 264 steps per episode, which suggests that it is learning to survive longer within the game. This indicates that the agent is becoming better at avoiding mistakes that lead to early game endings.

Learning Observations: The model shows noticeable progress in both scores and rewards as training continues. This improvement is likely the result of factors such as better exploration-exploitation balancing, more accurate policy evaluations, and continuous learning from each episode. The smoother progression after episode 500 highlights the agent's growing understanding of the game and its strategies.

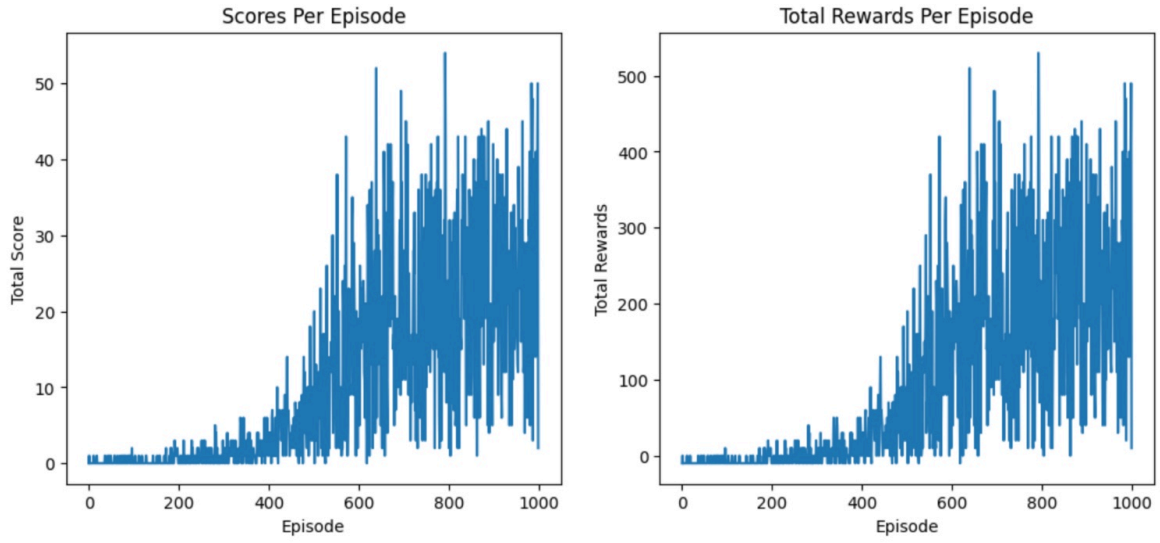
→PER: Prioritized Experience Replay (PER):

Prioritized Experience Replay (PER) is an improvement over traditional experience replay in

reinforcement learning. In the basic method, experiences (state, action, reward, next state) are stored in a replay buffer and sampled randomly for training. PER, however, gives higher priority to experiences with higher temporal-difference errors, meaning more surprising or significant experiences are replayed more frequently. This helps the agent learn from the most valuable experiences, accelerating the learning process and enhancing performance.

Why PER for Snake Game:

In the Snake game, some states are more important for the agent's success than others. With PER, the agent can focus on learning from more crucial experiences, such as when the snake is near food or obstacles. This prioritization enables faster learning and helps the agent improve its decision-making abilities, allowing it to avoid collisions and target food more effectively. As a result, PER leads to better gameplay, helping the agent achieve higher scores and longer survival in the game.




```
Average Number of steps taken by the model in each episode is:  262.859
```

1. Episode Scores:

Scores gradually improve over time, with a noticeable increase from episode 200 onwards. The impact of Prioritized Experience Replay (PER) becomes evident, as the agent focuses on learning from more valuable experiences, resulting in better performance starting from episode 500.

Although there are still fluctuations in scores, they become less severe as training progresses. This suggests that PER helps the agent stabilize its learning by prioritizing critical experiences, which smoothens the learning process.

2. Exploration-Exploitation :

Exploitation vs Exploration: During the early episodes, the agent explores different actions. Over time, PER allows the agent to focus more on exploiting rewarding experiences, reflected in the more consistent performance in later episodes.

Trends: PER supports the agent's shift from exploration to exploitation by prioritizing significant experiences. This is seen in the steady increase in performance, as the agent refines its strategy by relying on valuable past experiences.

3. Overall Performance :

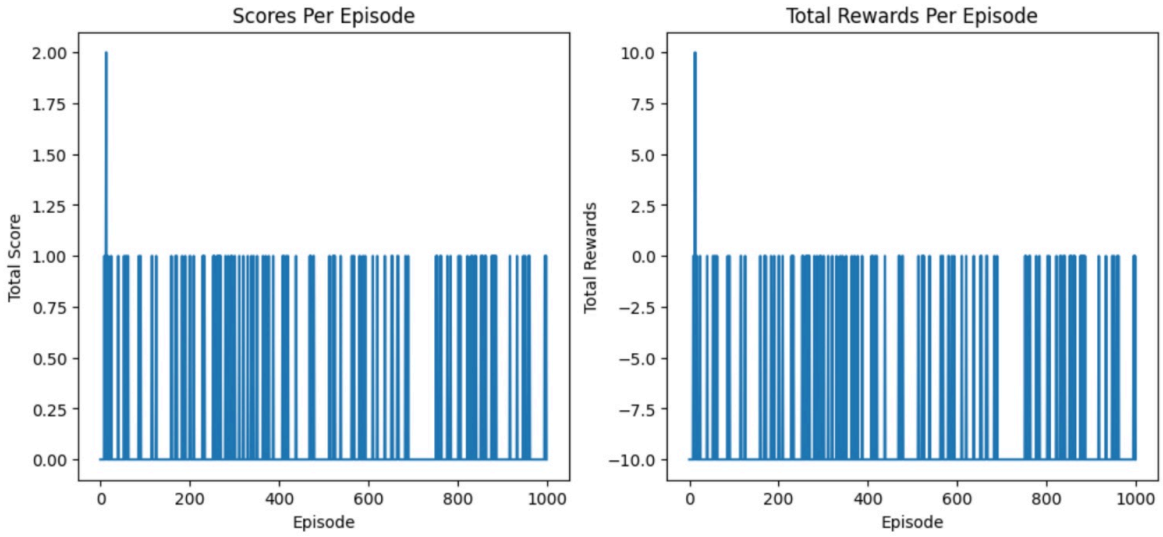
Average Lifespan: The agent survives an average of 264 steps per episode. This increase indicates that the agent is becoming better at avoiding early mistakes and learning from valuable experiences, allowing it to last longer in the game.

Learning Observations: The agent demonstrates significant improvement in scores and rewards over time. This growth can be attributed to Prioritized Experience Replay, which accelerates learning by allowing the agent to focus on the most important past experiences. The more stable performance after episode 500 highlights the agent's refined decision-making process, aided by PER.

→ **PPO:** Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a reinforcement learning method that focuses on improving the policy in a controlled and stable manner. It uses a clipped objective function to restrict how much the policy can change with each update, preventing large adjustments that could destabilize learning. PPO is valued for its simplicity and effectiveness in ensuring stable training while making gradual improvements to the policy.

Why PPO for Snake Game:
PPO is well-suited for the Snake game because of its ability to handle complex and dynamic environments where the agent needs to make decisions based on current and past states. Since Snake requires balancing between survival and collecting food, PPO's controlled learning approach helps the agent avoid taking overly risky actions, which could lead to failure.




```
Average Number of steps taken by the model in each episode is:  322.185
```

1. Episode Scores:

The agent's performance remains fairly consistent but low throughout the episodes, suggesting that it has not improved or adapted much over time. There is no clear progress in scores, indicating the model may not be learning effectively.

The scores exhibit frequent minor variations around a base value, implying that the agent has difficulty achieving stable performance.

2. Exploration-Exploitation:

Exploitation vs Exploration: The exploration-exploitation balance seems to remain static, as seen in the score graph. The lack of consistent improvement in scores suggests that the agent has not effectively shifted from exploration to exploitation.

Trends: The agent likely continues to explore random actions, as evidenced by the absence of steady progress in the scores. This indicates that the agent is not effectively utilizing previously learned strategies.

3. Overall Performance:

Average Lifespan: With an average of 322 steps per episode, the agent is able to survive for a decent amount of time in each episode. However, this does not translate into higher scores or better rewards, indicating limited learning.

Learning Observations: Although the agent manages to last longer in the game, its learning process appears inefficient. There is no noticeable improvement in the performance metrics, suggesting that the PPO model requires further adjustments or hyperparameter tuning to enhance its performance in this task.

REFERENCES

DQN
https://medium.com/@shruti.dhumne/deep-q-network-dqn-90e1a8799871
DoubleDQN
https://medium.com/@qempsil0914/deep-q-learning-part2-double-deep-q-network-double-dqn-b8fc9212bbb2
A2C
https://medium.com/intro-to-artificial-intelligence/the-actor-critic-reinforcement-learning-algorithm-c8095a655c14
Snake game AI
https://medium.com/@nancy.q.zhou/teaching-an-ai-to-play-the-snake-game-using-reinforcement-learning-6d2a6e8f3b1c
PER :
https://towardsdatascience.com/how-to-implement-prioritized-experience-replay-for-a-deep-q-network-a710beecd77b
Dueling DQN :
https://towardsdatascience.com/dueling-deep-q-networks-81ffab672751