A Comparison of Learning Strategies in Freeze Tag: Behavior Cloning vs Curriculum Learning with RL Agents

Nilam Honmane[1], Yash Nikum [2], Pranav Potdar [3], Vedant Pawashe[4], Harsh Sarda[5]
[1,] *Prof. at Zeal College of Engineering and Research, Pune, India*
[2,3,4,5] *Student at Zeal College of Engineering and Research, Pune, India.*

***Abstract*—This study compares Curriculum Learning (CL) and Behavior Cloning (BC) in training smart agents for a multi-agent Freeze Tag game environment constructed with Unity ML-Agents. The game consists of two types of agents—taggers and runners—and involves cooperative as well as competitive interactions. We utilize Proximal Policy Optimization (PPO) as our base reinforcement learning algorithm, supplemented with CL for progressive skill learning and BC for imitation learning from expert demonstration. Training utilized parallel environments to maximize data throughput, and systematic evaluation by agent-vs-agent games. Results show that the BC-trained taggers had marginally better win rates, but the CL-trained taggers had better strategic behavior, resource use, and learnability. These results emphasize the trade-offs between imitation learning and curriculum-guided progression in large-scale multi-agent systems.**

Keywords: **Multi-Agent Learning, Curriculum Learning, Behavior Cloning, Reinforcement Learning, PPO, Unity ML-Agents**

I. INTRODUCTION

Reinforcement Learning (RL) has progressed essentially in tackling complex decision-making issues, supported by calculations like Proximal Approach Optimization (PPO) and stages such as Solidarity ML-Agents. These apparatuses empower operators to memorize versatile practices through interaction with energetic situations, with applications crossing mechanical technology, independent route, and 3D recreations. Multi-agent recreations offer wealthy testbeds for assessing AI learning systems, particularly where agreeable and antagonistic practices rise. The Solidify Tag diversion represents such complexity, including deviated roles—runners point to sidestep and unfreeze partners, whereas taggers point to solidify all rivals utilizing constrained assets. This makes Solidify Tag a perfect environment for examining key learning beneath different conditions. This work explores and compares two conspicuous learning approaches for preparing taggers: Educational programs Learning (CL), which continuously increments assignment trouble, and Behaviour Cloning (BC), which depends on impersonation of master exhibits. Whereas both have appeared guarantee, their comparative adequacy in multi-agent, team-based situations stay underexplored. To disconnect the effect of tagger preparing, runners are reliably prepared utilizing CL over all tests. We utilize Solidarity ML-Agents to actualize a measured and reproducible preparing environment, encouraging future inquire about in encapsulated AI and intelligently reenactment settings environment, facilitating future research in embodied AI and interactive simulation settings.

II. BACKGROUND

Here, we will deal with some topics that the reader should take into account to be able to understand the work, that is, from the Freeze-tag genre to some Machine Learning (ML) topics and to the Unity engine and to the ML-agents Toolkit.

2.1 Freeze-Tag

Freeze Tag is an old playground game that combines aspects of chasing, tagging, and cooperation. Conventionally, one or more players become "taggers," whose role is to freeze the other players by tagging them. After tagging a player, they have to remain frozen in position—usually with their arms outstretched—until another live player comes to "unfreeze" them through physical contact. This mechanism promotes thinking ahead and teamwork among teammates. The game lasts until all the players are frozen or a given time expires.

2.2 Machine Learning (ML)

Machine Learning (ML) is the discipline

concerned with developing systems that have the ability to learn to conduct tasks without being explicitly programmed to do so. Contrary to conventional software that obeys pre-defined rules, ML-based systems are capable of learning and improving as they go through new information or variations in their surrounding environment.

2.3 Deep Learning

Deep Learning is a subset of machine learning that employs artificial neural networks with lots of layers to process complicated data. Deep networks are especially well-suited to interpreting raw input data, which makes them well-suited for such tasks as recognizing images, voice assistants (such as Siri or Alexa), recommendation systems for content, and even the creation of images. What distinguishes deep learning is its requirement for tremendous quantities of data—these models are optimized when they're trained on huge datasets, which is why they tend to be utilized in state-of-the-art AI implementations.

2.4 Reinforcement Learning (RL)

Reinforcement Learning is also a branch of machine learning, but it's concerned with instructing an agent how to go about making a sequence of decisions so that it reaches a goal. It's based on the manner in which animals and humans acquire knowledge from their interactions with the world. In RL, an agent is dropped in an environment where it can sense its state through sensors and perform actions from a predefined set. Through trial and error over time, the agent learns what actions result in more desirable outcomes. In so doing, it creates a policy—or strategy—that enables it to make improved choices and maximize its long-term benefits, as shown in Figure 1.

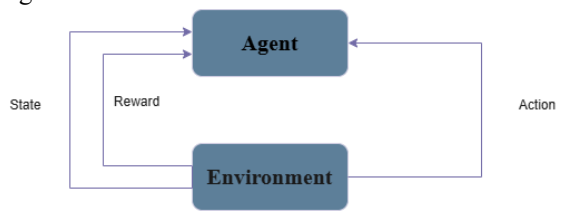


Figure 1: Agent interaction with the environment in RL

A Reinforcement Learning (RL) system is made up of an agent that exchanges information with an environment by making actions derived from a policy a strategy for mapping states to actions. The environment reacts with a new state and a reward signal, which the agent utilizes to modify its policy to try and get the maximum long-term rewards

2.5 Behaviour Cloning

A Behaviour cloning is an imitation learning approach that involves learning to conduct tasks through the imitation of expert behavior. Rather than learning via trial and error as in curriculum learning, behaviour cloning involves supervised learning to train the agent on a set of demonstrations from a human expert. The objective is for the agent to mirror the actions of the expert in similar circumstances. This method is particularly helpful in settings where it is hard to define a reward function or where the exploration may result in failure or unwanted results.

2.6 Curriculum Learning

In Curriculum Learning gets its inspiration from how humans learn—beginning with easy tasks and progressing upwards to harder tasks. In machine learning, this corresponds to incrementally increasing the ease of training as the model or agent becomes better. In Supervised Learning, this could be starting with easy examples and adding increasingly hard ones with time. In Reinforcement Learning (RL), typically, it means increasing the complexity of the environment or task as the agent gains experience.

2.7 Proximal Policy Optimization Algorithm

Proximal Policy Optimization (PPO) is an OpenAI-developed model-free reinforcement learning algorithm. PPO trains a policy function that takes the state of the environment and outputs an action. PPO alternates between collecting data by interacting with the environment and optimizing a surrogate objective function based on stochastic gradient ascent.

2.8 GAIL

Generative Adversarial Imitation Learning (GAIL) is a model-free imitation learning algorithm that can train agents to learn policies directly from expert demonstrations in the absence of explicit reward signals. It makes use of concepts from Generative Adversarial Networks (GANs) by training a policy (generator) to imitate expert behavior and a discriminator to differentiate between expert and agent trajectories. The adversarial setup makes it possible for GAIL to generalize well to complex, high-dimensional worlds.

III. TRAINING ARCHITECTURE

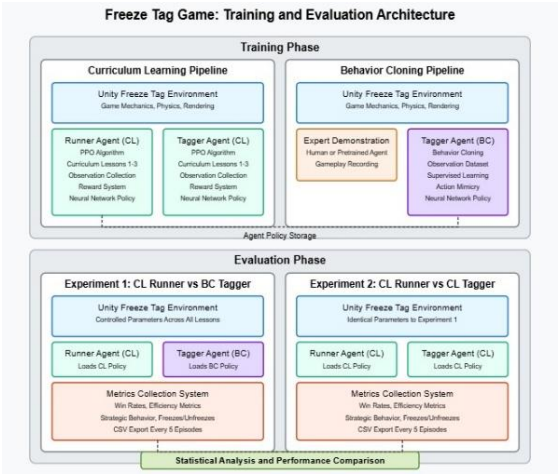


Figure 2: Architecture

3.1 Behaviour Cloning:

In this project, we used behaviour cloning for training game agents in Unity with the ML-Agents toolkit. Freeze Tag is a game with two agent types: taggers and runners. We gathered demonstration data by controlling both tagger and runner agents manually on the master level with hardest difficulty, where taggers learned how to gather and shoot freeze balls to freeze runners and learn how to gather wall balls and use them to defend. This demonstration data was utilized to train the agents through behaviour cloning so that they could learn from expert strategies and overcome the stage/level wise training employed by curriculum learning. In doing this, we sought to develop agents that could play their roles in the game environment effectively with maximum difficulty, learning survival and attack strategies directly from human demonstrations.

3.2 Curriculum Learning

To be able to train tagger agents in an organized way, we used a curriculum approach with three very different lessons. The lessons are designed as progressively more complex environments and game requirements. The curriculum is developed to create the agents' tagging habits, resource management, stress decision-making, and finally, how they will work together in a dynamic multi-agent environment. The learning curve was controlled by a uniform runner survival level of 2.0 for all training sessions so that improvements in performance translated into actual skill development and strategic adjustment.

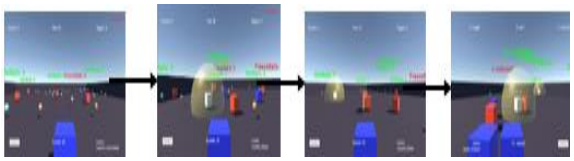


Figure 3: Levels with increasing difficulty are shown from left to right.

3.3 ML-Agents

In our Freeze Tag project, we employed Unity's ML-Agents Toolkit to train and test intelligent agents within a multi-agent setting. ML-Agents facilitated smooth integration of learning algorithms such as Behavioral Cloning and PPO by binding Unity with Python-based training scripts. We specified agent observations, actions, and rewards in Unity, enabling realistic and interactive training situations. The toolkit further accommodated curriculum learning and effective metric monitoring, thereby making it a must for creating adaptive and tactical agent behavior in the context of Freeze Tag.

IV. IMPLEMENTATION

4.1 Unity

For our deployment, we utilized the Unity game engine and ML-Agents Toolkit to implement and simulate the Freeze Tag environment. Unity gave us a solid 3D foundation upon which we could create custom prefabs for taggers, runners, obstacles, and FreezeBalls, allowing us to have rich agent interactions. We heavily used Unity Events to monitor and log important gameplay statistics like freezes, unfreezes, and FreezeBall usage in real-time throughout training and evaluation. The ML-Agents Toolkit supported the interface between Unity and Python-based reinforcement learning algorithms to enable agents to learn through experience in the environment. We also produced standalone build files of the environment for reproducible training performance and platform-independent experimentation.

4.2 Use of Raycast with Game Environment

Application of Raycast with Game World Raycasting within Unity is important for facilitating agents to perceive their environments realistically and make rational choices. Through virtual rays projected to scan obstacles, pickups, and other agents, raycasting mimics line-of-sight and spatial perception. It provides this information, incorporated into agent perceptions, with added support for navigation, collision prevention, and strategic interaction, boosting both training efficiency as well as game realism.

4.3 Hardware Used for Training

The training and testing process was conducted on a personal laptop with the follow- ing specs:

- CPU: Ryzen 7 5800H
- RAM: 16 GB;
- GPU: NVIDIA RTX 3050TI @ 4 GB VRAM

4.4 Game Environment Setup:

The game consists of two competing agent types:

- Runners: Agents that attempt to escape freezing by taggers and can also unfreeze other teammates.
- Taggers: Agents that strive to freeze all runners before the time is up.

4.4.1 Gameplay Mechanics

- Core Loop: Taggers attempt to freeze every runner before the time's up, and runners attempt to stay alive.

4.4.2 Victory Conditions:

- Runners win if one or more runners are still unfrozen when time runs out.
- Taggers win if they manage to freeze all runners before the time runs out.

4.4.3 Agent Abilities:

Runners:

- Move freely in the environment.
- Create walls to block taggers and freeze balls.
- Unfreeze teammates (takes 3 seconds of standing nearby).
- Collect wall balls to create barriers.

Taggers:

- Move freely in the environment.
- Directly freeze runners on contact (if they have freeze balls).
- Shoot freeze balls at runners.
- Collect freeze balls for ammunition.

4.4.4 Items:

- Wall Balls: Allow runners to create temporary barriers.
- Freeze Balls: Used by taggers to freeze runners, either through direct contact or by shooting them.

Figure 4: Environment in which agents were trained on the spawn points scatted around the arena

V. TRAINING AND TESTING

To validate which training architecture performed most optimally, we trained a few agents to play a basic Freeze-tag game, in which the agents confronted each other in square shaped arena. We trained two kinds of agents: with Curriculum learning and also Behaviour Cloning; Each agent was trained for one million steps, a figure that we established as viable.

Table 1: Details of each training type.

Training Type	No. of Lessons	Description
Curriculum Learning (Runner)	3	Used a 3-stage Curriculum Learning to teach the agent how to move, collect wall balls, dodge the freeze balls, and spawn walls, unfreeze the fellow frozen runners
Curriculum Learning (Tagger)	3	Used a 3-stage Curriculum Learning to teach the agent how to move, shoot the freeze balls, and collect spawned freeze balls and learned to freeze runners
Behaviour cloning (Tagger)	1	Used previously recorded data into the agent and made them play among themselves using GAIL.

5.1 Agents

Agents sense their environments from variable inputs, such as simulated audio signals and a rotational line-of-sight system for opponent detection. Navigation and collision avoidance are done by Unity ML-Agents' Raycast sensors, which give information about environmental barriers close to them.

5.2 Agents Sight

In our project, every agent has a 360-degree field of view raycast system that casts rays up to 50 tiles of distance so that it can sense the nearby objects, obstacles, and other agents for navigation and decision-making.

5.3 Inputs

The below table shows the Raycast sensors for both the agents (Taggers and Runners) and the values for both sensors are same.

Table 2: Agent's sensor's settings

Description	Value
Rays per direction	75
Max ray degrees	180
Sphere cast radius	1
Ray length (unity units)	50
Stacked raycast	1
Start vertical offset	0
End vertical offset	0

Table 3: Taggers Observations

Observation	Vector Size
Position	3
Forward Direction	3
Freeze Ball Percentage	1
Shoot Cooldown State	1
Runners Frozen Percentage	1

Table 4: Runners Observations

Observation	Vector Size
Position	3
Forward direction	3
Frozen state	1
Wall ball percentage	1
Wall pull down state	1

5.4 Agent Collider

All the agents were fitted with a collider that only reacted to collisions with walls and obstacles. In case a collision occurred, the agent was rewarded negatively so that it would be discouraged from remaining in or colliding with these objects. This reduced the number of times the agents would get trapped and also enhanced their pathfinding, though sometimes they still got trapped close to walls.

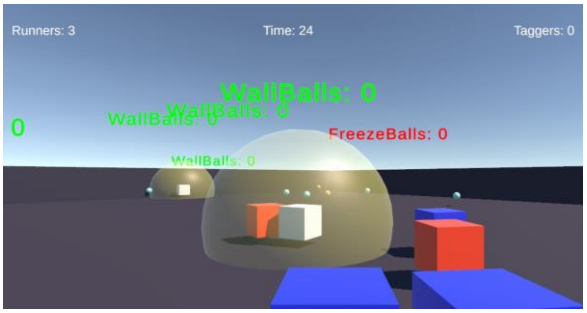


Figure 5: The collider (around the agent) that tells the agent (in white) if they are near an obstacle (in orange).

5.5 Parallel Training

In order to enhance training efficiency and accelerate policy convergence, we employed Unity's multi-environment feature with ML-Agents by configuring the num_envs parameter in the training YAML file. This allowed several instances of the environment to run concurrently so that agents can gather varied experience data in parallel. In both Curriculum Learning and Behavioral Cloning, this helped speed up learning by exposing agents to more varied scenarios. In spite of hardware constraints on parallel instances, this approach improved significantly the training throughput and agent performance.

VI. TRAINING

The succeeding sub-sections outline the various forms of training that were undertaken in the course of this research.

6.1 Curriculum Learning Training

6.1.1 Lesson one: In this lesson, basic skills such as shooting and aiming freeze balls in a reduced setting with 4 taggers and 4 runners, plenty of resources, and a timer of 90 seconds were taught. Agents learned core mechanics under low-distraction situations and were tested with accuracy and basic reactivity.

6.1.2 Lesson Two: With 5 taggers and 5 runners, fewer assets, and an 80-second cap, this phase focused on decision-making within limitations. Shorter cooldowns and restricted freeze balls favored coordination, resulting in more strategic play and cooperation.

6.1.3 Lesson Three: The last stage had 6 taggers and 6 runners, limited resources, and a 70-second time limit. short cooldowns and close conditions called for accuracy and strategy.

6.2 Behaviour Cloning Training

In our project on Freeze Tag, we used Behavioral Cloning (BC) with the Generative Adversarial Imitation Learning (GAIL) algorithm to train the tagger agent in a multi-agent setting. The tagger was trained through imitation learning by following expert human demonstrations, and the runner agents were run using a pre-trained model built based on Curriculum Learning. Demonstrations were captured on the last and most difficult level of the environment, making it possible to train the tagger directly in the most difficult situation. 200 episodes of demonstration data were gathered to ensure a rich and varied set of training data. GAIL improved the imitation process by allowing the tagger to generalize past literal imitation, learning strong behaviors by adversarial training without the need for explicit reward signals. This solution greatly enhanced training effectiveness and agent performance in high-stakes, dynamic game-playing.

VII. THE TESTING SYSTEM

To measure the performance of various training protocols for tagger agents, we created a controlled experimental framework comparing Curriculum Learning (CL) and Behaviour Cloning (BC). The two types of taggers were compared against a shared baseline of runner agents that were trained with CL to provide an unbiased comparison. Both tagger types were tested on 25 episodes under the same conditions, including agent capabilities, reward schemes, and item settings. Major indicators like freezes, shot attempts, runner survival rates, and win/loss results were tracked and interpreted. This framework made it possible to have an easy evaluation of the strengths and weaknesses of each training method in the cooperative-competitive dynamics of the Freeze Tag setting.

7.1 Metrics and Performance Measurement:

The performance of the Freeze Tag game is gauged by major gameplay indicators that represent the competency and synergy of the agents in the environment. These are: runner wins, tagger wins, time episode, length of episode, freeze balls picked up, wall balls picked up, total freezes, total unfreezes, walls activated, freeze balls activated, freeze ball hits, touches freezes, agents frozen at end of episode, total unfreeze time, fastest unfreeze, average unfreeze, wall hits to tagger, wall hits to freeze ball projectiles, failed unfreeze attempts, longest survival after unfreeze, and shortest survival after unfreeze. Increased values of wins, freezes, unfreezes, and freeze ball use represent better performance, whereas decreased values of losses, time-outs, and episode length represent greater efficiency and strategy. These measures were chosen to reflect the essential goals of the Freeze Tag setting, where victory relies on prompt actions, resource allocation, and collaboration among agents. Through these metrics, we compare agents trained with various learning methods based on their capacity to learn, act strategically, and affect team performance within a dynamic multi-agent environment.

VIII. RESULTS

Here, we outline the results that are derived from the testing framework.

Metric	BC	CL
Win Rate (Tagger)	69.2%	65.4%
Avg. Episode Length (s)	35.1s	41.3s
Freeze Balls Collected	13.6	77.52
Wall Balls Collected	20.2	18.0
Total Freezes	4.7	5.3
Total Unfreezes	1.2	2.0
Total Time Spent Freezing	12.9s	25.6s
Wall Hits to Tagger	4.7	9.3
Failed Unfreeze Attempts	2.0	1.2

Table 5: Raw results of tests

8.1 Observations

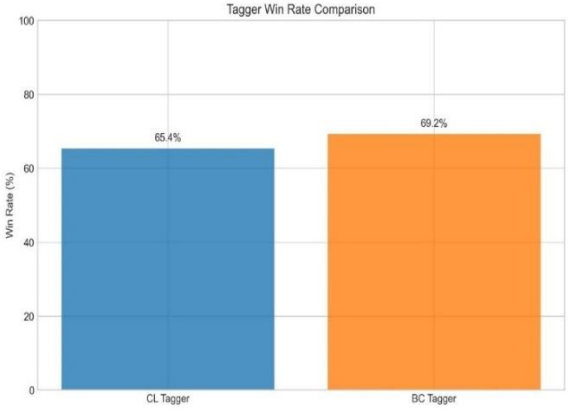
In this section, we will describe the observations made during training and testing.

Tagger Win Rate: BC-trained taggers slightly outperformed CL agents (69.2% vs 65.4%), suggesting a minor edge in strategy adoption through imitation.

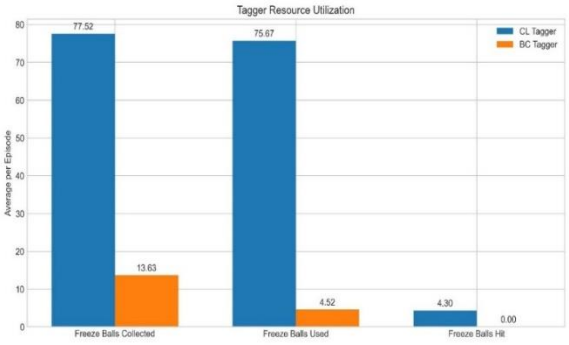
Freeze Ball Collection: CL agents collected far more freeze balls (77.5 vs. 13.6), indicating a resource-focused, exploratory strategy.

Freezing Actions: CL agents had more total freezes (5.56 vs. 4.93), showing better use of freeze balls for strategic play.

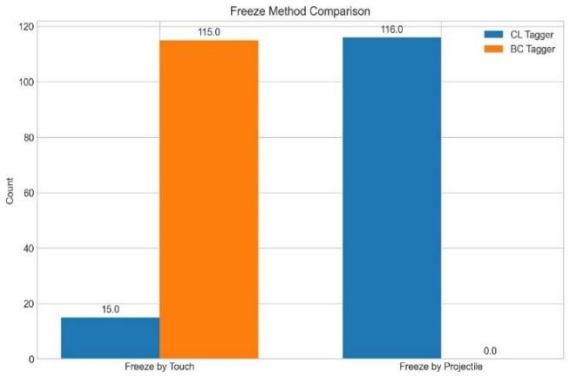
Direct Freezes (Touch): BC agents had more direct freezes (4.26 vs. 0.56), reflecting a close-combat style learned from demonstrations.
Frozen Runners at End: BC agents left more runners frozen (3.63 vs. 3.30), suggesting a faster, aggressive playstyle.
Wall Collisions: CL agents showed more wall interactions (10.4 vs. 4.8) and projectile-wall hits (2.1 vs. 0.04), indicating more active movement and tactical use of projectiles.



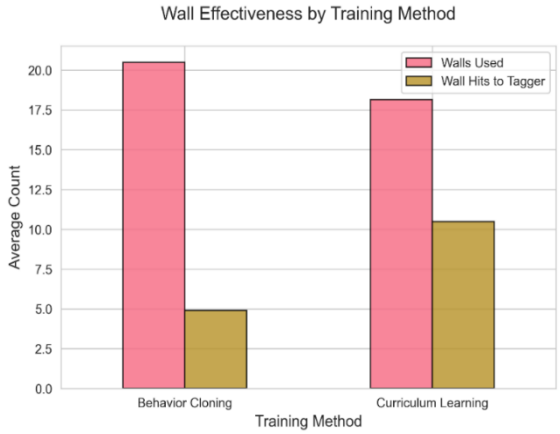
Graph 1: Win Rate (CL vs BC)



Graph 2: Resource Utilization by Tagger Agent



Graph 3: Freeze Method used by Tagger Agent



Graph 4: Wall Effectiveness Against Tagger Agents

Overall:
Learning Curriculum encourages agents that explore intensely with their surroundings, using resources such as freeze balls and moving more dynamically across the map. These agents are more adaptable and strategic in their approach. By contrast, Behavior Cloning produces agents that mirror expert actions closely, tending to perform well in simple cases but being poor at long-term planning as well as innovative problem-solving. In dynamic or novel environments, CL agents should be superior to adapt. BC agents are more likely to converge quickly on acquired behaviors but can be poor in new settings.

IX. CONCLUSION

This work compares Behavior Cloning (BC) and Curriculum Learning (CL) in tagger agent training in the Freeze Tag environment, emphasizing strategic learning vs. imitation-based behavior trade-offs. Although BC agents won at a slightly higher rate, they used extensive direct contact to freeze runners and exhibited minimal usage of strategic mechanics like freeze ball collection. By comparison, CL agents were more intentional, resource-based in their actions, with increased environmental interaction and improved employment of projectile-based tactics. These results indicate that CL produces more generalizable, context-sensitive agents with the ability to plan strategically and generalize. Though BC is useful for quick skill copying in static environments, CL is more stable and scalable for dynamic multi-agent worlds. Thus, Curriculum Learning is a valuable paradigm for training intelligent long-term agent behavior in realistic gameplay situations.

X. FUTURE SCOPE

Future work for this project involves investigating more advanced RL algorithms such as PPO variants and transformer models to enhance agent coordination. Adding vision-based inputs is possible to facilitate richer perception and greater generalization. Moving into dynamic worlds with intricate maps and stochastic components will pose further challenges to agents and enhance adaptability. Also, combining self-play and adversarial training is possible to create more durable strategies. Transferring these agents into real-world domains like multi-robot systems and autonomous coordination tasks is a long-term objective.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[2] Pedro Almeida, Vítor Carvalho, AlbertoSimões "Reinforcement Learning as an Approach to Train Multiplayer First Person Shooter Game Agents."

[3] Taresh Dewan, Aloukik Aditya, Manva Trivedi Ao Chen " The Use of Reinforcement Learning in Gaming:The Breakout Game Case Study ".

[4] McPartland, M.; Gallagher, M. Reinforcement Learning in First Person Shooter Games. IEEE Trans.Comput. Intell. AI Games 2011, 3, 43–56. [CrossRef]

[5] Unity Team. The ML-Agent's Github Page. Available online: https://github.com/Unity-Technologies/ml-agents (accessed on 20 June 2023).

[6] Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V; Mastering the game of Go with deep neural networks and tree search. Nature 2016,529, 484–489. [CrossRef]

[7] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning:An introduction. MIT press, 2018

[8] Gagniuc, Paul A. Markov chains: from theory to implementation and experimentation. John Wiley & Sons, 2017

[9] K. Arulkumaran, M.P. Deisenroth, M. Brundage and A.A. Bharath, "Deep reinforcement learning: A brief survey," IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, 2017

[10] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup and D. Meger, "Deep reinforcement learning that matters," In Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1, April 2018

[11] C. Szepesvári, "Algorithms for reinforcement learning," Synthesis lectures on artificial intelligence and machine learning, vol. 4, no. 1, pp. 1-103, 2010.

[12] D.J. Soemers, C.F. Sironi, T. Schuster and M.H. Winands, "Enhancements for real-time Monte-Carlo tree search in general video game playing," In 2016 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1-8, September 2016.

[13] Z. Wei, D. Wang, M. Zhang, A.H. Tan, C. Miao and Y. Zhou, "Autonomous agents in snake game via deep reinforcement learning," In 2018 IEEE International conference on Agents (ICA), pp. 20-25, July 2018.

[14] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In Proceedings of the 12th International Conference on Machine Learning (ICML 1995), pages 30–37. Morgan Kaufmann, 1995.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms,"arXiv.org,https://arxiv.org/abs/1707.06347, arXiv:1707.06347[cs.LG].