IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 11, Issue 5, May 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105013 |

Breakout Ball Game

Gauri Vinod Kadam, Arya maroti Dasarwad

Department of Computer Engineering, Marathwada Mitra Mandal Polytechnic, Pune, India

Department of Computer Engineering, Marathwada Mitra Mandal Polytechnic, Pune, India

ABSTRACT: Reinforcement learning (RL) algorithms, which utilize real task experience to generate a progressive management policy, are typically referred to as trial and error learning techniques. The reinforcement learning theory is a psychological perspective on how agents might exert the most control over their environment.

The primary distinction both supervised learning and reinforcement learning is that partial feedback on the events that were learnt is given to the learner. An RL agent gradually practices and develops a strategy for long-term rewards by becoming able to map states to the best route of action through trial and error.

KEYWORDS: ball, bricks, paddle, Intelije Idea Score

I. INTRODUCTION

Reinforcement learning (RL) is the name given to focus on goals algorithms that discover how to carry out a particular task or how to optimize over multiple runs along a dimension; for instance, over numerous moves, they can optimize the number of points won in a game. Under the correct situations, RL algorithms are capable of superhuman performance from a starting point.

Such algorithms are reinforced when they choose the proper course of action and penalized when they choose the wrong one, much like a pet who is encouraged through correction and punishment. When the state action space is too big to be entirely known, RL performance suffers. In order to accomplish this, we are utilizing Deep Reinforcement Learning.

Better effectiveness.

Artificial neural networks and RL architecture are combined in deep reinforcement learning, which enables software-defined agents to discover the optimum behaviors in a virtual environment to accomplish their goals. We have trained a neural network on state-action space samples to learn to decide how essential those are in relation to our goal of boosting learning instead of utilizing a lookup table to store, index, and update all conceivable states and their values, which is challenging with very large situations.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105013 |

A. Basic Definitions

Understanding the notions of agent, state, action, environment, and rewards can help us better understand reinforcement learning.

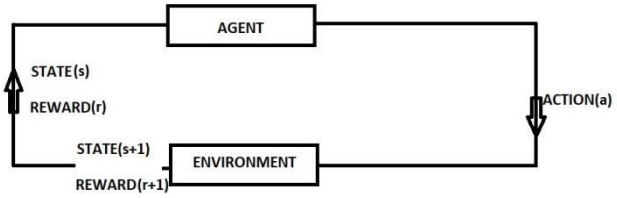


Figure 1 illustrates that the main elements of the reward learning process are the agent and the environment. An agent is a thing that acts (has a set of options for actions) in a context (a setting that the agent experiences and that reacts to it). A state is a situation that the agent is actually in right now, meaning it's a place or a point in time that determines how the agent's Q-value will change. A reward is also given to the agent, which we use to determine whether their action was successful or unsuccessful in each state, and it can be either positive or negative, which has an effect on the Q-value.

An agent responds to the environment by acting upon any given state, and the environment responds by returning the new state of the agent along with any incentives that may have been received. Rewards may be delayed or given right away.
In reality, they control the handler's behavior. Another word related to rewards is the discount factor, which can be calculated using future incentives that the agent has learned about in order to reduce the impact of such rewards on the agent's decision-making process.

Paddle acts as an agent in our bespoke breakout game setting, which also features a wall, bricks, and a ball. The agent (paddle) has three possible actions: moving the paddle left, moving the paddle right, or leaving the paddle idle. States of the game include whether it is in progress, lost, or won, as well as the ball's x and y coordinates, velocity, paddle's x position, array of the remaining bricks' coordinates, number of frames since the game began, and current score.

The items in our specially designed breakout space include a paddle, a ball, a wall, and a block made of bricks in various colors. A ball that strikes the paddle earns the agent 3 points; if it misses the paddle, the agent loses 3 points. Rewards are given to the agent in accordance with the color of the brick when a ball strikes it (blue=8, green=7, yellow=5, orange=4, red=3). Each (state, action) pair's Q-value will be given to the network, which will aid in learning. The action with the highest Q-value will be chosen to advance the game after obtaining Q-values for each unique (state, action) pair that will be kept in the network.

B. Asserting Thesis

A crucial aspect of DQNs is the "memory" since, as was already mentioned, the trials are used to continuously train the model. However, we add the trials to memory and train them on a random sample of that memory rather than training on the trials as they come in.

This depreciated value for the anticipated future returns on the state is reflected in the gamma factor. Value specified for but will range from 0 to 1. Since the paddle's behavior will be fully random and will eventually start to decrease with subsequent iterations, we have followed the Epsilon Greedy Search algorithm, in which the value of gamma starts out at 1 and decreases to 0.95. The DQN and DDQN algorithms will calculate the Q-value for each different (state, action) combination, and that Q-value will be utilised to determine the optimum action the paddle should do for the following state (velocity, ball coordinates).

Since the environment of Atari games is extremely uncertain when considering the states and actions connected to the environment, which makes it applicable to real-life situations, we decided to implement Deep Reinforcement Learning on Atari games.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105013 |

II. PROBLEM DEFINITION

The goal of our project is to develop a virtual environment for the video game "Breakout" that can be demonstrated to be a near replication of the real-life world. The virtual environment will learn the real-life environment over time and will act accordingly. The environment must be appropriately learned in order to replicate a real-world setting and respond to changes. To do this, we will use the aforementioned Reinforcement Learning algorithms and demonstrate which one is more effective in such circumstances.

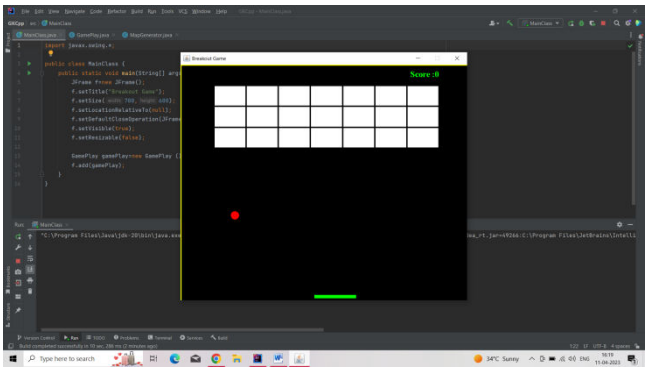
First, the issue that we looked into was reinforcement learning. A subset of machine learning called reinforcement learning is used to define and address the issue of how an agent interacts with its surroundings by developing learning techniques that maximize reward [1]. The Markov Decision Process (MDP), which is basically a process where an agent takes action to update its state to get reward and interact with environment, is a classic and standard model of reinforcement learning.

interact with your surroundings [2].
The game itself is the second issue. Our game is a parody of Breakout. A moving ball, a paddle and six rows of bricks are the game's building blocks. To hit the ball, the agent, or paddle, can move left or right. When the ball is struck, it will rebound and then smash the bricks to get the desired result. This game must be capable of responding to input and moving on to the following frame. If the ball is missed, the game will finish, and the ball and paddle will then be placed back in their original locations.

After the ball touches the bricks, they will vanish. The ball and paddle are first positioned, and all six rows of bricks are loaded, at the start of the game. The total number of bricks damaged, the number of times the ball was missed, and the hit rate of the ball are all displayed in the game.

The third issue is to construct a network structure that compares DQN with DDQN in terms of reward acquisition, loss at each epoch, and hit rate in our simulated Breakout environment in order to determine which method works better.

Design of Gui



III. RELATED RESEARCH WORK

The introduction of TD-gammon can be traced back to reinforcement learning. In 1992, IBM researcher Gerald Tesauro created an algorithm called TD-Gammon that specializes in playing backgammon and uses time difference learning and neural networks.

A three-layer neural network is used in TD-gammon. There are 40–80 neurons in the middle hidden layer, and 198 units serve as the input representation for the backgammon position. An estimation of the value function is the output's final form [3]. A multi-layer perceptron with one hidden layer was employed by TD-gammon to approximate the value function using a model-free reinforcement learning approach similar to Q-learning.

There was an accepted belief that the TD-gammon strategy only worked in backgammon as a result of the less

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105013 |

successful attempts to implement TD-gammon into various board games. This might be the case because the value function is especially smooth thanks to the dice's randomization, which also aids in exploring the state space.

A series of studies [5] proposed the pairing of deep learning and reinforcement learning methods, mostly using Q learning. The deep reinforcement learning methodology was shown to successfully handle difficult problems because it can learn from data at different levels of features, in contrast to previous reward learning approaches that struggled with feature selection. Mnih was able to successfully train a deep RL agent using inputs with thousands of pixels from visual images. Through this method, it was able to play Atari games, play Alpha Go, and other games beyond the capabilities of humans.

Mnih's Deep Q network bot simulates human-like performance when playing Atari games by processing sensory data using artificial neural networks. Later research by Van Hassel [7] enhanced the system by using double deep Q-Learning, which reduces overestimation and produces estimates that are more accurate.

Researchers showed that the deep Network agent has been able to surpass the efficiency of all prior algorithms and reach a level equivalent to that of a skilled human games tester using the same algorithm, network architecture, and hyperparameters across a variety of 49 different games in the paper "Human-level control through deep reinforcement learning" [5].

IV. METHODOLOGY

We chose to build our own environment to train the agent in, which imitates the environment of the OpenAI Atari game Breakout-ram-v0, in order to compare the performance of the AI agent in playing the breakout game based on different algorithms. Following environment setup, we constructed the networks for the DQN and Double DQN algorithms. These are the precise steps:

1) Setting up the Breakout environment, which includes the background, the paddle and the ball, defining bricks, controlling paddle movement, handling collisions, updating state and environment, and using turtle library, a Python graphics library that can be used to create a variety of objects and shapes and give them animations using the penup() function by varying the speed during object creation.
2) We have defined separate functions to define bricks, control paddle movement, handle collisions, update state and environment, and reset the environment if the paddle misses the ball. Other functions include next iteration(), which computes the parameters for the next state, move positive x(), which moves the paddle to the right, and move negative x(), which moves the paddle to the left.
3) 3) Using the Breakout environment, create the DQN and Double DQN algorithms and tune hyper-parameters (such as the discount factor gamma, learning rate, and epsilon). employing software libraries such as random, numpy, keras, collections, matplotlib, and others.

V. ACKNOWLEDGEMENT

The professors in communication Jessie Stick gold-Sarah and Michael Trice, the TAs Devon Roster, Rishi Naidu, Pranav Kaundinya, and Luis Fernandez, as well as Prof. Gim Hom for his guidance and assistance during the semester are all acknowledged by Jemale and Jonathan.
The authors appreciate the crucial support and direction provided by their project's directors, Alessio Russo and Dominos Tranos.

VI. CONCLUSION

In summary, both Deep Q-learning and Double Deep Q-learning models performed better in terms of hit-rate when the game was run for the training phase using the hyper-parameters listed in Table III. This is shown in Figure 9. The ratio of the number of times the ball strikes the paddle to the total of the number of times the ball strikes the paddle and the number of times the ball misses the paddle, for the specified number of episodes, in this case 100, is the hit rate.

In order to replicate the outcome from [5], we construct a reinforcement learning algorithm in this paper. After some practice, our agent becomes just as good at Breakout as the average person, scoring an average of 20 as opposed to the human score of 31, and achieving a score of 168.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105013 |

We explore and assign a few likely causes—most notably uncertainties in the network setup of [5]—to this disparity. An iterative and experimental method was not feasible due to the significant computational burden of training the agent.

Finally, we talk about potential enhancements that could be made to the agent. We outline many potential methods from current research: Q-learning twice, experience prioritized

REFERENCES

1. https://subscription.packtpub.com/book/game-development/9781785281532/1/ch01lvl1sec09/an-overview-of-breakout
2. https://levelup.gitconnected.com/game-development-breakout-in-javascript-5e5d142d3203
3. https://spiderlabweb.com/breakout-game/

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details