Snake Game using Reinforcement Learning
Deep Learning Assignment #2

Bernardo Calvo
MIEI — 59187
NOVA School of Science and Technology
Caparica
b.calvo@campus.fct.unl.pt
Group 2

Bruno Carmo
MIEI — 57418
NOVA School of Science and Technology
Caparica
bm.carmo@campus.fct.unl.pt
Group 2

Sahil Kumar
MIEI — 57449
NOVA School of Science and Technology
Caparica
ss.kumar@campus.fct.unl.pt
Group 2

Abstract

In this project, we explored Deep Reinforcement Learning algorithms to train an agent that can play the game of Snake at a high level. In order to better understand the different variants of this algorithm and to further improve our results we implemented two deep reinforcement learning techniques, Experience Replay and Target Network, and experimented with two different exploration algorithms, Epsilon-Greedy and the Upper Confidence Bound. In our experiments, after testing with and without the different deep learning techniques we show that our agent is most efficient when using both Experience Replay and a Target Network while using an Epsilon-Greedy exploration algorithm.

1 Introduction

Reinforcement learning is a type of machine learning that allows an agent to learn from its environment by receiving feedback in the form of rewards or punishments. The agent's objective is to maximize the cumulative reward by optimizing its value function or Q-function, an action-value function that determines the value of being at a particular state and taking a specific action at that state. This value is often referred as the the Q-value (the "Quality" of that action). Deep Reinforcement Learning uses neural networks to approximate, given a state, the different Q-values for each possible action at that state.

The goal of this assignment is to use Deep Reinforcement Leaning, more specifically the Deep Q-Learning algorithm, to train an agent that can play the game of Snake at a high level. To achieve this we will be using various techniques such as Experience Replay, Target Network and trying various exploration strategies. We will also study how models using these techniques compare in terms of learning speed, performance, and stability.

The Snake game will be given as an image consisting of a board and a border with varying width and height, one or more fruits and optionally grass and grass growth. The agent (the snake) has 3 possible actions, move straight, to the left or to the right. When the snake head enters a cell, it receives a reward equal to the amount of grass present in that cell, when it eats a fruit it receives a reward of 1.

Preprint. Under review.

Eating a fruit makes the snake grow its tail by 1 and if it enters a cell with its tail or with the game border the game ends.

In the following sections, we will present the methodology used, as well as analyze the obtained results for the different models and finally the conclusion of the work.

2 Methodology

The state space and action space of the snake game are too big to perform Q-Learning, as such, we trained our snake agent using Deep Q-Learning, which consists in optimizing a Q-function that returns a Q-value, that is, a value of taking a specific action in a state, using a neural network.

Considering that a specific time step from the snake game can be represented as an image, we used a convolutional neural network as the backbone network of our snake agent to learn characteristics of the images. Stacking up with the convolutional neural network, we used fully connected layers, such that the deep q-network (the convolutional neural network plus the fully connected layers), returns a Q-value for each of the possible actions in the snake game.

Inspired by the theoretical classes, we trained our snake agent using the following algorithm:

Algorithm 1: Deep Q-Learning

Data: deep q-network $dqn_\theta(x)$, exploration strategy π, positive integer $num\_episodes$, discount factor γ
Result: Q-value function
$y \leftarrow 1$;
$X \leftarrow x$;
$N \leftarrow n$;
for $i \leftarrow 1$ ***to*** $num\_episodes$ **do**
  Observe state S_0
  $t \leftarrow 0$
  repeat
    Chose action A_t using π
    Take action A_t and observe reward R_{t+1} and state S_{t+1}
    Get Q-values $Q(S_t, a)$ for all possibles actions a using $dqn_\theta(x)$
    Get Q-values $Q(S_{t+1}, a)$ for all possibles actions a using $dqn_\theta(x)$
    if S_{t+1} *is a terminal state* **then**
      $Q(S_t, A_t) \leftarrow R_{t+1}$
    else
      $Q(S_t, A_t) \leftarrow R_{t+1} + \gamma \times max_a(Q(S_{t+1}, a))$
    end
    Fit $dqn_\theta(x)$ on S_t, $Q(S_t, a)$ for all possibles actions a, for 1 epoch using the Adam optimizer
  until S_t *is a terminal state*;
end

However, using the algorithm 1, the snake agent only trains for one board image per time step, which does not yield good results. Therefore, and according to the assignment, we implemented a snake agent that uses an Experience Replay buffer in the training. The Experience Replay buffer consists of a collection of examples of the game played following some heuristic. Therefore, we created a heuristic in order to fill the experience replay with examples of moves that aim to maximize the final score in the game, rather than just performing random actions. By calculating the Manhattan distance between the apple, which is the main objective, and the snake's head for each of the 3 possible moves, the action resulting in the shortest distance is chosen, and making sure that this action will not cause the snake to collide with its own tail. Since the game typically takes place on a squared grid, indirectly, this heuristic ensures that the snake does not collide with the walls.

Inspired by [2], when using the Experience Replay buffer, algorithm 1 is changed, such that, instead of getting Q-values just for the current state S_t and the corresponding next state S_{t+1}, the algorithm gets the Q-values for a sample of the examples inside the buffer and their corresponding next states.

Then, fits the $dqn_\theta(x)$, on that sample and on that Q-values. The sample size depends on the buffer size, which has more details in section 3.1.

Nevertheless, the weights of the $dqn_\theta(x)$ are constantly changing, which in turns affects stationarity of the target Q-values, that correspond to the values towards which we try to push the Q-values. As such, and according to the assignment, we implemented the Target Network technique that consists in using another network to provide the target Q-values and then periodically update this network using the weights of the original network.

Using a Target Network the algorithm 1 is changed, such that, instead of getting the Q-values $Q(S_{t+1}, a)$ for all possibles actions a using $dqn_\theta(x)$, we get them using the Target Network. Also, after N number of episodes, after ending the **repeat**...**until** loop, we update the Target Network with the weights of the original network. The N hyperparameter depends on the number of episodes, which has more details in section 3.1.

According to the assignment, we implemented the following exploration strategies:

- ϵ**-Greedy**: the action is selected according to the following formula:

$$A_t = \begin{cases} \underset{a}{argmax}\ Q_t(S_t, a), & \text{with probability } 1 - \epsilon \\ \text{a random action}, & \text{with probability } \epsilon \end{cases} \tag{1}$$

  where $Q_t(S_t, a)$ corresponds to the Q-value for state S_t when taking action a obtained using $dqn_\theta(x)$. We generate a random value between 0 and 1. If this value is below the current value of ϵ, this means the snake agent will explore the environment and one of the possible actions is randomly chosen. Otherwise, the snake agent will exploit its "knowledge" and the action is selected based on the highest Q-value for the current state.
- **Upper Confidence Bound**[1]: the action is selected according to the following formula:

$$A_t = \underset{a}{argmax} \left[Q_t(S_t, a) + c\sqrt{\frac{\log(t)}{N_t(a)}} \right] \tag{2}$$

  where $Q_t(S_t, a)$ corresponds to the Q-value for state S_t when taking action a obtained using $dqn_\theta(x)$, c corresponds to the confidence, t corresponds to the time step and $N_t(a)$ corresponds to number of times action a has been taken prior to time step t. With this strategy, rather than performing exploration by simply selecting an arbitrary action, the exploration-exploitation balance is changed as the snake agent gathers more knowledge of the environment. It moves from being primarily focused on exploration, when actions that have been tried the least are preferred, to instead concentrate on exploitation, selecting the action with the highest estimated reward.

3 Results

We obtained our results by performing various experiences according to section 3.1. To compare the different results we obtained from different techniques used to train the snake agent, we do an ablation study as described in section 3.2.

3.1 Experimental Design

As part of the assignment we implemented two Deep Reinforcement Learning techniques: Experience Replay and Target Network.

To compare, how the implemented snake agent performs using both techniques, we take into account the number of steps the snake agent took by episode, its performance (how much score it got per episode), and its stability (if the scores stabilizes or not). Considering these metrics we have done an ablation study, that consists of running the snake agent with one of the techniques above or with both of them and then compare the results obtained by these runs with the results obtained when running the snake agent without any of the techniques, without changing the hyperparameters and maintaining the same architecture of the convolutional neural network.

However, the assignment has hyperparameters that are external to the snake agent, such as, the board width, the board height, the border size, the maximum amount of grass in a location and the growth

rate of the grass per step. In order to minimize running time, we used a small board (14x14 with border size equal to 1) and the recommended values for the grass (0.05 for the maximum amount of grass and 0.001 for the grass growth rate). These were the hyperparameters values used in the ablation study.

For the purpose of minimizing the running time, we also used a simple convolutional neural network represented in figure 1, that is used to predict the Q-values for the 3 different actions.

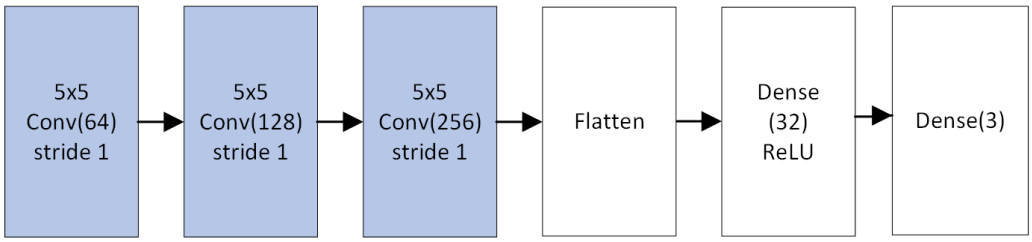


Figure 1: Snake agent architecture network.

Also, we chose to use the Adam optimizer with a 0.001 learning rate after comparing it with the Stochastic Gradient Descent optimizer with the same learning rate, that had worse results. For the loss, we chose the Mean Squared Error. We used 500 episodes to train the snake agent, because of limited computer resources, and with a discount factor or gamma of 0.9 (which is recommended to be between 0.9 and 1.0), which tells how much importance should be given to future rewards.

To minimize running time but also to maximize learning, whenever we used Experience Replay we chose the buffer size to be equal to 50k, after comparing results with buffers of different sizes (3k, 5k, 10k). We also used this buffer size with a 1024 sample size in the ablation study.

Whenever we used the Target Network, according to the number of episodes above, the weights were updated every 10 episodes.

3.2 Epsilon-Greedy Agent vs Upper Confidence Bound Agent

Before we did the ablation study, we needed to decide which exploration strategy to use in the study. As such, considering that, hypothetically, a snake agent with both Experience Replay and Target Network is better than trained without both of them or only one of them, we generated results for a snake agent with Experience Replay and Target Network trained using the Epsilon-Greedy exploration strategy and the Upper Confidence Bound exploration strategy.

Considering the Epsilon-Greedy exploration, we experimented to decay the epsilon, linearly and exponentially. We obtained better results with a linear decay, as such, we chose to use the linear decay where the epsilon starts at 1.0, decays 0.002 per step until it reaches 0.1, and maintains this value for the rest of the experiments. For the Upper Confidence Bound exploration strategy, we used a confidence of 95%.

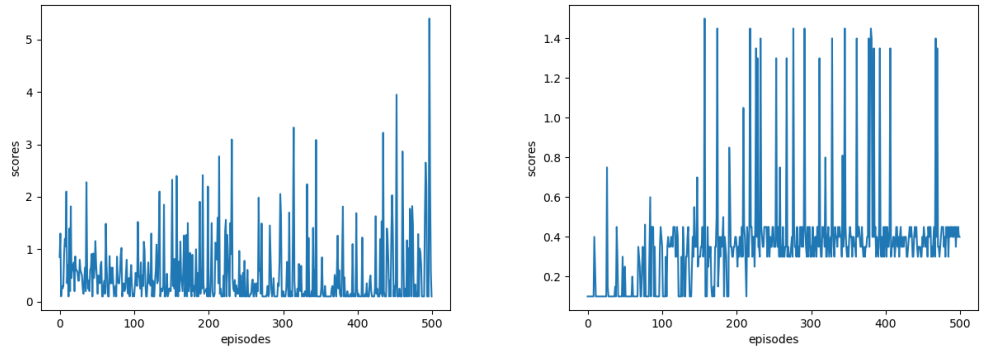
The plots in figure 2 show the score results we obtained for both exploration strategies.

By comparing the results, we can see that for the same hyperparemeters, the Epsilon-Greedy exploration strategy performed better than the Upper Confidence Bound strategy. Therefore, we used the Epsilon-Greedy exploration with the same hyperparemeters, for the rest of the experiments.

3.3 Ablation Study

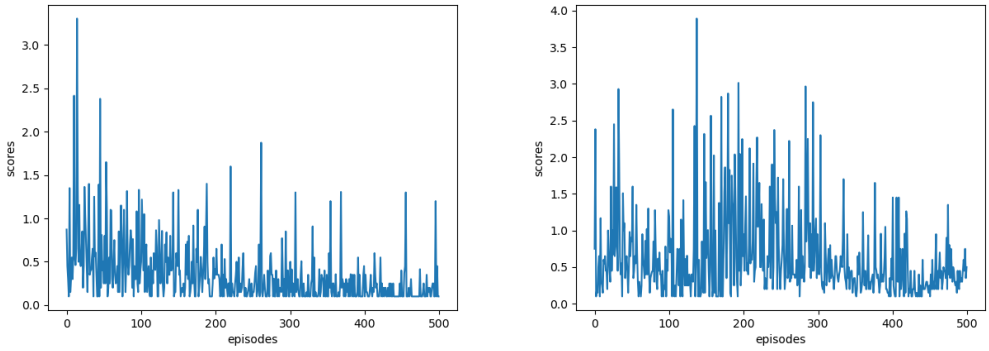
3.3.1 DQN vs. DQN-ER

Taking advantage of experience replay, the model does not need to learn the game from scratch and can learn based on other gameplay experiences, which in this case, are mostly generated from the heuristic mentioned before. Therefore, it is expected that by using Experience Replay, the model will learn better compared to not using it because it will take more efficient use of other experiences.



(a) Score plot for Epsilon-Greedy strategy (b) Score plot for Upper Confidence Bound strategy

Figure 2: Comparison of **exploration strategies**.



(a) Score plot of the snake agent **not** using Experience Replay (b) Score plot of the snake agent using Experience Replay

Figure 3: Comparison of **scores** between not using Experience Replay (left) and using Experience Replay (right).

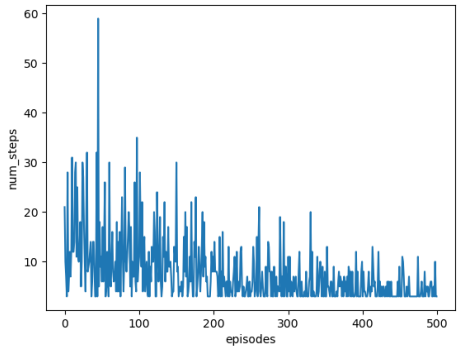
Accordingly to the plots, regarding the scores, represented in figure 3, we can observe that with Experience Replay, the model ends up achieving higher scores when compared to its non-usage. This leads to an increase in the number of steps per episode, as represented in the plots of figure 4, which in turns leads to an increase in running time.

As expected, we concluded that using Experience Replay, the model learns the basics of the game better.

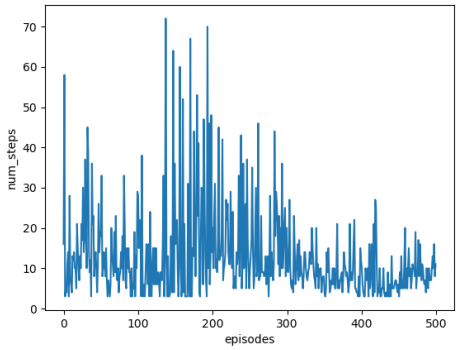
3.3.2 DQN vs. DQN-TN

By using a Target Network, the model doesn't have to predict the Q-values based on the same network that is constantly learning and changing its weights. This avoids the situation of "chasing its own tail" and consequently leads to greater stability, because, the weights from the q-network are copied to this target network, periodically. Therefore, it is expected that by using Target Network, the model will learn better compared to not using it.

Accordingly to the plots in figure 5, we can observe the models has similar scores with and without Target Network. However, we can observe that with the Target Network, the snake agent eats at least one apple more times than without the Target Network. This leads us to believe, that the agent is more stable when using this technique. This stability can also be seen after approximately, the 420 episodes, where the agent takes scores above a certain value and never too close to zero as before.

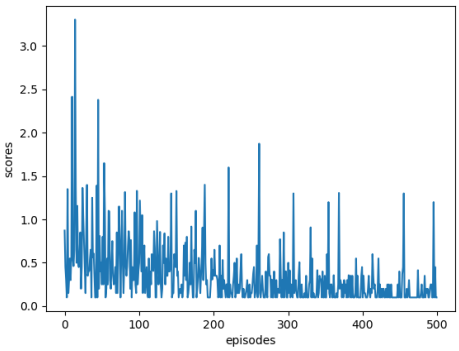


(a) Number of steps plot of the snake agent **not** using Experience Replay

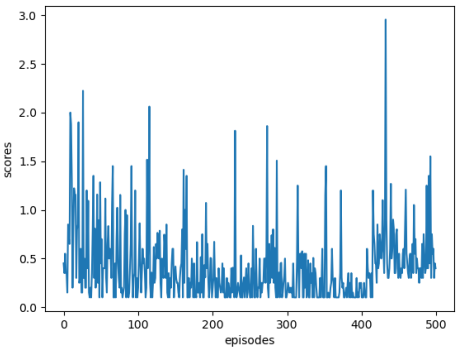


(b) Number of steps plot of the snake agent using Experience Replay

Figure 4: Comparison of **number of steps** between not using Experience Replay (left) and using Experience Replay (right).



(a) Score plot of the snake agent **not** using Target Network



(b) Score plot of the snake agent using Target Network

Figure 5: Comparison of **scores** between not using Target Network (left) and using Target Network (right).

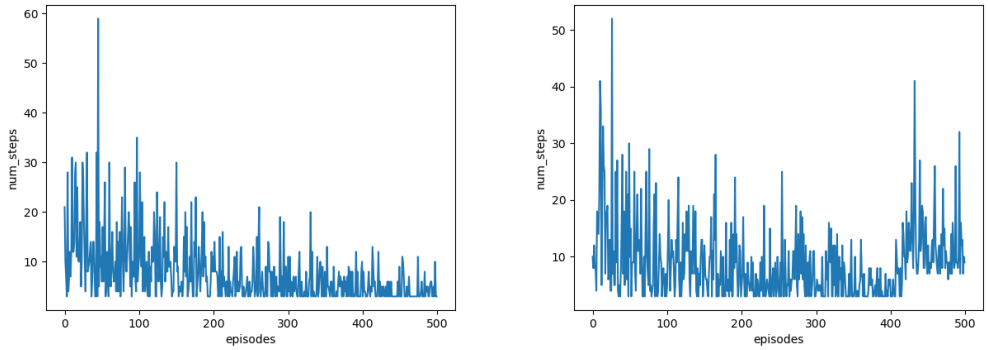
Regarding the number of steps taken per episode, represented in figure 6, as expected there is not a performance overhead when using a Target Network.

As expected, we concluded that using a Target Network the model is more stable in learning the basics of the game and does not have a performance overhead.

3.3.3 DQN vs. DQN-ER-TN

By using experience replay along with a target network, it is expected to have the advantages that both techniques offer, as previously explained. Specifically, this includes faster learning and greater stability of the model.

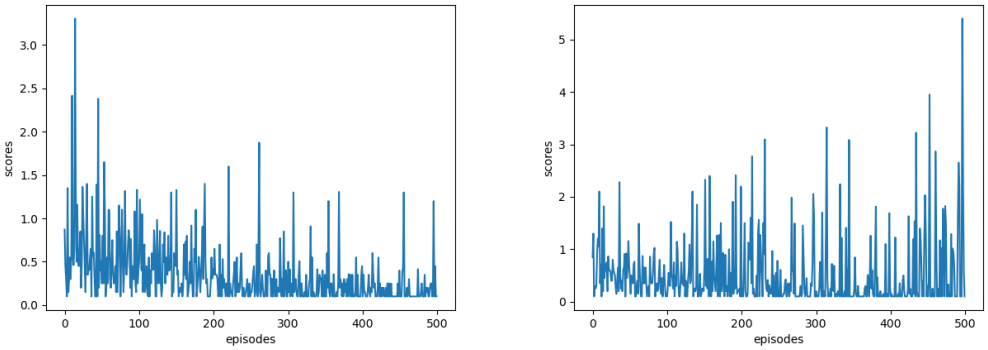
By analysing these plots, figure 7, we can observe the differences in performance between using Experience Replay and a Target Network and not using any techniques. Overall, we can conclude that while using both techniques the scores are much higher and the snake achieves better scores more frequently.



(a) Number of steps plot of the snake agent **not** using Target Network

(b) Number of steps plot of the snake agent using Target Network

Figure 6: Comparison of **number of steps** between not using Target Network (left) and using Target Network (right).



(a) Score plot of the snake agent **not** using Experience Replay and Target Network

(b) Score plot of the snake agent using Experience Replay and Target Network

Figure 7: Comparison of **scores** between not using Target Network and Experience Replay (left) and using both (right).

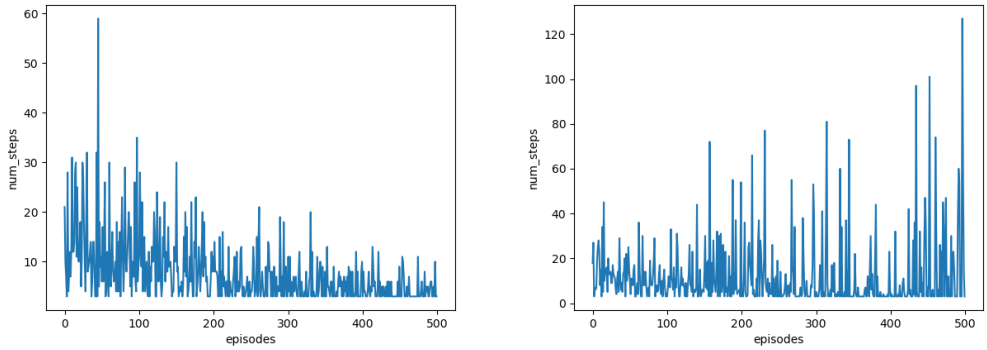
Regarding the number of steps taken per episode, represented in figure 8, as expected while using both Experience Replay and a Target Network, the snake makes a much higher number of steps in order to achieve better scores.

So, as a result of using both techniques, we can conclude that this leads to the model being able to learn better compared to not using it.

4 Conclusion

To conclude, with this work, it was possible to study in more depth the different components that make up the Deep Q-Learning algorithm applied to the game of Snake. To explore these components, we studied and compared results between using different exploration strategies, specifically the Epsilon-Greedy strategy and the Upper Confidence Bound strategy, as well as the use or non-use of Deep Reinforcement Learning techniques such as Experience Replay and Target Network.

After optimizing our results by testing various parameters such as using different optimizers, learning rates and Neural Networks as well as different number of episodes, epsilon values and replay buffer sizes we concluded and demonstrated that the Epsilon-Greedy strategy showed better results for this



(a) Number of steps plot of the snake agent **not** using Experience Replay and Target Network

(b) Number of steps plot of the snake agent using Experience Replay and Target Network

Figure 8: Comparison of **number of steps** between not using Target Network and Experience Replay (left) and using both (right).

problem. We also showed that the use of Experience Replay, makes the model learn the basics of the game more quickly, and with a Target Network, it becomes more stable. Therefore, in this case, the ideal model would involve using all these techniques in order to maximize the score, which was the one that achieved the highest score values on the presented results.

References

[1] Steve Roberts. *The Upper Confidence Bound (UCB) bandit algorithm*. Jan. 2021. URL: `https://towardsdatascience.com/the-upper-confidence-bound-ucb-bandit-algorithm-c05c2bf4c13f`.

[2] *The Deep Q-Learning Algorithm*. URL: `https://huggingface.co/learn/deep-rl-course/unit3/deep-q-algorithm?fw=pt`.