Introduction

Reinforcement learning is a machine learning training method where desired behavior is rewarded while undesirable behavior is penalized. These are called positive and negative reinforcement respectively. A reinforcement learning agent is capable of observing and interpreting its environment, acting then receiving a reward or penalty based on its action. A reinforcement learning agent is said to learn through trial and error.

In this paper, we will be focusing on the topic of positive reinforcement by using the game Brick Breaker as the environment. In Brick Breaker, a player bounces a ball to destroy bricks by moving a paddle left and right. The player loses if the ball leaves the screen and wins if all bricks are destroyed.

Q-Learning is the reinforcement algorithm utilized in this paper. The foundation of the code was written by Matthew Chan and is available at medium.com. The code has been modified to work with the game.

Genetic algorithm will also be employed in this paper and is used to determine the hyperparameters of the Q-Learning agent. Different techniques such as single/double crossover and tournament/roulette selection are used to ensure more variety during the evolution process to produce better results.

This paper aims to study the effects of different improvements of Q-Learning on the agent performance. Combinations of these improvements will then be tested to determine the best combination and get the best possible results. The effects of varying the game settings on the agent performance will also be studied and documented.

A GUI was also created for users to more easily change the settings of the project and streamline the process of hyperparameter tuning and data collection.

Literature Review

The main innovation in this paper is to combine existing improvements of the Q-Learning agent to produce a new, better solution. In the end, 7 research papers of interest were identified and listed below.

Firstly, **genetic algorithm** was used by Wicaksono, A. S., & Supianto, A. A. for hyperparameter optimization of the machine learning methods used in online news popularity prediction.

Secondly, H.R. Tizhoosh has introduced **opposition-based learning** as a new scheme for machine intelligence.

Thirdly, the **granularity of the state space** was shown to affect the results by Jacopo Fior and Luca Cagliero in their study of machine learning-based stock trading.

Furthermore, Michal Gregor and Juraj Spalek have also done research on optimistic exploration value functions. This has been adapted in this project as the **random initialization of the Q-tables**.

Moreover, research into action elimination with deep reinforcement learning has also been done by Tom Zahavy, et al. This has been adapted to varying the **size of the action space** in the project.

Apart from that, research into **different reward functions** on the training performance of a Double DQN has also been done by Stefan Šćepanović.

Finally, Double Q-Learning was introduced by Hado van Hasselt which lacks the overestimation bias of the Q-Learning algorithm. This has adapted to **N-tuple Q-Learning** which is another innovation in this paper where N Q-tables were created for each reinforcement learning agent.

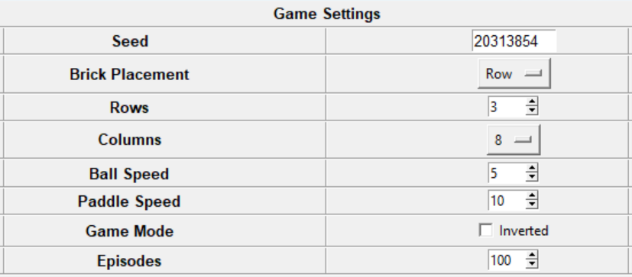
Thus, we can study the effects of combining multiple improvements and ascertain the best combination with the best results.

Graphical User Interface (GUI)

As mentioned above, users can easily change the settings of the project using the GUI created. The different settings of the GUI will be explained below.

## Game Settings

The game settings allow users to customize the different aspects of the game. The different options available to the users are shown below in Figure 1.



***Figure 1: Game Settings***

* **Seed**

The seed is the most important part of the settings. It controls all the randomness in the program and allows us to produce the same results provided the same seed is used. The default seed chosen is 20313854.

* **Brick Placement**

Brick placement controls how the 3 different brick types are arranged in the game. There are 3 options available which are Row, Column and Random.

Row alternates the different brick types by row starting with the strongest to the weakest. Column does the same but alternates it by column. Random produces a random arrangement of the 3 brick types.

* **Rows**

Rows controls the number of brick rows generated in the game. The default number of rows is 3 but it can vary from 1 to 10.

* **Columns**

Columns control the number of bricks in each brick row. The default number of columns is 8. However, the number of columns can only be changed to a factor of 600 up to the value of 15. So, the available options are 1, 2, 3, 4, 5, 6, 8, 10, 12 and 15.

* **Ball Speed**

Ball speed allows us to control how fast the ball moves in the game. The default ball speed is 5 but it can vary from 1 to 10.

* **Paddle Speed**

Paddle speed is how fast the paddle moves in the game. The default paddle speed is 10 but it can vary from 1 to 20.

* **Game Mode**

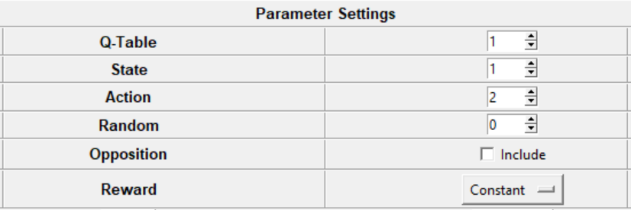
Game Mode allows users to invert the position of the paddle and the bricks. It allows the game to be played upside down.

* **Episodes**

Episodes control the number of failed episodes before a game is reset and is recorded as a failed run. The default value is 200 but it can vary from 100 to 500.

## Parameter Settings

The parameter settings allow users to control the different settings of the Q-Learning agent. The different options available are shown below in Figure 2.



***Figure 2: Parameter Settings***

* **Q-Table**

Q-table controls the number Q-tables generated for each agent. It is based on the concept of N-tuple Q-Learning which is inspired by Double Q-Learning. The default number of Q-tables is 1 but it can vary from 1 to 10.

* **State**

State allows users to control the granularity of the state space of the Q-tables. The agent is assigned a state based on the distance between the paddle and the ball.

The state space can only ever be a multiple of 2. This is to represent the 2 possibilities where the paddle is to the left and to the right of the ball. So, a value of 1 actually indicates 2 Q-tables and so on. The default value is 1 but it can vary from 1 to 10.

* **Action**

Action allows users to customize the size of the action space. There are only 2 options available which are 2 and 3. The 2 default actions are for the paddle to move left and move right while the 3rd action is for the paddle to do nothing. The default value is 2.

* **Random**

Random allows users to initialize the Q-tables with random values. There are 3 distinct choices where the chosen value is positive, negative or zero.

A value of 0 indicates that the Q-tables are to be initialized with 0s. A positive value means that the Q-tables are to be initialized with values drawn from a normal distribution in the range -n and n where n is the chosen positive value. A negative value has the same range but the values are instead drawn from a uniform distribution.

The default value is 0 but the value can range from -5 to 5.

* **Opposition**

Opposition allows users to make use of the concept of opposition-based learning. By including this option, the agent will explore both the chosen and opposite actions and update Q-tables for both results. However, the action of do nothing has no opposite action and is updated only once.

* **Reward**

Reward allows users to modify the reward function of the agent. There are 5 different reward functions available which are Constant, Time-Based, X-Distance, X-Distance (Center) and XY-Distance.

Constant returns a constant reward of 1 for each turn the agent has not lost the game. If the ball leaves the screen, a reward of 0 is given. This is the default option.

Time-Based returns a reward based on the number of turns the agent has survived in each episode. The longer the game goes on, the higher the reward given.

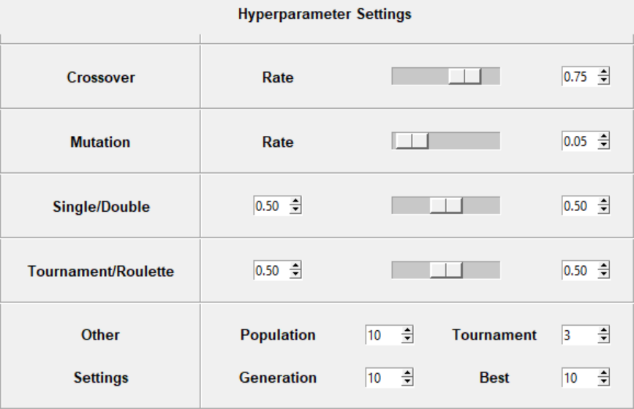
X-Distance returns a reward based on the horizontal distance between the midpoints of the ball and paddle. The shorter the distance, the higher the reward given.

X-Distance (Center) is similar to X-Distance in which the horizontal distance is used to calculate the reward. However, the entire paddle is used instead of just the midpoint. So, a ball anywhere above the paddle will receive the maximum reward. If the ball is to the left of the paddle, the left side of the paddle is used to calculate the reward and vice versa.

XY-Distance is similar to X-Distance in which the midpoints of the ball and paddle are used to calculate the reward. However, the Euclidean distance between the two is used to calculate the reward instead of just using the horizontal distance.

## Hyperparameter Settings

The hyperparameter settings allow users to control the process of hyperparameter tuning of the agent. It is accessed by pressing the tuning button. The options available are shown below in Figure 3.



***Figure 3: Hyperparameter Settings***

* **Crossover Rate**

Crossover rate allows users to control the percentage of chromosomes that perform crossover to produce new chromosomes. The default crossover rate is 75%.

* **Mutation Rate**

Mutation rate controls the percentage that a gene in each chromosome will undergo mutation and turn into a new gene. The default mutation rate is 5%.

* **Single/Double**

Single/Double controls the process of crossover. The left percentage denotes the chance that a successful crossover is single crossover while the right percentage is the chance for double crossover. The default chance for both events are 50%.

* **Tournament/Roulette**

Tournament/Roulette controls the process of selection. The left percentage denotes the chance that tournament selection is used while the right percentage is the chance for roulette selection. The default chance for both events are 50%.

* **Population**

Population controls the number of chromosomes of each generation. The default population size is 10 but it can range from 2 to 100. The population size can only be a multiple of 2 since adjacent chromosomes are used in the process of crossover to produce new chromosomes.

* **Tournament**

Tournament controls the number of chromosomes selected for a tournament in tournament selection. The default value is 3 but it can range from 1 to 100. If the tournament size is greater than the population size, the fittest chromosome will be chosen each tournament.

* **Generation**

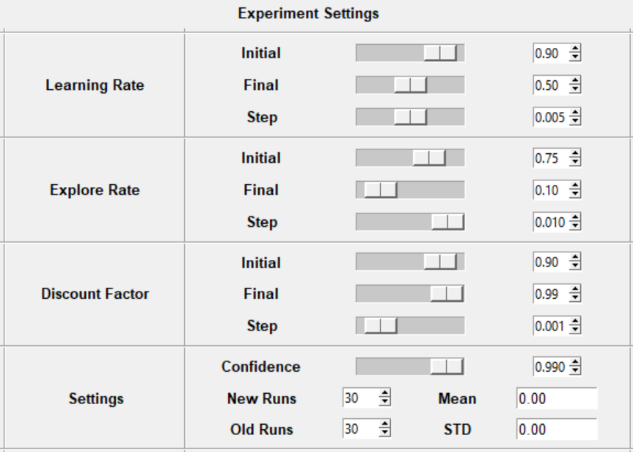
Generation controls the number of iterations of genetic algorithm. The default value is 10 but it can range from 1 to 100.

* **Best**

Best controls the number of the fittest chromosomes displayed at the end. The default is 10 but it can range from 1 to 10.

## Experiment Settings

The experiment settings allow users to control the process of comparing different parameter settings. It is accessed by pressing the experiment button. The options available are shown below in Figure 4.



***Figure 4: Experiment Settings***

* Learning Rate
* Explore Rate
* Discount Factor
* Confidence
* New Runs
* Old Runs
* Mean
* STD