Implementation and Performance Evaluation of Curriculum Learning and Behavior Cloning for Multi-Agent Freeze Tag Environment using Unity ML-Agents

Sachin Shankarrao Damre[1], Dr. Sushil Kulkarni[2]
[1]*M. tech 3rd Semester Student, Department of Computer Science and Information Technology, MBES College of Engineering, Ambajogai, India*
[2]*Head and Guide, Department of Computer Science and Information Technology, MBES College of Engineering, Ambajogai, India*

***Abstract*—Recent advances in Reinforcement Learning (RL) have enabled autonomous agents to learn complex decision-making strategies through continuous interaction with dynamic environments. Among the various training methodologies, Curriculum Learning (CL) and Behavior Cloning (BC) have gained significant attention for improving learning efficiency and policy quality in multi-agent systems. Curriculum Learning adopts a progressive training strategy by gradually increasing task complexity, allowing agents to acquire foundational skills before solving more challenging scenarios. In contrast, Behavior Cloning leverages expert demonstrations to directly imitate desired behaviours, thereby reducing the exploration burden during the initial stages of training.**

This paper presents the design, implementation, and comparative evaluation of Curriculum Learning and Behavior Cloning within a custom-developed 3v3 Freeze Tag environment implemented using the Unity ML-Agents framework. The environment models a cooperative-competitive game in which runner agents attempt to survive and rescue teammates while tagger agents seek to freeze all opposing agents within a predefined time limit. The proposed implementation integrates Proximal Policy Optimization (PPO) for reinforcement learning and combines Behavior Cloning with Generative Adversarial Imitation Learning (GAIL) to enhance policy learning from expert demonstrations.

***Index Terms*—Reinforcement Learning, Deep Reinforcement Learning, Unity ML-Agents, Curriculum Learning, Behavior Cloning, Generative Adversarial Imitation Learning, Proximal Policy Optimization, Freeze Tag Environment, Multi-Agent Systems, Artificial Intelligence.**

I. INTRODUCTION

Artificial Intelligence (AI) has transformed the development of autonomous systems by enabling software agents to learn optimal behaviours directly from interactions with complex environments. Traditional rule-based game agents rely heavily on manually designed heuristics, making them difficult to adapt to changing scenarios or unforeseen situations. Reinforcement Learning addresses this limitation by allowing agents to improve their decision-making abilities through continuous trial-and-error interactions with the environment while maximizing cumulative rewards.

Recent breakthroughs in Deep Reinforcement Learning have demonstrated remarkable success across numerous challenging domains, including Atari video games, robotic manipulation, autonomous driving, industrial process optimization, and multi-agent strategic games. Algorithms such as Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and Soft Actor-Critic (SAC) have significantly improved the capability of autonomous agents to solve high-dimensional decision-making problems without requiring explicit programming.

To facilitate effective learning, the environment incorporates several advanced reinforcement learning concepts, including ray-based perception sensors, continuous movement control, reward shaping, curriculum scheduling, demonstration recording, and event-driven interactions. The implementation further supports progressive difficulty adjustment through

multiple curriculum lessons, allowing agents to gradually transition from basic navigation to sophisticated team-based tactical behaviour.

II. LITERATURE REVIEW

The rapid advancement of Artificial Intelligence (AI) has significantly transformed the field of autonomous decision-making systems. Reinforcement Learning (RL), a branch of machine learning, enables agents to learn optimal policies through continuous interaction with an environment instead of relying on manually programmed rules. During the past decade, reinforcement learning has evolved from simple tabular algorithms to sophisticated deep learning architectures capable of solving highly complex gaming and robotics problems. This evolution has motivated researchers to investigate new learning strategies that improve convergence speed, policy stability, and cooperative behaviour in multi-agent environments.

One of the earliest breakthroughs in Deep Reinforcement Learning was achieved through the Deep Q-Network (DQN) algorithm, which demonstrated that deep neural networks could successfully learn control policies directly from raw pixel inputs while playing Atari 2600 games. Although DQN significantly improved reinforcement learning performance, it was primarily designed for discrete action spaces and often suffered from overestimation bias, unstable convergence, and poor scalability in multi-agent environments.

To overcome these limitations, several policy-gradient algorithms were introduced, among which Proximal Policy Optimization (PPO) has become one of the most widely adopted algorithms for continuous control problems. PPO stabilizes policy updates by restricting excessive parameter changes during optimization, thereby achieving improved convergence while maintaining computational efficiency. Due to its robustness and simplicity, PPO has become the default reinforcement learning algorithm within the Unity ML-Agents Toolkit and has been successfully applied in robotics, autonomous navigation, and intelligent game agent development.

Problem Statement

Training intelligent agents for cooperative multi-agent environments remains a challenging task because agents must simultaneously learn navigation, adversarial interactions, resource collection, teammate coordination, and long-term strategic planning. Conventional reinforcement learning approaches often require extensive exploration before converging to effective policies, whereas imitation learning methods depend heavily on the availability and quality of expert demonstrations.

Therefore, a comprehensive implementation is required to investigate whether Curriculum Learning or Behavior Cloning provides superior learning performance when both methods are trained and evaluated under identical Freeze Tag game environments using Unity ML-Agents.

The implementation aims to compare both methodologies in terms of convergence speed, strategic behaviour, resource utilization, cooperation efficiency, learning stability, and overall game performance.

III. RESEARCH OBJECTIVES

The primary objectives of this research are:

- To design and implement a realistic 3v3 Freeze Tag environment using the Unity ML-Agents framework.
- To implement Curriculum Learning using the PPO algorithm for progressive reinforcement learning.
- To implement Behavior Cloning with GAIL using expert demonstrations collected from the developed environment.
- To design an effective observation space and reward engineering mechanism supporting cooperative and competitive gameplay.
- To evaluate both learning methodologies using standardized experimental metrics including win rate, episode duration, freeze frequency, resource utilization, wall interactions, and teammate recovery.
- To analyze the strengths and limitations of Curriculum Learning and Behavior Cloning for intelligent multi-agent decision-making.
- To provide practical recommendations for selecting appropriate learning methodologies in future reinforcement learning applications involving cooperative autonomous agent.

IV. PROPOSED SYSTEM

4.1 Overview of the Proposed System

The proposed system implements an intelligent multi-agent reinforcement learning framework within a custom designed Freeze Tag environment developed using the Unity ML-Agents Toolkit. The objective is to investigate the effectiveness of two distinct learning paradigms—Curriculum Learning (CL) and Behavior Cloning (BC)—for training autonomous agents in a cooperative-competitive game environment.

The environment consists of two opposing teams: Runner agents and Tagger agents. Runner agents aim to survive until the episode concludes while assisting frozen teammates by unfreezing them. Conversely, Tagger agents attempt to freeze all runners before the episode time limit expires. Both teams interact with a shared environment containing collectible resources such as Freeze Balls and Wall Balls, which influence strategic decision-making during gameplay.

The proposed framework integrates multiple AI components, including reinforcement learning, imitation learning, curriculum scheduling, reward engineering, ray-based perception, and event-driven interactions. These components collectively enable agents to learn complex cooperative and adversarial behaviours without relying on manually programmed strategies.

Unlike traditional game AI systems that use predefined rules, the proposed implementation enables agents to continuously improve through interaction with the environment. The combination of PPO-based reinforcement learning and demonstration-driven imitation learning allows the study to compare exploration-based learning against supervised policy initialization under identical experimental conditions.

4.2 System Architecture

The overall architecture of the proposed system consists of five major layers:

1. Unity Simulation Environment

The Unity game engine serves as the simulation platform where all physical interactions, object movements, collisions, and episode management occur. The environment includes:

- Runner agents
- Tagger agents
- Freeze Balls
- Wall Balls
- Dynamic walls
- Episode controller
- Reward manager
- Curriculum controller

Unity provides realistic physics simulation, collision detection, and event management while communicating with the external training process through the ML-Agents interface.

2. Agent Observation Layer

Each intelligent agent continuously observes its surrounding environment before selecting an action.

The observation layer collects information including:

- Current position
- Forward direction
- Agent velocity
- Frozen state
- Cooldown timers
- Available resources
- Percentage of frozen teammates
- Nearby walls
- Opponent positions
- Collectible objects

In addition to vector observations, each agent utilizes Ray Perception Sensors, allowing detection of surrounding objects without explicit positional knowledge. These observations become the input state for the reinforcement learning policy network.

3. Decision-Making Layer

The decision-making layer contains the trained neural network responsible for selecting actions based on the observed environment.

Two independent learning approaches are implemented:

Curriculum Learning Pipeline

- PPO Neural Network
- Progressive curriculum scheduler
- Reward optimization
- Continuous policy improvement

Behavior Cloning Pipeline

- Expert demonstrations
- Behavioral cloning initialization
- GAIL discriminator
- PPO fine-tuning

Although both pipelines ultimately produce action policies, the learning mechanisms differ significantly, enabling a fair comparative analysis.

4. Environment Interaction Layer

Once an action has been selected, the agent interacts with the Unity environment.

Typical actions include:

Runner Agents

- Move
- Rotate
- Collect Wall Balls
- Build defensive walls
- Rescue teammates

Tagger Agents

- Move
- Rotate
- Collect Freeze Balls
- Shoot Freeze Balls
- Freeze runners by direct contact

Following each action, Unity computes environmental changes and returns updated observations and rewards.

5. Learning and Evaluation Layer

The final layer updates the learning policy based on the observed rewards.

For Curriculum Learning:

- PPO computes policy gradients.
- Rewards update network parameters.
- Curriculum scheduler advances lesson difficulty.

For Behavior Cloning:

- Expert demonstrations initialize policy.
- GAIL discriminator compares expert and generated trajectories.
- PPO refines the learned policy through interaction.

Training continues until predefined convergence criteria are achieved.

4.3 Overall Workflow

The proposed implementation follows an iterative reinforcement learning workflow consisting of multiple training episodes.

Step 1: Environment Initialization

The Freeze Tag environment initializes:

- Runner agents
- Tagger agents
- Collectible resources
- Obstacles
- Episode timer
- Curriculum parameters

All agents begin from randomized initial positions to prevent policy over fitting.

Step 2: Observation Collection

Each agent gathers environmental information using:

- Vector observations
- Ray perception sensors
- Internal status variables
- Team statistics

These observations form the current state representation.

Step 3: Action Selection

The policy network predicts actions based on the current observations.

Possible actions include:

- Navigation
- Rotation
- Shooting Freeze Balls
- Wall creation
- Rescue actions

The selected action depends on the learning methodology being evaluated.

Step 4: Environment Response

Unity updates the simulation according to the chosen action.

Possible outcomes include:

- Freeze successful
- Runner rescued
- Wall constructed
- Resource collected
- Collision detected

The environment then calculates immediate rewards.

Step 5: Reward Assignment

The reward manager computes numerical rewards according to predefined reward functions.

Positive rewards encourage:

- Team victories
- Resource collection

- Strategic gameplay
- Successful freezing
- Teammate rescue

Negative rewards discourage:
- Defeat
- Being frozen
- Poor coordination
- Inefficient resource usage

Step 6: Policy Update
The PPO optimizer updates policy parameters.
Curriculum Learning additionally evaluates whether lesson progression conditions have been satisfied.
Behavior Cloning continues refining policies through GAIL after demonstration-based initialization.

Step 7: Performance Evaluation
Training metrics are exported periodically for analysis.
Recorded metrics include:
- Win rate
- Episode length
- Freeze count
- Unfreeze count
- Freeze Ball usage
- Wall Ball collection
- Resource utilization
- Tactical interactions

4.4 Agent Pipeline
The proposed system employs two specialized intelligent agent types with distinct objectives and decision-making processes.
Runner Agent Pipeline

Environment Observation
↓
Vector Observations
↓
Ray Perception
↓
Policy Network
↓
Movement Decision
↓
Wall Creation / Rescue
↓
Reward Calculation
↓
Policy Update

The runner policy emphasizes:
- Survival
- Cooperation
- Resource collection
- Defensive wall placement
- Teammate recovery

Tagger Agent Pipeline

Environment Observation
↓
Freeze Ball Availability
↓
Ray Detection
↓
Policy Network
↓
Target Selection
↓
Freeze Ball Attack
↓
Reward Calculation
↓
Policy Update

The tagger policy focuses on:
- Efficient target pursuit
- Freeze Ball management
- Offensive positioning
- Team coordination
- Episode completion

4.5 Advantages of the Proposed Framework
The proposed implementation offers several advantages over conventional reinforcement learning environments:
- Integration of reinforcement learning and imitation learning within a single experimental framework.
- Progressive curriculum scheduling that improves policy convergence and training stability.
- Event-driven reward engineering encouraging cooperative and adversarial behaviors.

- Ray-based perception supporting realistic environmental awareness without requiring global state information.
- Comprehensive evaluation using behavioral metrics beyond cumulative reward.
- Modular architecture that supports future extension with additional learning algorithms, larger environments, or self-play mechanisms.

V. METHODOLOGY

5.1 Unity ML-Agents Framework

The proposed environment is implemented using the Unity ML-Agents Toolkit, which provides a comprehensive framework for training intelligent agents in physics-based simulations. Unity offers several advantages over conventional reinforcement learning environments, including realistic physics, customizable game mechanics, event-driven interactions, and seamless integration with modern deep learning libraries.

Within the Unity framework, every agent inherits from the Agent class and follows a continuous interaction cycle consisting of observation collection, action execution, reward calculation, and policy updates. This interaction enables agents to improve their behaviour through repeated exposure to diverse environmental conditions.

The Unity environment also manages episode initialization, collision detection, resource spawning, timer control, and curriculum progression. Communication between Unity and the Python training environment occurs through the ML-Agents API, where observations are transmitted to the neural network, and selected actions are returned to Unity for execution.

This architecture enables efficient synchronization between simulation and training while maintaining reproducibility across multiple experimental runs.

5.2 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is selected as the primary reinforcement learning algorithm because of its stability, computational efficiency, and suitability for continuous control problems.

Unlike traditional policy-gradient algorithms, PPO constrains policy updates by limiting the magnitude of parameter changes during optimization. This clipping mechanism prevents unstable learning caused by excessively large policy updates while preserving sufficient exploration.

During each training iteration, the PPO algorithm performs the following sequence:

1. Collect observations from all agents.
2. Execute actions predicted by the policy network.
3. Receive rewards from the environment.
4. Compute discounted returns and advantages.
5. Update actor and critic networks using clipped surrogate objectives.
6. Repeat until convergence.

The actor network predicts the probability distribution of possible actions, whereas the critic network estimates the expected future return of each state. Together, these networks allow the agent to balance exploration and exploitation effectively.

The selection of PPO is particularly advantageous for the Freeze Tag environment because the game involves continuous movement, dynamic interactions, and cooperative decision-making among multiple agents.

5.3 Behavior Cloning

Behavior Cloning represents a supervised learning approach in which agents imitate expert demonstrations instead of discovering behaviours entirely through exploration.

In the proposed implementation, demonstration data are collected by manually controlling the runner agent in the most challenging Freeze Tag environment. The recorded trajectories include observations, actions, rewards, and environmental states.

The Behavior Cloning training process consists of the following steps:

1. Record expert gameplay demonstrations.
2. Store demonstration trajectories using Unity Demonstration Recorder.
3. Convert trajectories into training samples.
4. Train the policy network using supervised learning.
5. Initialize reinforcement learning with the cloned policy.

Unlike Curriculum Learning, Behavior Cloning immediately provides the agent with effective behaviours, significantly reducing exploration time during early training.

However, because the learned policy depends heavily on demonstration quality, the approach may struggle when encountering states that were absent from the demonstration dataset.

5.6 Generative Adversarial Imitation Learning (GAIL)
To overcome the limitations of conventional Behavior Cloning, the implementation integrates Generative Adversarial Imitation Learning (GAIL).
GAIL extends imitation learning through an adversarial optimization process involving two competing neural networks:

- Generator: Represents the agent policy attempting to imitate expert behaviour.
- Discriminator: Distinguishes between expert demonstrations and generated trajectories.

During training, the policy continuously attempts to produce behaviours indistinguishable from expert demonstrations, while the discriminator improves its ability to detect non-expert behaviour. This adversarial process gradually refines the policy and enhances its generalization capability.
Combining Behavior Cloning with GAIL enables agents to retain expert knowledge while continuing to improve through reinforcement learning.

VI. DISCUSSION

The comparative evaluation highlights the complementary strengths of Curriculum Learning & Behavior Cloning.
Curriculum Learning encourages gradual acquisition of increasingly sophisticated behaviours. By progressively increasing task complexity, agents learn navigation, resource management, cooperation, and tactical decision-making without becoming overwhelmed by the full complexity of the environment. This results in policies that generalize effectively to unseen situations and demonstrate robust long-term performance. Behavior Cloning, in contrast, substantially reduces the exploration burden by leveraging expert knowledge. Agents achieve competent behaviour much earlier in the training process, making this approach particularly valuable when high-quality demonstrations are available. Nevertheless, cloned policies may struggle in states not represented within the demonstration dataset, reducing adaptability under novel conditions.

The experimental findings suggest that a hybrid training strategy combining Behavior Cloning for policy initialization with Curriculum Learning and PPO for subsequent optimization offers a promising direction for future research. Such an approach can exploit the rapid convergence of imitation learning while retaining the adaptability and robustness of reinforcement learning.

VII. CONCLUSION

This research presented the implementation and comparative evaluation of Curriculum Learning and Behavior Cloning within a custom-developed multi-agent Freeze Tag environment using Unity ML-Agents.
A comprehensive reinforcement learning framework incorporating PPO, GAIL, reward engineering, ray-based perception, and progressive curriculum scheduling was successfully implemented. Experimental evaluation demonstrated that Behavior Cloning provides faster initial convergence through expert demonstrations, whereas Curriculum Learning develops more adaptable and cooperative policies over extended training.
The results indicate that no single learning paradigm is universally optimal. Instead, the choice of methodology should be guided by application requirements, availability of expert demonstrations, and the desired balance between learning efficiency and long-term adaptability.
The proposed implementation establishes a reproducible framework for investigating reinforcement learning algorithms in cooperative multi-agent environments and contributes practical insights applicable to intelligent robotics, autonomous systems, and game artificial intelligence.

VIII. FUTURE SCOPE

Future work may extend the proposed implementation in several directions:

- Integration of self-play reinforcement learning to improve policy robustness.
- Investigation of transformer-based reinforcement learning architectures.
- Automatic curriculum generation using adaptive difficulty estimation.

- Multi-environment transfer learning for improved generalization.
- Explainable reinforcement learning techniques for policy interpretation.
- Larger-scale environments involving heterogeneous agent roles.
- Real-world robotic validation using Unity-to-ROS integration.

REFERENCES (ILLUSTRATIVE)

[1] V. Mnih *et al*., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017. [Online]. Available: https://arxiv.org/abs/1707.06347.

[3] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *Proc. 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 1889–1897. [Online]. Available: https://arxiv.org/abs/1502.05477.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[5] Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–35, 2017, doi: 10.1145/3054912.

[6] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016. [Online]. Available: https://arxiv.org/abs/1606.03476.

[7] Unity ML-Agents Toolkit Documentation, Unity Technologies, accessed Jul. 6, 2026.

[8] OpenAI: Learning Dexterous In-Hand Manipulation, OpenAI, "Learning Dexterous In-Hand Manipulation," 2018, accessed Jul. 6, 2026.

[9] DeepMind: Playing Atari with Deep Reinforcement Learning, V. Mnih *et al*., "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.