Reinforcement Learning: Implementation and Evaluation of a Double Deep Q-Network Agent in the Waterworld Game

Popescu Tudor-George
Master of Artificial Intelligence and Optimization
Faculty of Computer Science Iași
Iași, România
tudor.popescu@info.uaic.ro

Romila Vlad-Alexandru
Master of Artificial Intelligence and Optimization
Faculty of Computer Science Iași
Iași, România
vlad.romila@info.uaic.ro

***Abstract*—This report presents the development and evaluation of a Double Deep Q-Network (DDQN) agent for the Water World game. By leveraging a DDQN approach as the primary reinforcement learning technique, the agent achieved a significant improvement in performance, demonstrating great stability and accuracy in the decision-making process. The overall success of this model was quantitatively evidenced by an average score of 10.21 during the training phase and scores of 9.7 and 12.5 in the testing phase with environments of 60 and 5 molecules, respectively. The improved game scores during both phases indicate the effectiveness of the DDQN model in complex, strategic gameplay scenarios.**

***Index Terms*—Deep Q-Network, Double Deep Q-Network, Reinforcement Learning, Agent, Replay Buffer**

I. Introduction

A. Problem Overview

Fig. 1. Waterworld Game

The Water World game, characterized by its simplistic yet strategic gameplay, involves a player-controlled molecule navigating through green (beneficial) and red (harmful) molecules. The player receives +1 points for every green molecules that it eats and -1 points for every red molecules that it eats. Every molecule eaten will have a chance of ½ to respawn a green molecules, and ½ to respawn a red molecule.The goal is to maximize the score by optimizing the consumption of green molecules and avoiding red ones.

B. Solution and Results

The final solution employed three main components:

1) A Main Neural Network, which uses inputs from player sensors that contain information about nearby molecules (type of molecule and its position) and outputs the desired action
2) A Target Network, which is used to provide the target Q-values during the training updates
3) A Replay Buffer which stores experiences in a deque data structure. This buffer allowed for efficient learning by enabling the agent to learn from a diverse and randomized set of past experiences.

The agent was trained over 2500 episodes, at a 1000 frame cap per episode, showing a consistent improvement in its decision-making capabilities. The training parameters chosen for the Waterworld game were: 60 molecules (30 green, 30 red), 30 frames per second. Key metrics included the average score per episode and the total score per episode. During the training phase, **an average score of 10.21** was obtained. During testing phase, **an average score of 9.7** was obtained in 10 episodes with 60 Molecules (the episode finished when number of frames reached 1000). A notable result was represented by **an average score of 12.5** which was obtained in 10 episodes with 5 molecules (the episode finished when the game finished).

The paper is organized as follows: Section II presents the state of the art, discussing foundational work and recent advancements in the application of Deep Q-Networks to game environments. Section III introduces our proposed approach, detailing the architecture of the DDQN model, the sensory input processing, and the replay buffer mechanism. Section IV describes the training methodology, including the initialization of the neural network models and the setup of the training environment. Section V, Benchmark Performance, reports on the empirical results obtained during both the training and testing phases, supported by graphical representations of the

agent's performance. Finally, Section VI concludes the paper with a summary of findings, discusses the implications of our research, and suggests directions for future work to further enhance the DDQN agent's capabilities.

II. State of art

The development of our Deep Q-Network (DQN) agent for the Water World game is deeply rooted in the foundational work of Mnih et al., as detailed in their landmark paper "Playing Atari with Deep Reinforcement Learning." This paper was pivotal in demonstrating the potential of DQN frameworks in mastering complex control tasks. Mnih and colleagues showcased how a neural network could learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. This approach resonates with our methodology, where the agent learns to navigate and make decisions in the Water World game based on sensory inputs from the environment. [1]

While specific benchmarks for the Water World game are not extensively documented, parallels can be drawn with DQN's applications in similar environments. For instance, in the realm of video game environments, DQN models have shown significant proficiency. A study by Mnih et al., titled "Human-level control through deep reinforcement learning" published in Nature, reports on the success of DQNs in handling complex environments with high-dimensional inputs, akin to the challenges posed by the Water World game. Their findings underscore the capability of DQNs to not only learn but also to outperform human-level expertise in certain scenarios. [2]

III. Proposed Approach

Our approach used for maximing the score in the Waterworld game includes constructing two neural network models, processing sensory inputs, defining the action space, utilizing a replay buffer, and other critical elements to facilitate effective learning and decision-making in a simulated environment:

A. DDQN Model Architecture

In developing our DDQN model architecture, we drew inspiration from the work outlined in 'Deep Reinforcement Learning with Double Q-learning' [3]. This research highlights the tendency of standard Q-learning to produce overoptimistic value estimates, particularly in environments with a large number of actions. Double Q-learning, as utilized in our DDQN model, addresses this issue by decomposing the maximum operation in the target into action selection and action evaluation. This separation significantly reduces the upward bias in value estimation, leading to more accurate assessments of the policy's quality and more effective decision-making processes.

Regarding our DDQN model, the network structure includes an input layer size determined by $(numSensors * maxMolecules * 3)$. This accounts for the sensors, each detecting a maximum number of molecules, with each molecule's data consisting of three values. Two hidden layers, each with 256 neurons and ReLU activation, process the input data. The output layer size equals the number of actions (4 actions: up, down, left, right), representing the agent's potential moves. The DDQN consists of two neural networks with identical structures. These networks function together: the main network is used to select the best action, while the target network, which is a delayed copy of the main network, is used to evaluate the action. This reduces the overestimation bias.

B. Sensory Input Layer

The sensory inputs comprise the relative positions and types of nearby molecules (green or red) in relation to the player molecule. At each frame, the game state is a sensor-based representation of the environment. An area with a radius of 200 pixels and having the player molecule at its center is analyzed and computed as an input layers. Each sensor (the player has 8 sensors in total) detects a maximum number of molecules. For each detected molecule, it records a value (1 for 'GOOD' molecules and -1 for 'BAD' molecules) and computes the normalized position of the molecule relative to the player. Sensor readings are normalized relative to a detection radius of 200 units and flattened into an input vector, ensuring consistent input size for the neural network.

C. Replay Buffer

Advancements in reinforcement learning, including efficient learning strategies like replay buffers, have contributed significantly to the field, especially in video game environments. [4] In this scenario, a replay buffer of size 10,000 is utilized to store and randomly sample experiences (state, action, reward, next state, done). It has a batch size of 32 for replay and is updated at each step (frame) during the training phase. The replay is done once every 50 steps (frames). This buffer is instrumental in facilitating experience replay, enabling the agent to learn from past actions and outcomes, thereby enhancing the learning process's efficiency and stability.

D. Training

The agent was trained over 2500 episodes, with the model's weights updated at a frequency of every 100 steps (frames). The training involves selecting actions, storing experiences, and learning from mini-batches of 32 experiences (using the replay buffer). Two models are initialized - the main model and the target model. The target model is employed in the replay method of the agent. This method is called periodically during training to update the main model based on experiences sampled from the replay buffer. If the next state is not terminal, the target model predicts the Q-values for the next state, and the maximum Q-value is used to calculate the target Q-value for the current state-action pair. This target Q-value is then used to update the main model. The target model is updated every 100 frames to match the main model's weights. The Waterworld parameters are 60 molecules (30 good, 30 bad), a maximum number of 1000 frames per episode and a game speed of 30 frames per second.

E. Testing

In the testing phase of our DDQN agent for the Waterworld game, we conducted a series of experiments to evaluate the effectiveness of the trained model. This involved testing the agent across 10 episodes to gauge its performance and decision-making capabilities in the game environment. Each episode consisted of 60 randomly generated molecules and lasted for a maximum of 1000 frames.

IV. Benchmark Performance

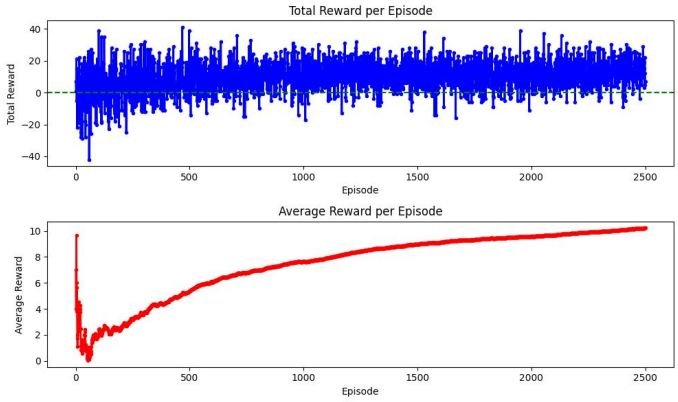

Fig. 2. Training phase plot

A. Training Phase Performance

The DDQN agent's training phase spanned over 2500 episodes, where its performance was rigorously documented and analyzed: **an average score of 10.21 per episode**.

The top chart in Figure X illustrates the total reward per episode throughout the training period. From the graph, it is evident that the agent experienced a high degree of variability in total reward, with some episodes reaching peak rewards while others dipped into negative values. Despite this, a discernible trend toward improvement can be observed, indicating a general learning progression.

The bottom chart details the average reward per episode, which provides a clearer picture of the agent's learning curve. Initially, the agent's performance was highly erratic, as indicated by the steep fluctuations in average reward. However, as episodes progressed, the average reward displayed a robust increasing trend, settling into a more consistent upward trajectory. This suggests that the agent was effectively assimilating the game's mechanics and strategies, leading to steadily improved outcomes as it navigated through the Waterworld environment.

B. Testing Phase Performance

During the testing phase, our trained DDQN agent was subjected to a series of episodes to evaluate its performance post-training. The agent's ability to generalize its learned strategies was assessed, with key metrics being the total and average scores across these test episodes. **An average score of 9.7** was obtained in 10 episodes with 60 Molecules (the episode finished when number of frames reached 1000). A notable result was represented by **an average score of 12.5** which was obtained in 10 episodes with 5 molecules (the episode finished when the game finished).

The results demonstrated that the agent was able to effectively apply its learned policies, as indicated by a consistent score close to the training phase's latter average reward. However, the testing phase also revealed areas where the agent's strategy could be further optimized, particularly in episodes with environmental setups not previously encountered during training.

V. Conclusions and Future Directions

In conclusion, the development and evaluation of a Double Deep Q-Network (DDQN) agent for the Water World game have demonstrated significant improvements in decision-making and stability over traditional DQN models. The incorporation of a replay buffer and the dual-network architecture of DDQNs played a pivotal role in enhancing the learning process and overall performance in complex environments.

Looking forward, several avenues can be explored to further refine and expand upon this work:

- **Algorithm Optimization**: Exploring advanced replay buffer techniques like prioritized experience replay to enhance learning efficiency.
- **Network Architecture Exploration**: Experimenting with various neural network architectures for improved generalization and adaptability.
- **Transfer Learning**: Implementing transfer learning to quickly adapt the agent to similar or more complex environments.
- **Multi-Agent Scenarios**: Extending the model to multi-agent learning for exploring cooperative and competitive dynamics.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou Daan Wierstra, Martin Riedmiller, "Playing Atari with Deep Reinforcement Learning", Department of Computer Science, University of Toronto, 2013

[2] Mnih, V., Kavukcuoglu, K., Silver, D. "Human-level control through deep reinforcement learning", Nature 518, https://doi.org/10.1038/nature14236, 2015

[3] Hado van Hasselt, Arthur Guez, David Silver, "Deep Reinforcement Learning with Double Q-learning", 2015

[4] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao, "A Survey of Deep Reinforcement Learning in Video Games", University of Chinese Academy of Sciences, 2019