International Journal of Artificial Intelligence
p-ISSN: 2407-7275, e-ISSN: 2686-3251

Original Research Paper

A Deep Reinforcement Learning Agent for Snake Game

Md Meem Hossain[1], Akinwumi Fakokunde[1], Omololu Isaac Olaolu[1]

[1] *School of Computing, Engineering & Digital Technologies, Teesside University. Middleesbrough, Tees Valley, United Kingdom.*

Article History
Received:
21.08.2023

Revised:
23.09.2023

Accepted:
02.10.2023

***Corresponding Author:**
Md Meem Hossain
Email:
mdmeemhossainm@gmail.com

Abstract: After watching AlphaGo a Netflix documentary which presents how AlphaGo is an AI computer game developed by deep-mind technologies based on deep reinforcement learning (DRL). Since then, my interest in reinforcement learning has been growing. In this project, I will apply reinforcement learning to develop an agent to play snake game. Where Deep learning will implement a neural Network to help the agent (snake) to learn what action must take to get a state. If we describe deep reinforcement learning (DRL) model where agent interacts with an environment and chooses an action. Based on action, agents receive feedback from the environment as states (or perceives) and rewards. A state = an array with 11 input values, each input values represent a neural network that provides an output of 3 values, each one represents three possible actions the agent (snake) can take (Straight, Right Turn and Left Turn).

Keywords: Deep Reinforcement Learning, Neural Network, Snake Game.

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

1. Introduction

In recent years, games, robotics, self-driving cars, industry automation and remote sensing are utilizing Deep reinforcement learning to implement in real life [1]. In this Reinforcement Learning development of the Snake game, I will use DRL - Deep Reinforcement Learning Model Reinforcement Learning and Deep Learning (DL) combines together in place of the old, supervised approach because traditional ML (Machine Learning) algorithms require training with an input state and target label, but in this example, we do not know which the best action is to perform at each step of the game. A wide variety of computational techniques are available in the Reinforcement Learning sub-field of machine learning, which allows an agent to learn the appropriate control strategies or policy by directly interacting with its environment [2]. The deep reinforcement learning agent or DRL agent perceives the environment and gets the environment's state iteratively. Next the agents perform an action based on state and get a value function (reward) from the environment. Agent intends to achieve a better action (policy) by gaining the maximum value function. DRL allows RL to deal with problems that have high-dimensional action and state space [3]. Deep Neural Network helps to compress feature driven approximations in the Deep Reinforcement learning solution which allows representation of environment's state and action [4] [6].

2. Literature Review

Deep mind used neural networks in Atari games to recruit and train agents that eventually outperformed real players on multiple games in 2013 [5]. Policy Gradient and Deep Neural Network (DNN) can be approximated by agent through value function (reward) and policy. Some of the problems in deep neural network can be avoided using the large storage sequence space of tables and slow tables which are the parts of table store data. It used Transmission and Deep Neural Network to approximate value functions and policies. Huge storage establishments of columns and slow feedback are the issues associated with table store data, these issues can be solved using deep neural network (DNN) measurement.

2.1. PACMAN in Deep Reinforcement Learning (Q-learning)

The aim of this research is to teach the Pacman agent using deep reinforcement learning or Q-learning to operate cleverly to get higher scores by avoiding ghosts, consuming food, and being as scared of the ghosts as possible. Researchers motivated to work on this project not only because they want to do reinforcement learning and integrate a neural network on their own, but also because we think to see a trained Pacman agent is visually appealing. All of the reinforcement learning techniques they used in this project were built using the code for the Pacman game emulator. They apply various Deep Reinforcement learning methods to the traditional game Pacman [7].

2.2. Deep Reinforcement Learning in ATARI game

A new deep reinforcement learning (DRL) model is beautifully presented in this paper, and it is shown that using only raw pixel input, it can learn challenging control strategies for Atari 2600 video games [5]. In order to facilitate the instruction of deep networks for RL, an online Q-learning variant that communicated stochastic mini-batch updates with experience buffer was also presented in this research paper. Their method produced innovative results without changing the architecture or hyper-parameters in the six games out of seven those us tested.

2.3. Technology Used

This part will explain about the technologies that are used to develop the agent and environment for the snake game and to evaluate DRL model.

1. Python 3.10.1
   High-level, object-oriented programming languages are the main characteristics of Python. It is relatively concise compared to other languages like Java and has a simple syntax that helps to make its code very readable and simple to learn. It is frequently regarded as the best language for quickly developing high-quality applications.
2. PyCharm IDE (Integrated Development Environment)
   All the features necessary to build an application as rapidly as possible will be present in a good IDE. Professional-level IDE PyCharm has tools for debugging, code completion, and simple refactoring.

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

3. Pygame
   It is a collection of cross-platform Python game development tools. It provides Python programming language-compatible sound and graphics libraries.
4. Torch
   It is a Torch-based open-source library that is frequently used for Deep Learning applications on both CPUs and GPUs. It supports computational graphs at runtime and is quick and simple to use.
5. Numpy
   High performance multi-dimensional arrays and matrices are supported by NumPy, a Python package. It offers computational graph support during runtime. It offers optimization, is quick, and is adaptable.
6. Matplotlib
   The data set for machine learning projects contains a lot of information. Therefore, the Python matplotlib library allows the developer to plot a wide range of charts such as bar plots, scatter plots, histograms, and so on whenever the programmer is unable to analyse the data. Another name for Matplotlib is a library for data visualization.

3. Methodology

3.1. Reinforcement Learning

When we are young, the first thing we do is try waving our hands or crying in hopes of getting attention. We can learn about our immediate ambiances, various events and logic, action, and the outcomes, how to complete our goals by the responses we get back. Therefore, we can engage with outside surroundings in our child's development stage. It is proven that the foundation of learning from interaction invented the most intellectual theories. Reinforcement learning is considered a type of learning from interaction which can develop Artificial intelligence (AI) in the most proper way [8].

3.2. DNN - Deep Neural Network

Deep neural network consists of some nodes and layers. Input layer is known as the first in DNN where data is received from users or receiver device/function. The second layer is the hidden layers where data is transformed using activation functions and weights to transfer data to the output layer. Output layer is the end of the layer where the goal is predicted. Patterns and expand their prediction trained in the network by calculating the weights [9].

3.3. DRL - Deep Reinforcement Learning

Perception and decision making are the benefits of deep learning and reinforcement learning those developed Deep Reinforcement Learning and it has the capability to control output signals based individual input [6]. Thanks to this mechanism, which made human cognitive modes are much closer to artificial intelligence [10].

3.4. Algorithm for Reinforcement Learning

The Model, Policy, Action, State, and Operator that each algorithm employs can be used to make a rough comparison between them. There are several reinforcement algorithms include can be used to develop an agent to play snake game.

3.4.1. Monte Carlo (MC)

The Monte Carlo method is an easy and fast-going concept where an agent communicates with the environment and learns about states and rewards. In this concept, agent observed the environment and get a state from the environment and calculate the state to take a move or action based on the average feedback [12], [13]. If we look at Monte Carlo (MC) model, Reward is only achieved at a specific large stage where previous stage state and action are considered as good to enhance the reward mechanism of the previous stage state.

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

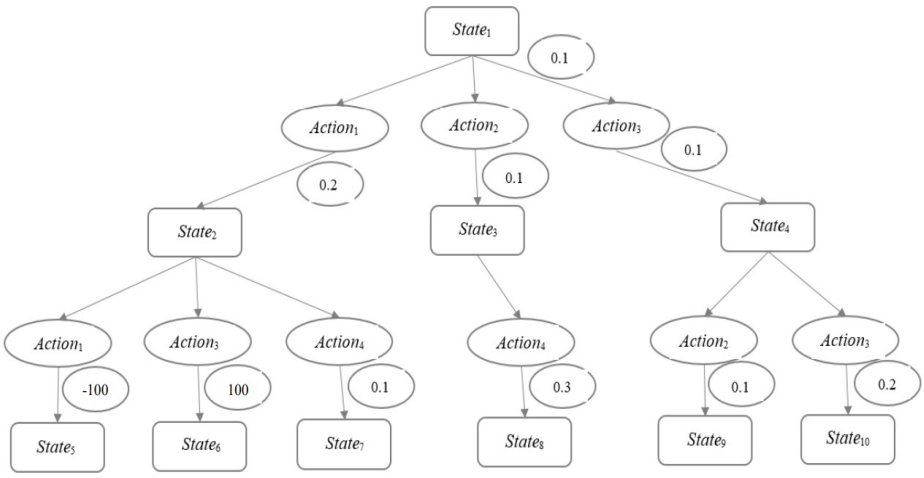

Figure 1. Monte Carlo Model (**Source Wu, C., 2019)

3.4.2. Markov decision process (MDP)

The Markov methodology is a procedure which considers the present state more than anything. In Figure 2, the Markov decision process prioritized the thinking of a scenario. In this model, ACTION = a0, a1 in the figure, STATE = S0, S1, S2, in the figure and reward will be given for each state are +100 and -100 (curve turn arrow in the figure) [14], [15].

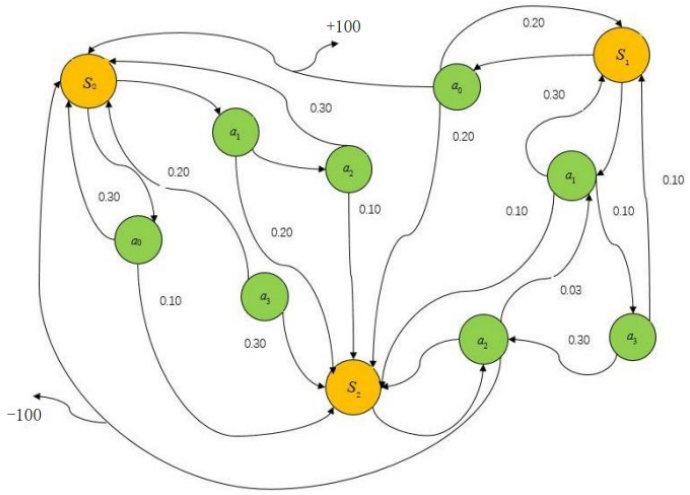

Figure 2. Markov Decision Process Model (**Source Wu, C., 2019)

3.4.3. Neural Network layers in Deep Reinforcement Learning

Deep neural network consists of some nodes and layers. Input layer is known as the first in DNN where data is received from users or receiver device/function. The second layer is the hidden layers where data is transformed using activation functions and weights to transfer data to the output layer. Output layer is the end of the layer where the goal is predicted. Patterns and expand their prediction trained in the network by calculating the weights [9]. Perception and decision making are the benefits of deep learning and reinforcement learning those developed Deep Reinforcement Learning and it has the capability to control output signals based individual input [6].

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

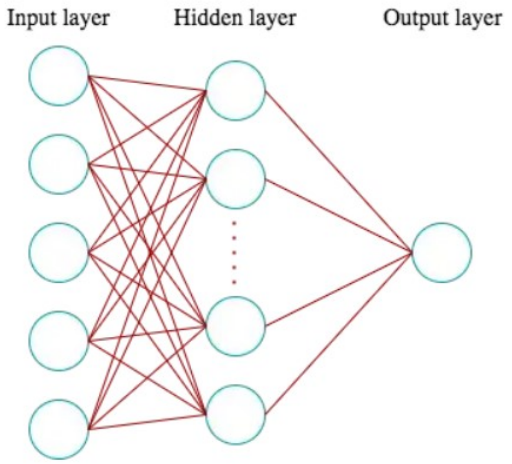

Figure 3. Neural Network in DQN (**Source https://towardsdatascience.com/)

3.5. Problem Formulation

Snake game is a very common 2D game which has three elements snake, boundaries and food and the game is played by a human. On a bordered plane, the player gets to control a dot, rectangle, or object. When the agent (snake) hits the screen boundary, a path or other obstruction, or itself, the player loses [16]. In this situation or problem our goal is to create an agent using Deep reinforcement learning which will help the snake to play the game and make scores.

3.5.1. Model

To solve the mentioned problem to create a Deep reinforcement agent to play snake game. Deep Reinforcement Learning or DRL model will be developed to control or play the whole game. Where reinforcement learning and deep learning combine together to become deep reinforcement learning (DRL). In DRL, Reinforcement learning (RL) helps to handle problems by high dimensional state and action in Reinforcement Learning (RL) assist to handle problems, other-hand high dimensional state and action transformed in a low dimensional function by helping of deep neural network (DNN) [17].

In my Deep Reinforcement Learning DRL model, Snake the agent interacts with an environment and chooses an action. Based on action, agents receive feedback from the environment as states (or perceives) and rewards. A state = an array with 11 input values in input layer, each input values represent a neural network that provides an output of 3 values in output layer, each one represents three possible actions in output layer the agent (snake) can take (Straight, Right Turn and Left Turn). Below I have shown the state, action, and rewards in my snake game model.

Table 1. Agent Component Represent

States		
Danger straight		
Danger right		
Danger left		
Direction left	**Policy**	**Value Function**
Direction right	Straight	Eat food +1
Direction up	Right turn	Game over -1
Direction down	Left turn	Else 0
Food-left		
Food-right		
Food-up		

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

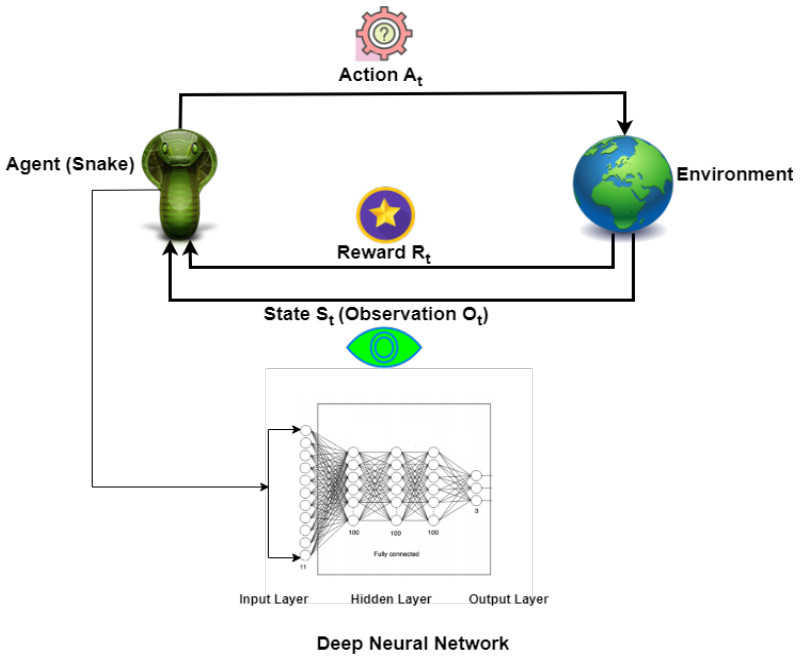

Figure 4. Snake Game Model Diagram

3.5.2. Agent Components

3.5.2.1. Agent State

The agent state is a reflection of the snake game that the agent can comprehend; it contains the crucial data that the agent must process in order to decide what action to take. Understanding how to use. The state is significant because it affects how well the agent performs and how quickly it processes information. If we provide more relevant data, the agent will perform much better but process information more slowly. An array of 11 values that contains the state's implementation in this model provides the agent with the data it requires to process and foresee the best course of action. Each value, which can either be 0 or 1, stands for a different thing.

Table 2. 11 States Input

No	State	Value
1	Danger straight	0 or 1
2	Danger right	0 or 1
3	Danger left	0 or 1
4	Direction left	0 or 1
5	Direction right	0 or 1
6	Direction up	0 or 1
7	Direction down	0 or 1
8	Food-left	0 or 1
9	Food-right	0 or 1
10	Food-up	0 or 1
11	Food-down	0 or 1

3.5.2.2. Policy

The moves are those that the agent may make in light of a specific state. The agent can only make three movements in this model: a right or left turn, and a straight line. Three values in an array, ranging from 0 to 1, can be used to represent these actions. The probability that each action will be the best choice is represented by each value.

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

Table 3. Three Policy or Actions

No	Policy
1	Right turn
2	Left turn
3	Straight

3.5.2.3. Value Function

The moves are those that the agent may make in light of a specific state. The agent can only make three movements in this model: a right or left turn, and a straight line. Three values in an array, ranging from 0 to 1, can be used to represent these actions. The probability that each action will be the best choice is represented by each value.

Table 4. Three Value Functions or Rewards

No	Function	Value
1	Eat food	+1
2	Game over	-1
3	Else	0

3.6. Agent and Environment

Based on pygame, we create our own snake environment. The setups in our environment vary. For instance, we can decide whether the snake can cross a boundary or not, the size of the environment, and whether there are any obstacles to the surroundings. By incorporating a few Methods into the Snake game, a Reinforcement Learning environment can be implemented.

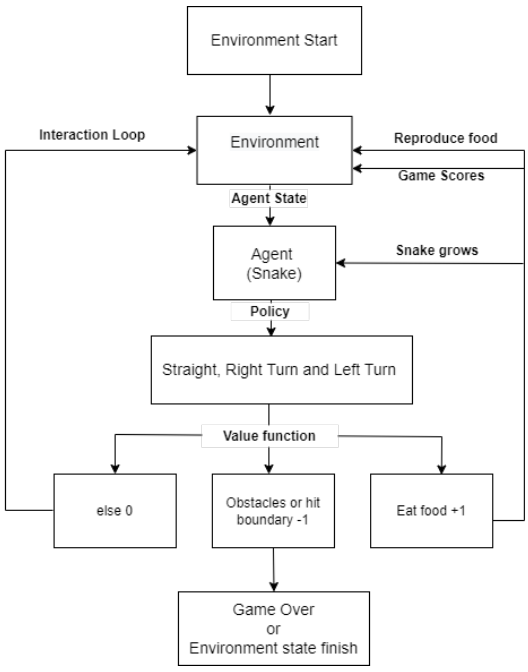

Figure 5. Snake Game Flowchart Diagram

Foods are the snake's reward. Red color object in the environment is identified as the food in the environment. The food stays in the same position until it is eaten by the snake, at this point food appears again in a different location. If the snake is successful in eating the food, add one to the score. The little snake will develop after eating the food, extending its body.

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

4. Finding and Discussion

4.1. Finding

The research resulted in the development of a trained DRL model. It consisted of 3 different layers where the first one had 11 nodes, second layer is a hidden layer, and the last layer is an output layer with 3 nodes. I used training and validation sets to easily measure the performance of my model, I collected the training and validation sets during evaluation and training time. Total rewards in the game episodes collected by the agent are considered evaluation indication. The graph in Figure 6. Shows the first few games where the agent has no idea what to do. So, it was trying to explore the environment and make some random moves. The rewards at this level were exceptionally low.

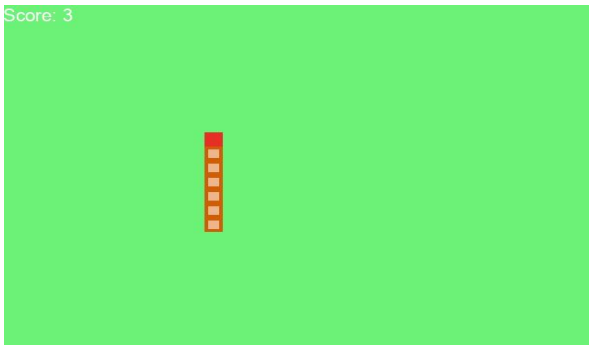

Figure 6. Snake Location Initial Rounds

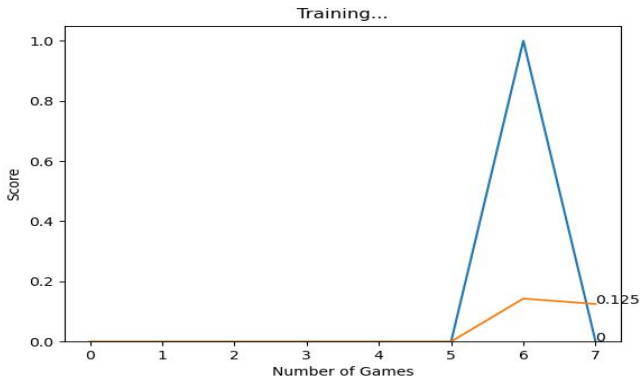

Figure 7. Rewards in Demographic Initial Rounds

Next the graph in figure 9 shows peak episodes of games where the agent learns all the strategy. And this time agents achieve higher rewards. This is because agents can avoid any obstacles such as avoiding their own body, not to hit boundaries and eat the food as much as possible to get rewards.

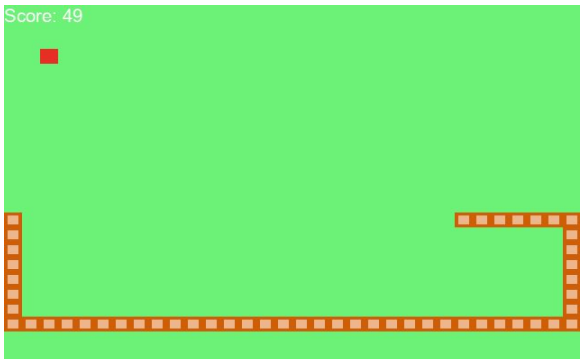

Figure 8. Snake Location Peak Rounds

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

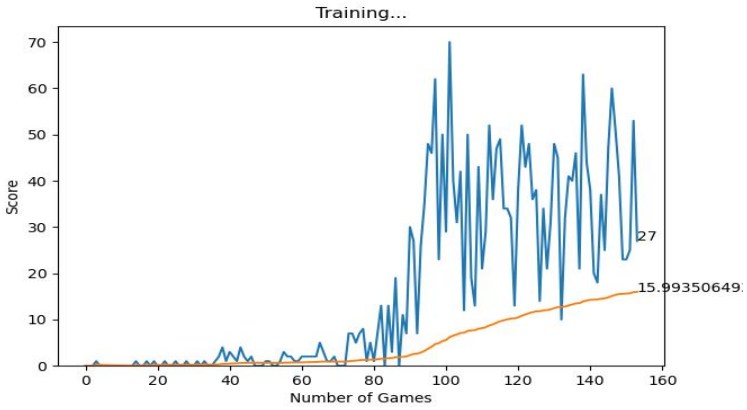

Figure 9. Rewards in Demographic Peak Rounds

4.1. Discussion

4.1.1. Agent Play with State

State parameters affect the behavior of the agent. Agent could achieve a similar or better performance by changing state space instead of playing experience replay. There are four state spaces that can apply. First, "no direction" where the agent will not get any direction. Second, "coordinates" where the food location maybe change with food coordinate (x, y) or the snake (x, y). Third, "direction 0 or 1" which is the actual state used in the model. Last but not least, "only wall" where agent will notify by the walls no other direction.

4.1.2. Agent Play with Rewards

The aim of the snake or agent is to reach food by taking the shortest path in the most logical way. There are a few rewards systems that I used in reward space.

1. Simple or Current Reward

Table 5. Simple or Current Reward

No	Rewards	Value
1	Eat food	+1
2	Game over	-1
3	Else	0

2. Surviving Reward

Table 6. Surviving Reward

No	Rewards	Value
1	Eat food	+1
2	Game over	-1
3	Snake survives	+2

By this reward system, snake will walk without getting killed. And snake will make more score by walking than eating the food

3. Increase Reward Value

Table 7. Increase Reward Value

No	Rewards	Value
1	Eat food	+10
2	Game over	-1
3	Snake survives	+2

By this reward system, snake could find the shortest path to eat food. But it has more chance to bite itself and die.

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

4.1.3. By Experience

Experience replay is the basis for the agent's learning (only 30 games are required). In this concept, agents acquire knowledge by experiencing and make use of them to pick up new information more quickly. Multiple replay steps are carried out at each regular step (batch size parameter). Snake benefits from this because the reward and subsequent state are relatively consistent when the same state-action pair is used.

5. Conclusion

In this study, I tried to present a Deep Reinforcement Learning model to coach an agent how to play the snake game. Reinforcement learning (RL) and deep neural network (DNN) related literature review has been shown and discussed. Compared to other solutions in literature review Deep Reinforcement Learning (DRL) was proved as a straightforward model which is very easy to train. In the snake game, agents learn to play games and achieve high scores time by time where the agent achieves high scores 46 and 52 after playing 140 games. It was pretty satisfying compared to another agent.

There is a limitation or problem in our solution. The agent does not learn to avoid scrolling or biting itself. It learns to avoid any obstacle in front of the snake head, but it cannot see the whole game. When the snake is longer, the agent will scroll itself and die. In consequence, the highest score of the game is always confined to a certain number.

To solve the problem stated in the limitation part, I can implement state space using Convolutional Neural Network and pixels. Using pixels and CNN (Convolutional Neural Network) agent could see the whole game instead of seeing just obstacles only [14]. So that the agent could recognize his location to avoid scrolling itself and it can achieve higher scores. In my future works I will try to explore to combine Reinforcement Learning and Convolutional Neural Network (CNN) to create an optimal model to play snake game.

References

[1] S. Kuutti, R. Bowden, Y. Jin, P. Barber and S. Fallah, “A survey of deep learning applications to autonomous vehicle control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712-733, 2020.

[2] R. S. Sutton and A.G. Barto. (2018). Reinforcement learning: An introduction. MIT press.

[3] K. Arulkumaran, M.P. Deisenroth, M. Brundage and A.A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26-38, 2017.

[4] D.P. Bertsekas, “Feature-based aggregation and deep reinforcement learning: A survey and some new implementations,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 1-31, 2018.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[6] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup and D. Meger, “Deep reinforcement learning that matters,” *In Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, April 2018

[7] A. Gnanasekaran, J. F. Faba and J. An. Reinforcement learning in Pacman. 2017.

[8] C. Szepesvári, “Algorithms for reinforcement learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1-103, 2010.

[9] Y. Patil, Snake Game Using Reinforcement Learning.

[10] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[11] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, G. Xu, H. Li, and X. Liang, “UAV autonomous target search based on deep reinforcement learning in complex disaster scenes,” *IEEE Access*, vol. 7, pp. 117227-117245, 2019.

[12] B. O'Donoghue, R. Munos, K. Kavukcuoglu and V. Mnih, “Combining policy gradient and Q-learning,” *arXiv preprint arXiv:1611.01626*, 2016.

International Journal of Artificial Intelligence, vol. 10, no. 2, pp. 92-102, December 2023. **DOI: 10.36079/lamintang.ijai-01002.565**

[13] D.J. Soemers, C.F. Sironi, T. Schuster and M.H. Winands, "Enhancements for real-time Monte-Carlo tree search in general video game playing," *In 2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1-8, September 2016.

[14] L. Xia, J. Xu, Y. Lan, J. Guo, W. Zeng and X. Cheng, "Adapting Markov decision process for search result diversification," *In Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval,* pp. 535-544, August 2017.

[15] R. Berthon, M. Randour and J.F. Raskin, "Threshold constraints with guarantees for parity objectives in Markov decision processes," *arXiv preprint arXiv:1702.05472*, 2017.

[16] R. Cai and C. Zhang. Train a snake with reinforcement learning algorithms. 2020.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski and S. Petersen, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529-533, 2015.

[18] Z. Wei, D. Wang, M. Zhang, A.H. Tan, C. Miao and Y. Zhou, "Autonomous agents in snake game via deep reinforcement learning," *In 2018 IEEE International conference on Agents (ICA),* pp. 20-25, July 2018.