Understanding The Self-Improving Machine Agent Learning Process in Gaming AI

Meghraj S. Sonawane[1], Meghna D. Surwase[2], Poonam P. Patil[3]
[1,2,3]*Department of Computer Application, K. K. Wagh Institute of Engineering & Education Research Nashik*

***Abstract*—The self-improvement processes of machine learning agents and algorithms used in gaming AI are study in this research study. We study the learning procedures, underlying algorithms, and how reinforcement learning methods improve agent performance. We offer insights into the possibilities of self-improving agents in the gaming industry and beyond by looking at a number of case studies, such as DeepMind and other gaming AI systems.**

***Index Terms*—Machine Learning, AI, Self-Improvement, Reinforcement Learning, Gaming AI.**

I. INTRODUCTION

The artificial intelligence (AI) has revolutionized a number of fields, and one important testing ground for cutting-edge machine learning methods is gaming. Gaming AI has demonstrated impressive ability in self-improvement and adaptive learning, especially through its DeepMind subsidiary. With an emphasis on reinforcement learning, algorithmic developments, and the ramifications of these technologies, this study seeks to clarify the mechanisms that allow these agents to improve their performance on their own Background

A. Machine Learning and AI in Gaming

Algorithms that learn from data to generate predictions or judgments without explicit programming are known as machine learning, a subset of artificial intelligence. These methods give agents in games the ability to adjust to player tactics, enhance gameplay, and produce captivating experiences.

B. Self-Improvement Mechanisms

In artificial intelligence, self-improvement is the capacity of an agent to gradually improve its performance through experience. This procedure frequently makes use of reinforcement learning (RL), in which agents discover the best course of action by being rewarded or punished according to their behavior.

II. LITERATURE REVIEW

This study investigates the self-improvement of machine learning agents in gaming AI [5],[6]. Its main goal is to comprehend the procedures and algorithms used in reinforcement learning [7]. In order to demonstrate the potential of self-improving agents in gaming and other fields, [7][8]. Without specialized programming, machine learning uses data to generate predictions. AI agents for gaming can improve gameplay, produce captivating experiences, and adjust to player strategies [8]. In artificial intelligence, self-improvement refers to agents progressively enhancing their performance via experience, frequently through reinforcement learning. In gaming AI, algorithms such as Proximal Policy Optimization [4], Deep Q-Learning, and Q-learning [11],[12] are frequently employed. These algorithms assist agents in learning to make decisions based on rewards and penalties and optimizing their actions. The study looks at examples such as AlphaStar and AlphaGo to demonstrate how self-improving [1].

III. METHOD

1. Problem Overview (understand problem):

The method's goal is to create a reinforcement learning algorithm that enables an AI agent for gaming to learn how to play games on its own by interacting with its surroundings. Examples of these games include strategy games, board games, and video games.

2. Design and Develop the Algorithm
Algorithm Selection Q-Learning A classic RL algorithm where the agent learns a Q-value function s. The goal is to learn the optimal policy by maximizing the cumulative reward over time. Deep Q-Learning DQN For games with large state spaces like video games, a deep neural network is used to approximate the Q-values. The state is input into the network, which outputs Q-values for all possible actions. Policy Gradient Methods These methods directly optimize the policy as, which is a distribution over actions given a state.

3. Process of Implementation
Environment Setup:
Utilize OpenAI Gym or Solidarity ML-Agents for making standardized situations where operators can be prepared and tried.

4. Agent Training
Agent training perform using Machine learning algorithms.
Training Procedure The agent learns through interactions with the game environment. The key steps in the training procedure are Initialization Initialize the policy or Q-table and other parameters such as the neural network weights, exploration rate Game Loop The agent repeatedly interacts with the environment the game by Observing the current state. Selecting an action based on the policy or Q-values, using an exploration strategy. Performing the action and receiving a reward Observing the new state in a memory buffer for experience replay.

5. Testing
To evaluate the performance of an agent in reinforcement learning, there are three main testing methods: Single Episode Testing, Multi-Episode Testing, and Compare Against Baseline testing. Single Episode Testing involves observing the agent's behavior in a few episodes to determine if it is following an optimal strategy. Multi-Episode Testing looks at the agent's average performance over multiple episodes to smooth out randomness. Comparing against a baseline, such as a random or rule-based agent, helps understand the effectiveness of the reinforcement learning approach.

6. Result Evaluation
Evaluation and Performance Metrics Cumulative Reward Measure the total reward accumulated by the agent over an episode or series of episodes. Convergence Speed Track how quickly the agent improves its performance and reaches near-optimal behavior.

IV. METHODOLOGY

Planning tests with different Reinforcement Learning methods and self-improvement instruments. Collecting and analyzing real-time information on specialist execution. Assessing the learning prepare and how viably operators progress independently. Utilizing factual strategies to analyze execution measurements and decide the victory of self-improvement procedures.

Reinforcement Learning Techniques
Fundamentals of Reinforcement Learning

1. The foundation of reinforcement learning is the interaction between an agent and its surroundings. In order to optimize cumulative rewards, the agent takes activities, gets input in the form of rewards, and modifies its tactics.
2. Markov Decision Process (MDP): The mathematical structure that outlines the agent's surroundings, states, choices, and incentives.
3. Policy Learning: The process agents use to devise plans for choosing actions according to the present situation.

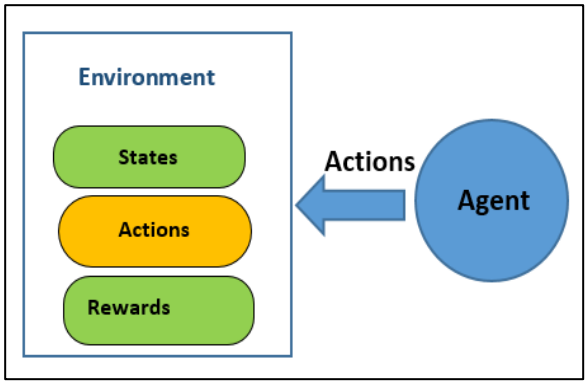

Fig.1 *Agent Perform Action on Module*

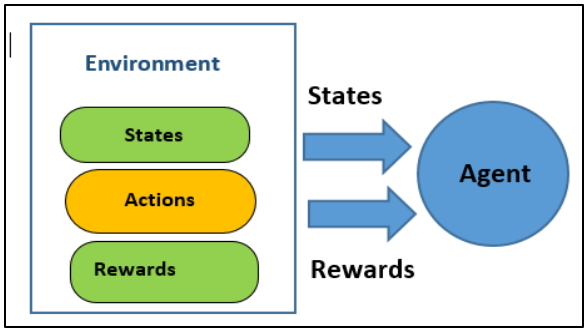

Fig.2 *Environment send the reword to agent*

Algorithms in Use Gaming AI employs several RL algorithms, but in this study, we majorly focus on Q-learning, Deep Q-Learning and Proximal Policy Optimization:

1. Q-Learning:

Q-Learning is notable among the Lookup-Table-based(fig.3) approaches we discussed earlier, as it lays the groundwork for Deep Q Learning. The process of Q-learning employs a Q-table that includes State-Action Values, which are commonly known as Q-values. In this Q-table, each row corresponds to a state, while each column denotes an action. The estimated Q-value for every state-action pair is recorded in the respective cell. We start by initializing all Q-values to zero. As the agent interacts with the environment and receives feedback, the algorithm progressively adjusts these Q-values until they converge to the Optimal Q-values. This adjustment is accomplished through the use of the Bellman equation.

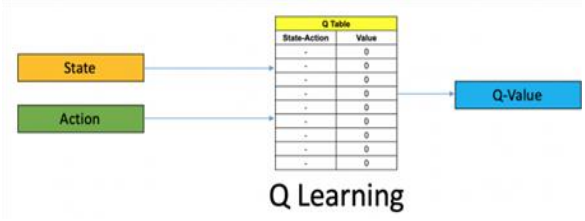

Fig.3 *Q Learning*

Q-value Update Rule: The Q-values are updated using the formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

Where:

- s is the current state.
- a is the action taken.
- r is the reward received after taking action a in state s.
- s' is the new state after action.
- a' is any possible action from the new state s'.
- α is the learning rate ($0 < \alpha \leq 1$).
- γ is the discount factor ($0 \leq \gamma < 1$)

2. Deep Q-Networks (DQN):

The deep Q-network (DQN) algorithm represents an off-policy reinforcement learning approach tailored for environments characterized by discrete action spaces. As shown in fig 4, A DQN agent develops a Q-value function to predict the anticipated discounted cumulative long-term rewards associated with adhering to the optimal policy.

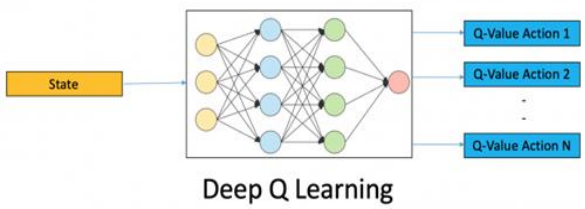

Fig.4 *Deep Q Learning*

This method is a specialized form of Q-learning that incorporates a target critic and an experience replay buffer. Additionally, the DQN agent is capable of offline training, allowing it to learn from previously collected data without the need for an active environment. For further details on Q-learning.

1. Set up the networks:

Configure both the online network Q online and the target network Q target with identical initial weights.

2. Engage with the environment:

The agent engages with the environment by choosing actions based on the policy derived from Q online, usually employing an epsilon-greedy approach (which balances exploration and exploitation).

3. Record experiences:

Every interaction (state, action, reward, subsequent state) is recorded in the experience replay buffer.

4. Select a mini-batch:

Randomly select a mini-batch of experiences from the replay buffer

5. Calculate the target:

For each selected sample, determine the target utilizing the equation:

$Y = r + \gamma\, a' \max Q_{target}(st+1, a')$
Where:
r is the reward obtained from the action a_t γ is the discount factor max a' $Q_{target}(st+1, a')$ is the maximum Q-value for the next state s_t+1 according to the target network.

6. Update the online network
Perform a gradient descent step to minimize the loss between the predicted Q-values and the target:
$Loss= (Q_{online}(s_t,a_t)-y)^2$

7. Update the target network
Periodically update the target network by copying the weights from the online network.

Target Update Methods Smoothing -
Adjust the target parameters at each time step by applying the smoothing factor τ. To define the smoothing factor, use the Target Smooth Factor option. $\phi_t=\tau\phi+(1-\tau)\ \phi_t$ Periodic- Periodically refresh the target parameters without applying smoothing (set Target Smooth Factor to 1. To define the frequency of updates, utilize the Target Update Frequency Parameter Periodic Smoothing — Revise the target parameters at regular intervals using a smoothing technique.

TABLE I: Target parameters

Update Method	Target Update Frequency	Target Smooth Factor
Smoothing (default)	1	Less than 1
Periodic	Greater than 1	1
Periodic smoothing	Greater than 1	Less than 1

(Courtesy: MathWorks visited on date:17/11/2024)

8. *Proximal Policy Optimization (PPO)*:
Proximal Policy Optimization (PPO) is an algorithm used in reinforcement learning (RL) that enables an agent to learn how to make decisions for task completion. As a policy gradient method, PPO operates by alternating between gathering data and refining a surrogate objective function. It falls under the category of actor-critic algorithms, comprising two main elements: the actor, which determines the actions to be taken, and the critic, which assesses the effectiveness of those actions.

Training Process for PPO
The training process in PPO typically follows these steps:
1. Collect Data: As fig 1&2 depicts, the agent interacts with the environment to collect trajectories (state-action-reward sequences).
2. Compute Advantages: For each trajectory, compute the advantage estimates at using the rewards and value function.
3. Compute the Objective: Using the new data, compute the PPO objective function (with the clipped term) and update the policy.
4. Policy Update: Update the policy network using the objective. This is done over multiple epochs for each batch of data.
5. Repeat: Repeat the process for several iterations or until convergence.

Case Studies
A. AlphaGo
AlphaGo [1], developed by DeepMind, utilized a combination of supervised learning and reinforcement learning to master the game of Go. By training on human games and competing against itself, it significantly improved its strategies, leading to ground breaking successes against top human players.

B. StarCraft II AI
DeepMind's AlphaStar demonstrated advanced self-improvement techniques in playing StarCraft II. It employed a multi-agent system that learned various strategies through competitive play, adapting to different opponents' styles over time. In this Study we find AlphaStar's gameplay to be highly commendable – the system excels at evaluating its strategic stance and determines the optimal moments to confront or retreat from its opponent. Additionally, while AlphaStar demonstrates exceptional and accurate control, it doesn't come across as superhuman – certainly not beyond what a human could potentially accomplish. In general, it appears quite balanced – as if it is engaged in an authentic game of StarCraft.
AlphaStar is a fascinating and unconventional competitor, exhibiting the quick reflexes and speed of top professionals while employing unique strategies and style. The training process for AlphaStar, which involved agents competing in a league, has produced gameplay that is astonishingly distinctive; it prompts a re-evaluation of how thoroughly professional players

have investigated the many possibilities within StarCraft.It was thrilling to observe how the agent formulated its own strategies that differed from those of the human players [6]. The limitations on its actions and the constraints of the camera view now create engaging gameplay – even though, as a professional, I can still identify some of the system's flaws.

Challenges and Limitations

Despite the advancements, several challenges persist:

1. *Computational Resources:* Training self-improving agents requires substantial computational power and time.
2. *Generalization:* Ensuring that learned strategies apply across different contexts remains a significant hurdle.
3. Ethical Considerations: The implications of highly adaptive AI agents raise questions about control, predictability, and fairness in gameplay.

V. CONCLUSION

Gaming AI's self-improving features indicate a major advance in machine learning abilities. By utilizing reinforcement learning and advanced algorithms, these agents are reshaping gaming experiences while also providing game-changing insights for multiple industries. In order to fully unlock the capabilities of self-improving AI, it will be essential to tackle the obstacles related to resource needs and ethical issues as technology progresses.

VI. FUTURE SCOPE

Enhanced NPC Behavior and Narrative Development Current Trends: Nowadays, non-playable characters (NPCs) typically operate under pre-defined actions or basic decision trees. However, by leveraging AI, NPCs can learn, adjust, and react to the player's actions in real time, resulting in more immersive and captivating environments. *Future Vision:* In the future, NPCs could possess intricate, evolving personalities and interactive experiences. Through the use of reinforcement learning (RL) and natural language processing (NLP), NPCs could cultivate distinctive behaviors that evolve over time based on the player's choices, actions, and even emotional undertones. 2.Personalized and Adaptive Game Design AI frameworks will survey the player's abilities and alter the game's trouble on the fly, guaranteeing that players are not one or the other disappointed nor bored. For case, in the event that an AI identifies a player is battling, it might alter the trouble or offer more in-game instructional exercises. On the other hand, it might increment the challenge for more gifted players to preserve engagement.

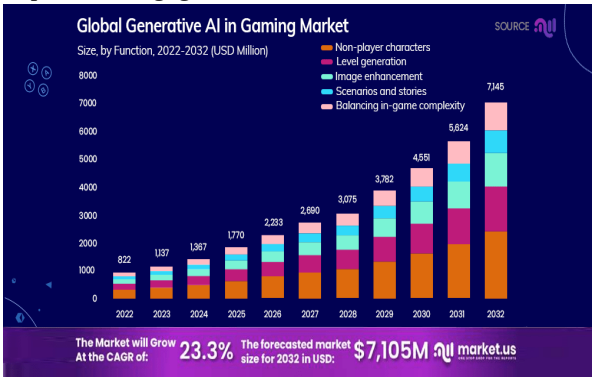

Fig.5 Global AI Market Trend

As Fig 5 depicts the market growth of AI in gaming from 2022 to 2032(expected) also in this research study we can see current and future expected enagement of AI in gaming industry.

REFERENCE

[1] O. Vinyals *et al.*, "AlphaStar: Mastering the real-time strategy game StarCraft II," DeepMind, 2019.

[2] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[5] M. Frutos-Pascual and B. García-Zapirain, "Review of the use of AI techniques in serious games: Decision making and machine learning," *IEEE Access*, 2017.

[6] K. U. Akkshay and B. Sreevidya, "Development and performance analysis of an AI-based agent to play computer games using reinforcement learning techniques," 2024.

[7] S. L. Epstein, "Game playing: The next moves," in *Proc. AAAI/IAAI*, 1999, pp. 987–993.

[8] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.

[9] Y. Wang and S. Gelly, "Modifications of UCT and sequence-like simulations for Monte-Carlo Go," in *Proc. IEEE Symposium on Computational Intelligence and Games*, 2007, pp. 175–182.

[10] L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 211–229, 1959.

[11] J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.

[12] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," University of Cambridge, Tech. Rep., 1994.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *IEEE Signal Processing Magazine*, 2017.