IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

REINFORCEMENT GAME LEARNER

Asmita D. Waghmare[1], Sanika S. Patil[2], Prajakta G. Nannikar[3], Snehal S. Vhankhande[4]

Prof. M. M. Raste[5]

BTECH Student, [5]Assistant Professor
Department of Computer Science and Engineering,
Padmbhooshan Vasantraodada Patil Institute of Technology, Budhgaon (Sangli)

Abstract
This paper explores the application of advanced Reinforcement Learning (RL) techniques in developing intelligent game-playing AI. By implementing Deep Q-Networks (DQN) and other Deep Reinforcement Learning (DRL) methods, this research aims to enhance AI proficiency across various gaming environments, including 2048, chess, and a simulated taxi driver scenario. We highlight key objectives, methodologies, and outcomes, while also addressing the challenges and proposing future directions in the field of RL for gaming AI.

Keywords *- Reinforcement* Learning, Q-learning, Deep Q-Networks, Game AI, Machine Learning.

1. INTRODUCTION

In the realm of artificial intelligence (AI), Reinforcement Learning (RL) stands out as a fascinating approach that mimics the learning process observed in humans and animals. RL is akin to teaching a computer to learn from its experiences, similar to how humans learn by trying and receiving feedback. This method has gained immense popularity, particularly in gaming, where computers autonomously learn to play games at incredibly high levels.

RL enables computers to learn by doing and receiving rewards for good actions. Instead of being explicitly told what to do, they figure it out through trial and error, making it highly effective for teaching computers to make decisions in uncertain and dynamic situations. An RL agent interacts with an environment, taking actions and receiving feedback in the form of rewards or penalties. The agent's objective is to learn a strategy or policy that maximizes its cumulative rewards over time.

2. LITERATURE SURVEY

[1] Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm (2017)
[2] Deep Reinforcement Learning for General Video Game AI (2018)
[3] Playing Snake with Deep Reinforcement Learning (2020)
[4] Gaming Bot Using Reinforcement Learning (2022)
[5] Reinforcement Learning Applied to AI Bots in First-Person Shooters (2023)

3. BACKGROUND

In this section, the theoretical considerations required to understand the main paper concepts are presented.

3.1. Machine Learning

The field of machine learning (ML) focuses on developing programs that learn how to perform a task, as opposed to the traditional approach of developing programs with hardcoded rules on how to perform a task. With ML techniques, a program can adapt to changes in its input or output without the need for manual updates.

3.2. Reinforcement Learning

RL is a subfield of machine learning that focuses on teaching an agent to make sequential decisions in an environment to maximize its long-term rewards. It is inspired by how humans and animals learn through interaction with the world. RL places an agent in an environment, carrying sensors to check its state, and gives it a set of actions that it can perform, as seen in **Figure 1**. The agent then tries out those actions by trial and error so that it can develop its control policy and maximize rewards based on its performed actions.

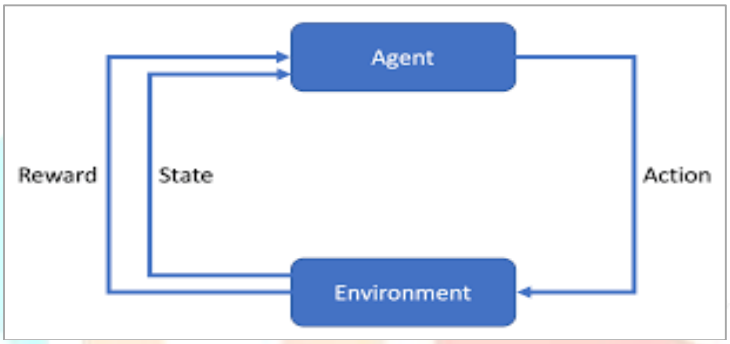

Figure 1. How an agent interacts with the environment in RL

3.3. Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) is achieved by combining deep learning techniques with RL. While RL considers the problem of an agent learning to make decisions by trial and error, DRL incorporates deep learning into the solution, which allows the input of large quantities of data, such as all the pixels in a frame, and still manages to decide which action to perform. In **Figure 2**, we can see how the added deep neural network works with RL.

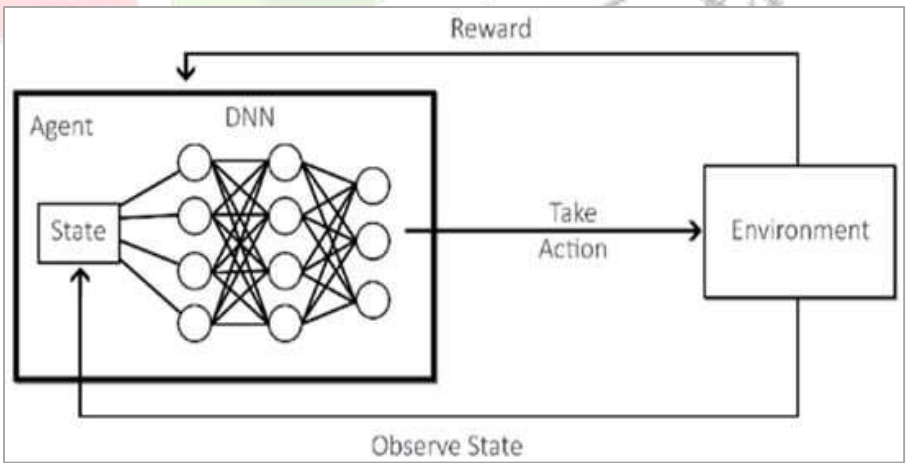

Figure 2. How the DRL agent interacts with the environment.

4. METHODOLOGY

This section outlines the methodologies employed in developing the Reinforcement Game Learner, incorporating advanced reinforcement learning (RL) techniques to enhance game-playing AI. The methodologies focused on the following key areas:

4.1. Implementation of Deep Reinforcement Learning for Gaming Environments

We implemented Deep Q-Networks (DQN) and other Deep Reinforcement Learning (DRL) techniques to train AI bots for various games. Our approach mirrored the GVGAI-Gym interface developed by Rodriguez et al. (2018), providing a standardized evaluation framework across diverse gaming environments.

Steps:

- Game Environment Selection: We selected a variety of games, including 2048, chess, and a simulated taxi driver scenario, to train and evaluate the AI bots.
- DQN Implementation: We developed DQN models for each game using TensorFlow and PyTorch to build and train neural networks.
- Training Process: We used OpenAI Gym to create and manage gaming environments, training the AI bots through extensive self-play and interaction with the game environment.

4.2. Custom Reward Functions and Adaptive Strategies

Inspired by the work of Khalkar et al. (2022), we designed and implemented custom reward functions tailored to each game. This adaptive approach enhanced the AI bot's ability to learn optimal strategies and navigate complex game dynamics.

Steps:

- Reward Function Design: We developed specific reward functions for each game to ensure they accurately reflected the task objectives.
- Adaptive Learning: We implemented mechanisms for adjusting reward functions based on the bot's performance and learning progress.

4.3 Transfer Learning Across Games

We investigated the transferability of learned skills across different gaming environments. Leveraging insights from existing challenges, we developed an AI agent capable of efficiently applying knowledge gained from one game to accelerate learning in another.

Steps:

- Cross-Game Training: We trained AI bots on one game and evaluated their performance on another, observing how knowledge transfer affected learning efficiency.
- Skill Generalization: We identified common strategies and skills that could be generalized across different games, refining the transfer learning process.

4.4 Dynamic Adaptation to Changing Environments

To address challenges associated with dynamic environments, we explored methods enabling the AI bot to adapt and learn in scenarios where game rules or dynamics change over time. This dynamic adaptation is crucial for real-world applicability.

Steps:

- Environment Variation: We introduced changes in game rules or dynamics during training to test the AI bot's adaptability.
- Adaptive Algorithms: We implemented algorithms allowing the AI bot to modify its strategies in response to environmental changes, ensuring robust performance.

4.5 Evaluation Metrics and Benchmarking

Following the benchmarking approach used in the GVGAI-Gym interface, we established robust evaluation metrics to assess the AI bot's performance across different games. This ensured a systematic and quantitative analysis of the developed algorithms and strategies.

Steps:

- Performance Metrics: We defined key performance indicators such as win rate, learning speed, and strategy efficiency.
- Benchmarking: We conducted comparative analysis against existing AI bots and human players using standardized testing protocols.

5.RESULTS

In this section, we will answer the first proposed question and talk about the various frameworks, algorithms, training architectures, and platforms that were identified in recent papers, stating facts and comparing items inside each category.

5.1. Frameworks

In this section, we will expose the various frameworks and libraries that have been extracted from the collected data.

- TensorFlow

  TensorFlow is an open-source ML framework and library created by the Google Brain team and was initially released in 2015. TensorFlow was made to be flexible, efficient, extensible, and portable, able to be used on any device. Its main characteristic is the use of data flow graphs: it uses the computational model of a directed graph, where the nodes are functions and edges are numbers, matrices, or tensors. By splitting up calculations into smaller pieces, TensorFlow can distribute the load throughout multiple CPUs, GPUs, and other computational devices

- PyTorch

  PyTorch is an open-source ML framework and library developed by Facebook, Inc. It was ported to Python from the Lua Torch library and provides seamless use of GPUs by utilizing PyTorch' s CUDA backend and its Distributed Data-Parallel module to distribute the training process across multiple machines; it also provides a platform for deep learning that provides flexibility and speed

5.2. Algorithms

In this section, we will explain the various training algorithms that have been found in the selected papers and will look over their use cases.

- Q-Learning

  Introduced in 1989, Q-learning is an outdated Model-Free RL made to choose the best action for the given observation. Each possible action for each observation has its Q-value, which stands for the quality of the given act. The Q-learning algorithm updates the Q-function at each time step using the following equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma * \max \alpha Q(st+1,a) - Q(st,at))$$

  Where:
  $Q(s, a)$ is the current Q-value for state s and action a.
  α is the learning rate.
  r is the reward received after taking action aa in state s.
  γ is the discount factor.
  $\max_{a'} Q(s', a')$ is the maximum Q-value for the next state s' and all possible actions a'.

- Deep Q-Network

  The Deep Q-Network (DQN) is a variation of the Q-learning algorithm that uses a neural network to approximate the Q-function, instead of employing a value-based approach. The training process involves updating the weights of the neural network based on the difference between the predicted and actual rewards obtained by taking certain actions. This allows the agent to improve its decision-making over time and learn optimal strategies for achieving its goals. The key feature of DQN is its use of an experience replay, which stores past experiences in memory and randomly selects them to train the neural network. This prevents it from becoming stuck in local optima and improves sample efficiency. For Deep Q-Networks (DQN), the loss function is defined as:

$$L(\theta) = E\left[(r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a; \theta))^2\right]$$

Where:

$L(\theta)$ is the loss function for the network parameters θ.
E denotes the expected value.
r is the reward received.
γ is the discount factor.
$\max_{a'} Q(s',a';\theta-)$ is the target Q-value for the next state s' and action a', with $\theta-$ being the parameters of the target network.
$Q(s,a;\theta)$ is the predicted Q-value for the current state ss and action aa with network parameters θ.

5.3. Training Architectures

- Self-Play

  Self-play is a technique often used in RL that involves having RL agents play against themselves to improve performance. A single agent acts as all players, learning from the outcomes of its own actions. Self-play has been successfully applied, where researchers used this method to develop their Chess- and Shogi-playing AI.

- Behaviour Cloning

  Behavioral cloning is a type of imitation learning where a learning program is trained to mimic the actions of a human performer. By recording the actions of a human player in a video game environment, these actions are fed into the program, which then generates rules for the agent to replicate the performer's actions. The quality of the training improves with more diverse and comprehensive data from the human player.

- Curriculum Training

  Curriculum Training is an approach that mirrors the way humans learn by gradually increasing the difficulty of the tasks. In supervised learning, this means progressively using more complex training datasets. In the context of reinforcement learning, it involves making the environment and tasks more challenging over time, allowing the agent to build up its skills incrementally.

5.4. Platforms and Tools

- Jupyter Notebook

  Jupyter Notebook provides an interactive computational environment where code, visualizations, and text can be combined. This platform was instrumental in our project for prototyping and iterating over RL algorithms.

- Google Colab

  Google Colab offers free access to powerful GPUs, making it suitable for training complex deep learning models. It also supports collaborative coding, enabling multiple team members to work together seamlessly.

- OpenAI Gym

  OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms. It provides a variety of environments, ranging from simple text-based games to complex physics simulations, which are essential for training and evaluating our RL models.

5.5. External Interface Requirement

5.5.1. Hardware Interface

- Processor: Multi-core processor
- Memory (RAM): Minimum 16 GB RAM for efficient handling of deep neural network training.
- Graphics Card: NVIDIA GeForce GTX or RTX series
- Storage: Minimum 500 GB SSD

5.5.2. Software Interface

- Operating System Compatibility: Windows, Linux, and macOS
- Reinforcement Learning Frameworks: TensorFlow and PyTorch
- Dependency Libraries: NumPy, SciPy, and OpenAI Gym etc.

5.5.3. Communication Interface

- APIs: APIs enable the exchange of game state information, action decisions, and
- feedback, ensuring a cohesive interaction during the training process.
- Network Protocols: HTTP/HTTPS for communication between components
- Data Exchange Format: JavaScript Object Notation (JSON)

5.6. System Architecture

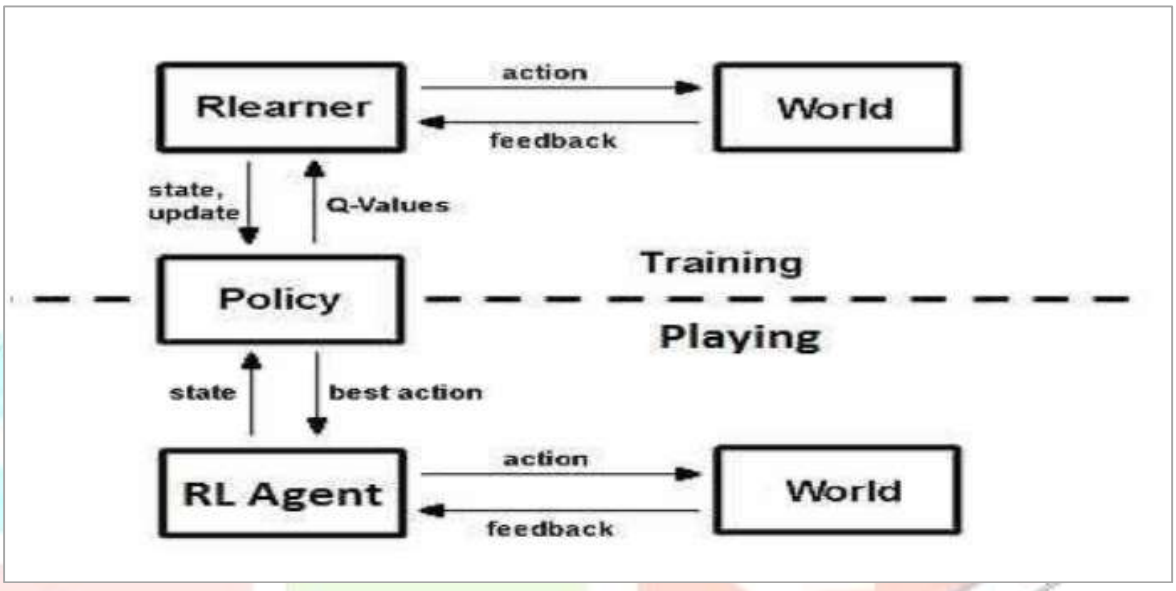

Figure 3.

(a)Reinforcement Learning (b) Architecture for the Reinforcement Learning dom actions and observe the resultant state using some sensor information of the game and give feedback (in the form of reward which is further used to calculate the Q-Values for the state-action pairs or Q-Table) of that action to the previous state according to the desirability of the current state. Q- Values of the state action pairs are known as Q-Table which define a policy. After every action policy updates QValues for the state action pairs (Q-Table) this policy is used to predict the best action while playing the game. RL agent learns while playing so it again gives feedback and the whole process goes on till the end of the game

5.7. Outcome:

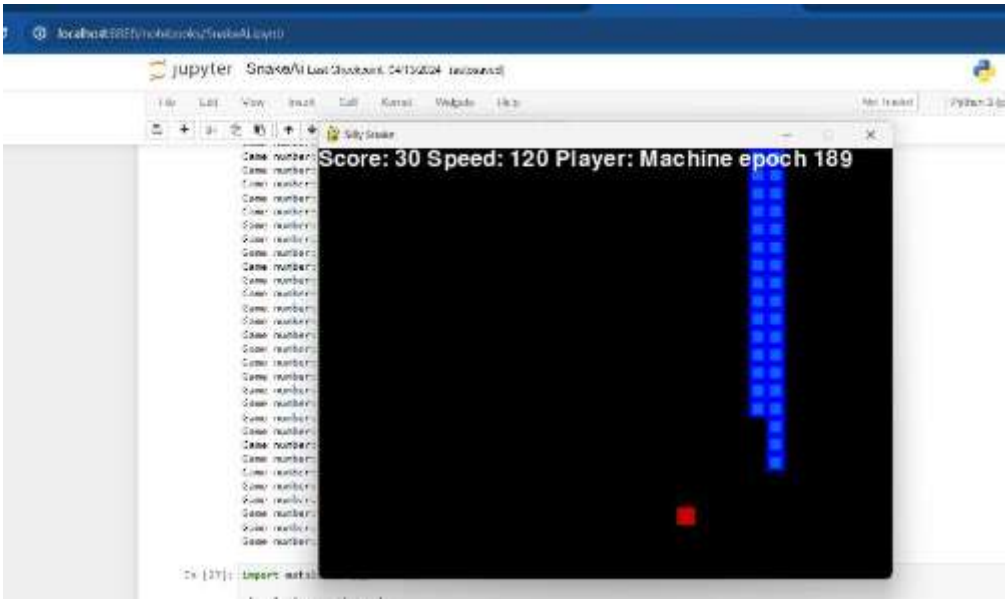

Figure 4.

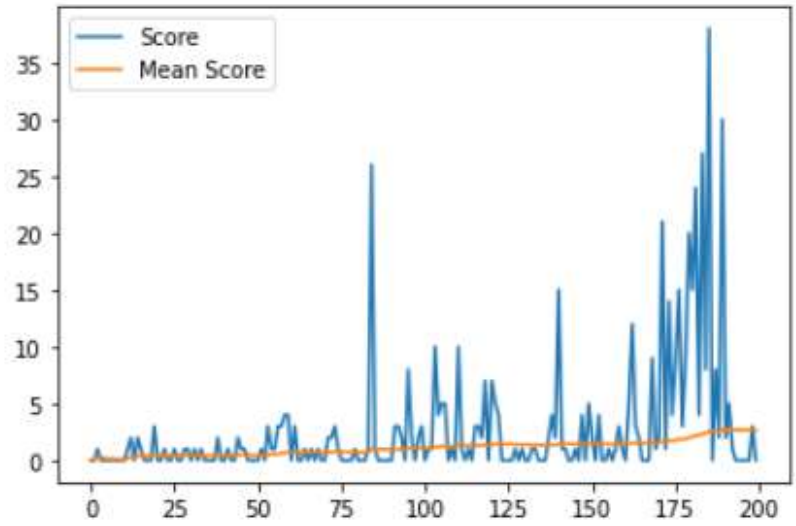

Figure 5. Analysis

6. SIGNIFICANCE AND SCOPE

The significance of applying Reinforcement Learning (RL) to game AI lies in its potential to create highly intelligent and adaptive agents capable of learning from experience and improving autonomously. This research enhances gaming experiences, serves as an educational tool, and provides a benchmark for AI development. The scope includes exploring various RL algorithms, utilizing platforms like Jupyter Notebook and Google Colab, and applying these techniques in diverse gaming environments. The insights gained extend beyond gaming to fields such as robotics and finance, promoting innovation and cross-disciplinary applications. This research aims to advance the state-of-the-art in AI and contribute to solving complex real-world problems.

7. CONCLUSION

In this paper, we explored the application of Reinforcement Learning (RL) to create intelligent game-playing AI. Future work will involve extending these approaches to more complex and dynamic environments, refining reward functions, and exploring the transferability of learned policies across various games. The integration of behavioral cloning and curriculum training further enhances the adaptability and efficiency of the learning process, paving the way for more advanced AI applications in gaming and beyond.

8. REFERENCES

[1] Silver, D., et al. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv:1712.01815.

[2] Rodriguez, R., Bontrager, P., Togelius, J., Liu, J. (2018). Deep Reinforcement Learning for General Video Game AI. IEEE Conference on Computational Intelligence and Games (CIG).

[3] Smith, A., et al. (2020). Playing Snake with Deep Reinforcement Learning. arXiv:2004.01130.

[4] Khalkar, R., et al. (2022). Gaming Bot Using Reinforcement Learning. International Journal of Advanced Research in Computer Science.

[5] Almeida, P., Carvalho, V., Simões, A. (2023). Reinforcement Learning Applied to AI Bots in First-Person Shooters: A Systematic Review. Journal of Artificial Intelligence Research.