# Deep Q-Learning based approach applied to the Snake game

Deep Q-Learning based approach applied to the Snake game

Victor MAYAUD[1], Elliot BOUCHY[1]

[1] Data Science, EURECOM, France

**Abstract**

In recent years, the application of Deep Q-Learning (DQL) in classical games has gained significant attention in the field of artificial intelligence. This paper presents a comprehensive approach to training an AI agent to play the Snake game using Deep Q-Learning. By leveraging a neural network to approximate Q-value functions, the agent learns to make optimal decisions through interactions with the game environment. The state of the game is represented by a detailed vector incorporating the positions of the snake and food, and the actions are chosen based on a balance of exploration and exploitation, controlled by an epsilon parameter. We explore various strategies for hyperparameter tuning and implement a stronger AI adjustment mechanism that considers future steps to optimize decision-making policies. The experimental results demonstrate significant improvements in the agent's performance over training epochs, highlighting the effectiveness of our approach. This study underscores the potential of advanced reinforcement learning techniques in developing intelligent agents for complex tasks.

## 1 Introduction

In the evolving landscape of artificial intelligence, the application of Deep Q-Learning (DQL) to classical games offers a fertile ground for exploring and demonstrating the capabilities of reinforcement learning. This study focuses on training an AI agent to play the Snake game using DQL, aiming to develop an intelligent agent that can learn optimal strategies through continuous interaction with the game environment. By employing a neural network to approximate Q-value functions, the agent adapts and improves its performance based on rewards and penalties encountered during gameplay. In this project, we further enhanced the complexity of the task by incorporating random blocks within the game environment, which challenged the AI to navigate more dynamically changing conditions.

## 2 Snake Game

The Snake game is a timeless arcade classic where players guide a growing snake to consume randomly appearing food items on the screen, navigating through a confined space to avoid colliding with walls or the snake's own lengthening body. As the snake eats, it grows longer and gameplay becomes increasingly challenging, requiring greater strategic planning, quick reflexes, and spatial awareness. The game's escalating difficulty with the snake's increasing speed and size, combined with its simple yet engaging mechanics, has cemented its status as a universally cherished game that tests players' coordination and decision-making skills.

## 3 Deep Q-learning model

### *3.1 First Approch*

we opted for a simple initial approach to familiarize ourselves with the model's architecture and learning mechanisms. Here is a detailed description of this approach, including the structure of the neural network model and the training and reward methods used.

**Architecture:**

The input layer is made up of 8 neurons. These neurons represent the features of the environment that the Snake can perceive. In our case, this includes information about the three blocks ahead, the current direction, and possibly other environmental indicators.

Hidden layers: There are two hidden layers in the network:

The first hidden layer has 16 neurons, followed by a ReLU (Rectified Linear Unit) activation function to introduce non-linearity.

The second hidden layer has 8 neurons, also followed by a ReLU activation function.

Output: The output layer has 4 neurons, representing the four possible actions the Snake can perform (up, down, left, right).

**Reward System**

The reward system is essential to guide AI learning by providing feedback on its actions. In our approach, we have designed a reward system based on different game scenarios, to encourage desirable behaviors and discourage those that lead to failure.

Loss of Game (Collision): When the Snake collides with a wall or rolls over, the game is over. In this case, a severe penalty of -2.0 is applied. This severe penalty is designed to dissuade the AI from taking actions that could lead to a collision, motivating it to find strategies to avoid these situations and prolong the game.

Victory (Completely Filled Screen): If the Snake manages to fill the entire screen with its body, this represents a victory. Although this situation is rare, a positive reward of 1.0 is awarded. This reward, though modest, signals to the AI that it has reached the game's ultimate goal. It encourages the AI to pursue strategies that maximize its growth until it fills the screen.

Eat an Apple: Each time the Snake eats an apple, a positive reward of 1.0 is awarded. This reward is designed to encourage the AI to actively seek out apples, which is crucial to its growth and to prolonging the game. By eating apples, the Snake grows, increasing the difficulty of the game, but also the reward opportunities.

Other cases: In all other situations where the Snake simply continues to move without eating an apple or colliding, the reward is neutral, i.e. 0.0. This neutrality means that the action neither significantly contributed to nor detracted from the AI's performance. This allows the AI to understand that these movements have no direct impact on its success or failure, and that it should concentrate on more meaningful actions such as avoiding obstacles and eating apples.

**Environment design**

Environment design The game environment was configured so that Snake could only perceive the three blocks directly next to him: one block straight ahead, one to the left, one to the right. This information was encoded as input for the neural network,

enabling the AI to make decisions based solely on this restricted vision.

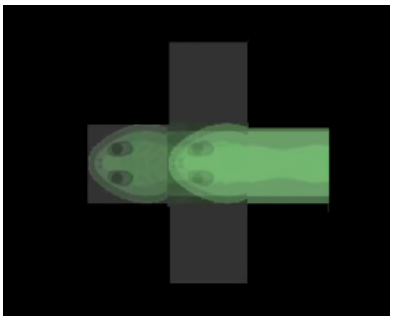

**Figure 1.** Snake-vision

**Limits and problems encountered**

However, this approach soon revealed its limitations. One of the major problems we observed was that the Snake tended to wrap around itself, resulting in frequent failures. Here are some details on the causes and consequences of this behavior:

Restricted vision: With its perception limited to the three blocks in front of it, the AI couldn't anticipate longer-term movements. It often made optimal short-term decisions without considering the Snake's future position, leading to situations where it wound itself up.

Lack of Planning: Limited vision prevented the AI from effectively planning its movements to avoid getting stuck. As a result, the Snake often ended up in a position where it no longer had safe movement options, resulting in defeat.

Ineffective training: The Deep Q-Learning model, despite numerous iterations of training, was unable to reliably learn to avoid this problem. Experiments showed that the Snake could survive longer by chance, but eventually failed systematically by winding up.

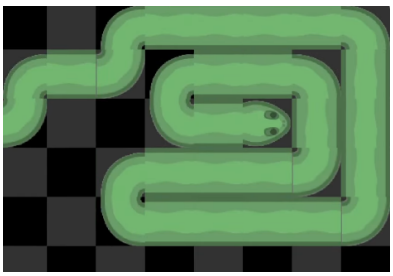

**Figure 2.** Snake-problem

### *3.2 Second Approch*

Our second approach focuses on the Snake's ability to perceive the whole game map, enabling it to make more informed and strategic decisions.

**Enhanced Neural Network**

For this second approach, we modified the neural network so that it could process information from the entire game map. Instead of receiving input limited only to the immediate blocks in front of the Snake, the new model takes as input a complete representation of the state of the game, i.e. the entire layout of blocks, apples and walls. This approach gives the AI a complete view of its environment, enabling it to anticipate future moves and avoid potential pitfalls.

**New Reward Function**

In addition to modifying the Snake's perception, we've also enhanced our reward function to incorporate a more sophisticated update model. The new reward function is defined by the following formula:

$$R = r + \gamma \cdot \max(R(s'))$$

This formula takes into account not only the immediate reward, but also potential future rewards, weighted by the discount factor. Using this approach, AI learns to evaluate its actions not only in terms of immediate gains but also by anticipating future gains, encouraging more strategic and sustainable behavior.

## 4 Learning

The learning phase involves training an AI agent to play the Snake game using Deep Q-Learning. The objective is to optimize the agent's decision-making policy by adjusting the neural network's weights based on the rewards received during gameplay. The process of learning is carried out through a combination of state representation, action selection, reward calculation, and network updates.

### *4.1 Reward*

The reward function is critical in guiding the agent's learning process. In the context of the Snake game, the reward is calculated based on the outcome of the game and the agent's actions. The `get_reward` function determines the reward as follows:

- If the game is over and the snake has not occupied all grid cells, the agent receives a penalty of -2.0.
- If the game is over and the snake has occupied all grid cells, the agent receives a reward of 1.0.
- If the snake eats the food, the agent receives a reward of 1.0.
- In all other cases, the reward is 0.0.

This reward structure encourages the agent to maximize its length by eating food while avoiding collisions with the walls or its own body.

### *4.2 Exploration*

Exploration is the process by which the agent tries out new actions to discover their effects, which helps it learn more about the environment. In the code, exploration is managed through an epsilon-greedy strategy, where the agent occasionally selects random actions. This randomness is controlled by the epsilon parameter, which determines the probability of choosing a random action over the one predicted by the neural network. Exploration ensures that the agent does not get stuck in suboptimal strategies and has the opportunity to discover potentially better actions.

To enhance the efficiency of the exploration phase, the learning rate for selecting random actions is gradually decreased. Initially set to favor high randomness, the learning rate allows the agent to explore a wide range of actions, avoiding early convergence on suboptimal policies. As learning progresses, this rate is systematically reduced, transitioning the agent from a learning phase dominated by exploration to a more autonomous phase where actions are chosen based on accumulated knowledge rather than random selection.

### *4.3 Exploitation*

Exploitation refers to the agent using its current knowledge to choose the action that maximizes the expected reward. This is

also managed by the epsilon-greedy strategy, where the agent selects the action with the highest predicted Q-value most of the time. As the agent learns and improves its policy, the epsilon parameter can be gradually reduced to favor exploitation over exploration, ensuring that the agent uses the best-known strategies more frequently.

By balancing exploration and exploitation, and by carefully managing the transition from high to low learning rates for random actions, the agent can effectively learn and improve its performance in the Snake game. This approach helps the agent to continually adapt, maximizing its rewards over time while becoming increasingly autonomous.

## 5 Result

The graphs below show the results obtained when training our agent with Deep Q-Learning for the Snake game.

The graph on the left shows the loss over time. Here's what we can deduce:

Loss decreases overall as training progresses, from around 1.4 to around 0.4. This indicates that the model is getting better at predicting optimal actions over time. Although the general trend is downwards, there are noticeable fluctuations in losses at each epoch. These fluctuations may be due to the AI's continuous exploration of new strategies, or to variations in game configurations.

Towards the end of training (after around 70 epochs), the loss seems to stabilize at around 0.4, suggesting that the model has reached a certain level of competence and can no longer learn from touching.

At the start of training, the snake's size remains relatively stable at around 4.0, indicating that the agent is not significantly improving its ability to consume apples and avoid entanglements.

From around 30 epochs onwards, there is a gradual increase in snake size. This shows that the AI is beginning to better understand how to navigate the game efficiently.

Towards the latter eras, there is an increase in snake size, reaching sizes of over 6.5. This indicates that the AI is no longer able to learn and is getting stuck.

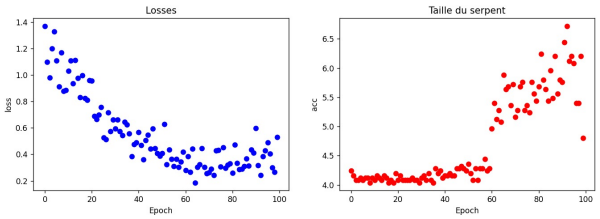


**Figure 3.** loss and snake's size over epoch with the best model

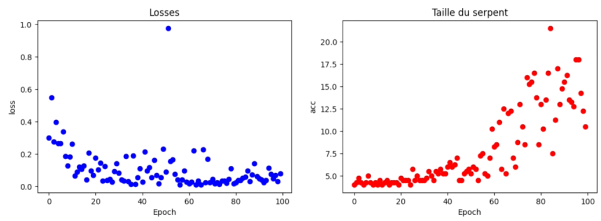


**Figure 4.** loss and snake's size over epoch with the best model

The graph on the left shows the evolution of losses over the different training periods. There is a general decrease in losses over time, indicating that the model is learning to optimize its decisions. However, variations persist, with occasional peaks, particularly around epoch 60, which could be attributed to the agent's exploration of new strategies.

The graph on the right shows the evolution of the snake's size over the eras. There is a gradual increase in the snake's size, particularly after epoch 50. This positive trend shows that the agent is improving its ability to survive longer and eat more apples, which translates into increased growth.

The results confirm the effectiveness of our Deep Q-Learning approach for training an agent to play the Snake game. The reduction in losses and the increase in the size of the snake over time testify to the significant progress made by the agent in optimizing its strategies.

## 6 Position with respect to the existing work

### *6.1 Classic Q-Learning and DQL approaches*

Work by Zhepei Wei et al (2018) and Alessandro Sebastianelli et al (2021) applied Q-Learning and Deep Q-Learning (DQL) to develop autonomous agents playing Snake. These studies showed that agents can learn to avoid collisions and maximize their score by using neural networks to approximate Q.

Traditional approaches have often encountered problems of long-term planning and restricted vision, leading to sub-optimal performance. For example, in previous studies, agents had a vision limited to their immediate environment, which prevented them from effectively planning their future movements.

### *6.2 Innovations and improvements to our approach*

Unlike previous approaches, our second approach enables the agent to perceive the entire game map. This global vision of the environment enables the agent to anticipate its future movements and avoid potential pitfalls, which significantly improves its performance.

We have introduced a more sophisticated reward function that incorporates a model for discounting future rewards. Using the formula

$$R = r + \gamma \cdot \max(R(s'))$$

our agent learns not only to evaluate immediate gains but also to anticipate future gains, thus fostering more strategic and sustainable behavior.

We explored various hyperparameter adjustment strategies and implemented a more robust adjustment mechanism to optimize the agent's decision policies. This includes dynamic adjustment of exploration and exploitation parameters to ensure that the agent does not settle on sub-optimal strategies, but continues to discover potentially better actions.

### *6.3 Comparison of results*

The experimental results show significant improvements on previous work. Our graphs show a progressive reduction in losses and a steady increase in snake size, illustrating that our agent is able to learn and apply more effective strategies over time.

## 7 Conclusion

In this project, we successfully implemented a Deep Q-Learning based approach to training an intelligent agent for the classic Snake game. By progressively adjusting the agent's learning parameters, we facilitated a smooth transition from a highly explorative phase to a more autonomous phase of gameplay. Initially, the agent was allowed to explore extensively, making random

moves to gain diverse experiences from the environment. Over time, as the agent's understanding of the game deepened, we decreased the randomness, thus shifting the focus towards exploiting the learned strategies to maximize performance.

This adaptive strategy not only optimized the agent's learning process but also highlighted the practical applications of reinforcement learning in creating intelligent systems capable of adapting to and excelling in dynamic environments. The gradual reduction of the learning rate for random actions ensured that the agent did not settle prematurely on suboptimal strategies and continued to improve its decision-making capabilities over time. As evidenced by the increasing length of the snake and decreasing game losses, our approach proved effective in enhancing the agent's ability to navigate the complexities of the game successfully.

## References


[1] Zhepei Wei, Di Wang, Ming Zhang, Ah-Hwee Tan, Chunyan Miao, and You Zhou. *Autonomous Agents in Snake Game via Deep Reinforcement Learning*. In 2018 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), page 1596. IEEE, 2018.

[2] Alessandro Sebastianelli, Massimo Tipaldi, Silvia Liberata Ullo, and Luigi Glielmo. *A Deep Q-Learning based approach applied to the Snake game*. In 2021 29th Mediterranean Conference on Control and Automation (MED), page 776. IEEE, 2021.