Improving Sample Efficiency in Multi-Agent Reinforcement Learning for Simulated Football Games via Exploration

Amir Baghi
SEED - Electronic Arts (EA)
Stockholm, Sweden
abaghi@ea.com

Jens Sjölund
Dept. of Information Technology
Uppsala University
Uppsala, Sweden
jens.sjolund@it.uu.se

Joakim Bergdahl
SEED - Electronic Arts (EA)
Stockholm, Sweden
jbergdahl@ea.com

Linus Gisslén
SEED - Electronic Arts (EA)
Stockholm, Sweden
lgisslen@ea.com

Alessandro Sestini
SEED - Electronic Arts (EA)
Stockholm, Sweden
asestini@ea.com

arXiv:2503.13077v2 [cs.LG] 4 Aug 2026

Abstract

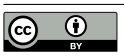
Multi-agent reinforcement learning has shown promise in learning cooperative behaviors in team-based environments. However, such methods often demand extensive training time, which inhibits their application for game-AI in standard game development. For instance, the state-of-the-art method TiZero takes 40 days to train high-quality policies for a football environment. In this paper, we hypothesize that better exploration mechanisms can improve the sample efficiency of multi-agent methods. Thereby, we propose utilizing a random network distillation bonus within the multi-agent TiZero framework, aiming to promote exploration. Additionally, we introduce architectural modifications to the original algorithm to enhance TiZero's computational efficiency. We evaluate the sample efficiency of our approach against original TiZero through extensive experiments. Our results show that random network distillation improves the sample efficiency per training phase by 13.3% compared with the original TiZero, enhancing generalization and adaptability to previously difficult scenarios. This highlights the better applicability of our variant in practical game development settings. Lastly, we qualitatively evaluate the gameplay of the produced models against a heuristic AI. We find that random network distillation leads to a higher accuracy in shooting, and it achieves higher behavioral stability as shown by the lower standard deviation achieved in gameplay metrics. The code is available at **https://github.com/electronicarts/marling**.

CCS Concepts

• **Computing methodologies** → **Multi-agent systems**; **Multi-agent reinforcement learning**; • **Applied computing** → **Computer games**.

Keywords

Multi-agent Reinforcement Learning, Sample Efficiency, Exploration, Game-AI



This work is licensed under a Creative Commons Attribution 4.0 International License.
FDG '26, Copenhagen, Denmark
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2495-4/2026/08
https://doi.org/10.1145/3815598.3815627

ACM Reference Format:
Amir Baghi, Jens Sjölund, Joakim Bergdahl, Linus Gisslén, and Alessandro Sestini. 2026. Improving Sample Efficiency in Multi-Agent Reinforcement Learning for Simulated Football Games via Exploration. In *Foundations of Digital Games (FDG '26), August 10–13, 2026, Copenhagen, Denmark.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3815598.3815627

1 Introduction

Multi-Agent Reinforcement Learning (MARL) is a natural extension of Reinforcement Learning (RL) to scenarios where multiple learning agents need to interact – through cooperation or competition – to reach a pre-defined team-based or globally common goal [1]. Its use spans a wide range of use-cases and problem spaces, such as multi-robot warehouse management and autonomous driving of multiple vehicles [6, 9].

Recently, MARL has been employed as game-AI in multi-player and multi-character game environments, where cooperation and competition are common challenges facing human players and bots alike [22, 25]. A notable example is AlphaStar, which successfully applied MARL in the real-time strategy video game *StarCraft II* [22]. Another use-case for MARL is controlling a team of football players in a game [10, 12]. Football is a useful testbed for multi-agent control methods, as it encapsulates the challenges of strategy, cooperation, and competition. Moreover, it has proven difficult to create human-like, emergent and representative team-based behavior for football via hand-crafted and scripted methods, which often fail to adapt dynamically to new situations [11].

Many notable MARL methods have been proposed that successfully learn useful policies for different domains [12, 25]. However, many of these methods suffer from poor sample efficiency, ultimately leading to excessive training times for reaching gameplay-suitable policies. For example, in the work by Lin et al. [12], the authors report a training time of ∼ 40 days across hundreds of processors. In this work, we recognize this as a significant drawback of such methods which inhibits their tractability in traditional game development settings where rapid iteration is essential. We further hypothesize that the sample efficiency of these methods can be improved via *better exploration*, and thus we tackle the problem from this perspective. Our contribution here is empirical, i.e., we



Figure 1: A screenshot of an 11-vs-11 scenario in the Google research football environment we use as a testbed for our training experiments and evaluations.

investigate this hypothesis through extensive experiments and ablations that showcase how better exploration translates into sample-efficiency gains, exhibited within the challenging team-based football domain. Additionally, we limit the scope and duration of our experiments here to a few days rather than multiple weeks or long horizons such as 40 days, due to our focus on a more rapid and practical development framework.

Thus, we focus on improving *TiZero*, a state-of-the-art multi-agent algorithm for team-based football game-AI [12]. Specifically, besides methodological and architectural changes, we propose a different exploration-promoting reward extension to TiZero: utilizing an exploration bonus in the form of Random Network Distillation (RND) [2]. We apply this technique, originally proposed for single-agent RL, to MARL, and train and evaluate the resulting methods in the football environment provided by Google Research Football (GRF) [10] and report both training statistics and qualitative gameplay metrics against a heuristic AI. Fig. 1 shows a screenshot of the environment.

In summary, our contributions and results are the following:

(1) We propose methodological and architectural modifications aimed at improving computational performance, as well as an exploration bonus in the form of RND, to enhance the sample efficiency of TiZero. We then empirically demonstrate their effects on training and gameplay performance.
(2) Restricting training to a few days to mirror realistic game-development settings, we evaluate the qualitative behavior of the team-based policies produced by the original TiZero and TiZero augmented with RND in the GRF environment against a baseline heuristic AI (HAI) which was less encountered during training. The results show that each method reaches a gameplay style with its own distinct attributes, with TiZero-RND producing a strategy with a higher accuracy in shooting. It further shows higher stability and certainty in its converged behavior compared to the original TiZero due to lower variance in reported metrics.
(3) We conduct an ablation study of fixed positional encoding of player IDs and its impact on training sample efficiency and stability compared to the original MLP-based encoding approach.

2 Related Work

In this section, we review relevant works in RL and MARL with a special focus on game-AI, football, reward shaping, and RL exploration.

2.1 Reinforcement Learning for Game-AI

RL has been gaining interest in the game AI and game development communities due to its potential to generate human-like and adaptive Non-Player Character (NPC) behaviors. Notable examples, such as AlphaStar, OpenAI 5, and GT Sophy [16, 22, 24], demonstrate the potential of RL to reach and even surpass human-level performance. However, the aforementioned approaches primarily focus on achieving super-human performances, regardless of the computational constraints. For instance, the work by OpenAI [16] required months of training. This significantly reduces the applicability of such approaches in a real-world game development pipeline [7].

At the same time, recent literature has shifted towards developing agents that are practical for use within game production environments. Some notable examples include: the work by Zhang et al. [29] that trained a DRL agent for the game Arena Breakout; the work by Sestini et al. [18], who proposed a sample-efficient single-agent framework enabling game designers to train and modify the agent within a few hours; and Shūkai, a practical DRL algorithm specifically tailored for commercial fighting games [28]. Although these works recognize sample inefficiency as one of the primary challenges limiting the applicability of RL in production, they largely focus on single-agent settings. Our study addresses this gap by presenting initial findings on improving sample efficiency in MARL, towards a more affordable multi-agent game-AI system.

2.2 Multi-Agent Reinforcement Learning in Football

The game of football has been proven to be a challenging multi-agent testbed. Therefore, recent research tackles this as a MARL problem. As a state-of-the-art method, TiZero employs a curriculum-based self-play strategy, learning a cooperative policy via modifications to the Multi-Agent PPO (MAPPO) policy loss while employing Centralized Training with Decentralized Execution (CTDE) [12]. This method reaches a high win-rate of $\sim 95\%$ against strong baselines. Fictitious cross-play – another state-of-the-art MARL method – couples self-play with the policy-space response oracle technique to achieve win-rates exceeding 94% in GRF [26]. In this paper, we base our work on TiZero as an exemplary and effective MARL method for football. However, TiZero and similar MARL methods are known to be sample inefficient, with the former requiring 40 days of training to reach strong gameplay performance. Similar to recent research, such as the work by Xu et al. [25], we aim to improve sample efficiency through reward shaping and exploration.

Among the available MARL football environments, *Google Research Football (GRF)* offers an intermediate level of abstraction for high-level team-based strategies while maintaining individual player control [10]. In contrast, JiDi Olympics Football simplifies the game by focusing primarily on movement and ball control without complex team strategies [23]. At the other extreme, MuJoCo Multi-Agent Soccer emphasizes low-level robotic control of players [14]. The balanced nature of GRF makes it the preferred choice for

our work, as it encapsulates both short-term individual skills and long-term strategic decision-making.

2.3 Reward Shaping and Exploration

Reward shaping is a technique in RL where the reward function is augmented with an additional term based on the agent transitions and heuristics regarding the desired goal, environment, and general agent behavior. It has been shown to accelerate learning by providing a dense, guiding, and intermediate signal of reinforcement in sparse environments [15].

A common form of reward shaping is intrinsic motivation. This technique rewards the agent for the inherent "satisfaction" of discovering new knowledge and skills [3]. Intrinsic motivation thereby encourages agents to autonomously discover skills that can be useful to achieve the end goal. As an example technique in reward shaping and intrinsic motivation, Random Network Distillation (RND) encourages exploration by providing intrinsic rewards based on prediction errors on a fixed, randomly-initialized neural network [2].

Specifically within exploration, a commonly employed technique in RL is entropy regularization, which promotes an exploratory optimization process by encouraging the agent to maintain stochasticity in its action selection. It has been utilized in the actor loss function of policy gradient methods, e.g., MAPPO [27, 30]. TiZero, which is the basis of our study, utilizes entropy regularization in its actor loss function as well, and so, it already benefits from one form of exploration.

Beyond entropy regularization, RND has shown effective exploration even in games [19], and thus, we employ it in this work to address the poor sample efficiency of the MARL algorithm under study.

3 Preliminaries and Setting

Here, we summarize relevant theoretical concepts that will help the reader understand subsequent sections. These include the MARL setting and the training environment.

3.1 Multi-Agent Reinforcement Learning

A MARL problem is typically modeled as a stochastic game, which generalizes the Markov Decision Process (MDP) to multiple agents [13]. A stochastic game is formally defined as:

$$\mathcal{M} = (\mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, P, \{r^i\}_{i\in\mathcal{N}}, \gamma), \tag{1}$$

where $\mathcal{N} = \{1, 2, \ldots, n\}$ is the set of agents, $\mathcal{S}$ is the state space, and $\mathcal{A}_i$ denotes the action space of agent i. The transition function $P : \mathcal{S}\times\mathcal{A}_1\times\cdots\times\mathcal{A}_n\times\mathcal{S} \to [0, 1]$ defines the probability of reaching the next state given the current environment state and joint actions, and each agent receives a reward $r^i : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \to \mathbb{R}$. Moreover, the discounting factor is denoted by $\gamma \in [0, 1]$.

At each time step t, the environment is in state s_t, and each agent selects an action a_t^i according to its policy $\pi^i(a_t^i|o_t^i)$, where o_t^i is the agent's observation, defined as an arbitrary transformation of the environment state s_t. In our case, o_t^i is different and contains specific information for each agent. The joint action $\mathbf{a}_t = (a_t^1, a_t^2, \ldots, a_t^n)$ determines the next environment state s_{t+1} based on P and s_t, and each agent receives a reward $r_t^i = r^i(s_t, \mathbf{a}_t)$. Lastly, the agents aim to maximize their discounted cumulative return:

$$G_t^i = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}^i. \tag{2}$$

Since agents interact in a shared environment, their actions are interdependent, requiring strategies that consider both individual and collective behaviors. Learning in MARL thus involves finding policies π^i that optimize rewards while accounting for coordination, competition, or cooperation among agents.

3.2 Google Research Football and Training Framework

GRF is a physics-based 3D football simulation implemented on top of an existing football game engine [10]. It provides both single and multi-player control options, as well as stochasticity in, e.g., shots, making it appropriate for RL and game-AI research. Moreover, there are pre-defined scenarios for both benchmarking and curriculum learning that include standard football rules, roles, and match states, while allowing for custom scenarios. In this work, we create our own 11-vs-11 scenarios for the curriculum and subsequent learning phases, as described in Section 5. For training, we use the distributed training framework for GRF by Song et al. [20], which is tailored to self-play and multi-agent methods. Furthermore, GRF contains a rule-based heuristic AI (HAI), which actively controls the selected player while applying basic football behavior to non-active ones. HAI accepts a configurable difficulty factor, where higher values correspond to faster decision-making and reaction times. We use this difficulty factor to adjust our curriculum and employ HAI as the baseline for evaluations (see Section 5).

Observation Space. We use customized observation vectors for the actor and critic components (see Section 4.1) following TiZero's design, which is based on GRF's floats observation vector, further augmented with more football-specific features, e.g., offside flags [10, 12]. In total, there are 330 float observation values for the actor and 220 for the critic. The elements of the observation space for the actor and critic are listed in more detail in Appendix B.

Action Space. In this work, we use the action set provided by GRF for the individual agents. These actions include movement in eight directions and various ways to interact with the ball, e.g., shoot or a high pass. Additionally, the agents can sprint (affecting their tiredness), slide tackle, or dribble. In total, we consider 18 discrete actions for each of our agents and exclude GRF's "Built-in AI" action (which transfers control to the HAI) from their action set.

Reward Function. GRF provides two built-in reward functions, and in this work, we use the "SCORING" function which returns +1 for scoring a goal and −1 for conceding one. Moreover, based on TiZero's design [12], we add to this built-in reward four additional dense signals extracted from the environment: a "Hold-Ball Reward" (+0.0001 per time-step when a teammate controls the ball), a "Passing-Ball Reward" (+0.05 for a successful pass), a "Grouping Penalty" (−0.00002 for each teammate pair getting too close and, consequently, −0.001 when the whole team clusters, considering a closeness threshold of 0.05 distance units), and an "Out-of-Bounds Penalty" (−0.001 when an agent moves outside the field boundaries). These constitute our base TiZero reward function, which we extend further in Section 4. Finally, this reward function is zero-sum across

the two teams, i.e., a positive reward for a team gives the equivalent negative for the other.

4 Methodology

In this section, we first describe TiZero as the basis of our method [12], and afterwards, the architectural modifications and other minor variations from the original approach with the aim of computational efficiency. We then explain our proposed augmentation by utilizing the Random Network Distillation (RND) exploration term in order to improve sample efficiency.

4.1 TiZero

TiZero is a MARL method using curriculum learning and self-play to train team-based football-gameplay policies [12]. The method utilizes an actor-critic paradigm where agents are trained together using a global reward signal and evaluated for actions individually, following the CTDE concept. Specifically, TiZero uses a shared actor network for individual agents, differentiating between them by taking player-specific information as input. Moreover, TiZero utilizes a policy loss in the form of Joint-ratio Policy Optimization (JRPO), which is a modification of MAPPO [27]. Specifically the JRPO loss considers a joint policy over the individual policies:

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t) \approx \prod_{i=1}^{n} \pi_\theta^i(a_t^i|o_t^i), \tag{3}$$

where $\pi_\theta^i(a_t^i|o_t^i)$ denotes the individual policy for agent i, and $\mathbf{a}_t$ and $\mathbf{o}_t$ denote the joint action and observations for all agents. This joint policy is then used in the loss below to create the full JRPO actor loss:

$$\begin{aligned} L^{\text{JRPO}}(\theta) = \hat{\mathbb{E}}_t \left[\min\left(\rho_t(\theta)\hat{A}_t, \Pi_{[1-\epsilon,1+\epsilon]}(\rho_t(\theta))\hat{A}_t \right) \right] \\ + \quad \beta\hat{\mathbb{E}}_t \left[H_{\pi_\theta} \right], \end{aligned} \tag{4}$$

where $\rho_t(\theta)$ is the ratio between the new and old joint policies, $\hat{A}_t$ is the global advantage estimation computed via generalized advantage estimation [17], $\Pi_{[1-\epsilon,1+\epsilon]}(\cdot)$ is a projection operator which clips its argument to the range $[1-\epsilon, 1+\epsilon]$ with ϵ as a hyper-parameter, and H_{π_θ} is the entropy of the current joint policy with β as its weighting hyper-parameter. Additionally, the critic uses a Mean Squared Error (MSE) loss, optimizing its value estimation towards a global reward signal for a given environment state.

TiZero trains the final joint policy from scratch through a two-stage pipeline of first curriculum learning and then self-play, which is further divided into two sub-stages: *Challenge* and *Generalize*, inspired by OpenAI 5 and AlphaStar, respectively [16, 22]. The purpose of the curriculum stage is to develop a basic understanding of simple tactics through a progression of football scenarios with increasing difficulties (e.g., moving towards the goal to shoot and score), whereas the self-play stage improves gameplay on top of this basic knowledge.

In self-play, TiZero improves its performance through a two-step training approach. In the *Challenge* phase, it primarily trains against the most recently-learned joint policy, improving strategy by further optimizing on the previously-learned behavior. In the *Generalize* phase, it mainly trains against older versions which the current policy struggles to defeat. This step helps TiZero overcome local optima from earlier training stages and addresses the non-transitivity issue in football policies [4].

4.2 Architectural Modifications

To improve computational efficiency and performance, we propose several architectural changes to TiZero. Firstly, we replace the Long Short-term Memory (LSTM) element in TiZero's actor and critic with a 4-layer Multi-layer Perceptron (MLP). This is because we are concerned with a traditional and industrial game development setting where time and resources are limited, and so, we aim to reduce the computational complexity of the methods under study as much as possible.

Moreover, for the player ID encoding in the actor architecture, we substitute the original MLP with the fixed positional encoding used in Transformer architectures and defined in the work from Vaswani et al. [21]. Fixed positional encodings often provide a more stable representation of player IDs, helping the shared actor network distinguish agents without introducing additional learnable parameters. We evaluate this approach in Section 5, where we ablate this technique and compare to the original MLP-based encoding approach. Our final, modified actor network architecture can be seen in Fig. 2.

We use similar modifications for the critic. Following MAPPO's approach, we also normalize the value estimate targets for better stability during the critic's training and preventing value explosion or vanishing gradients [27].

4.3 Random Network Distillation

As an augmentation of TiZero, we include a state exploration bonus using RND [2]. This technique provides a learned curiosity term that rewards novel and rare states. We extend the single-agent framework of Burda et al. [2] to MARL by defining a global, shared exploration bonus for all agents using a global state representation as the input to the RND module. More specifically, we use two neural networks: a fixed and randomly-initialized target network ψ, and a predictor network $\hat{\psi}$ trained on data collected by the agent. Our RND exploration bonus is then defined as the MSE between these two networks:

$$r_c(s_t) = (\hat{\psi}(s_{t+1}) - \psi(s_{t+1}))^2, \tag{5}$$

where s_t is the global state visited by the agents (i.e., the same one used by our value network) at timestep t. The predictor network trains on the randomly-initialized network's output, and the more frequently a state is visited by agents, the lower the MSE prediction error, and consequently, the exploration bonus will be. Using this bonus, we define our RND-based reward function:

$$R_{\text{RND}}(t) = R_{\text{TiZero}}(s_t, a_t, s_{t+1}) + r_c(s_t). \tag{6}$$

We use MLPs with ReLU activations for both target and predictor networks, with the predictor (four layers) having more layers than the target (two layers). Both networks output four values, with this output dimension treated as a hyper-parameter. We believe combining RND with the entropy term in the actor loss (see Eq. 4) provides a balanced form of exploration, since RND encourages exploration by rewarding novel joint states, while the entropy term maintains policy stochasticity, promoting diverse action selection.

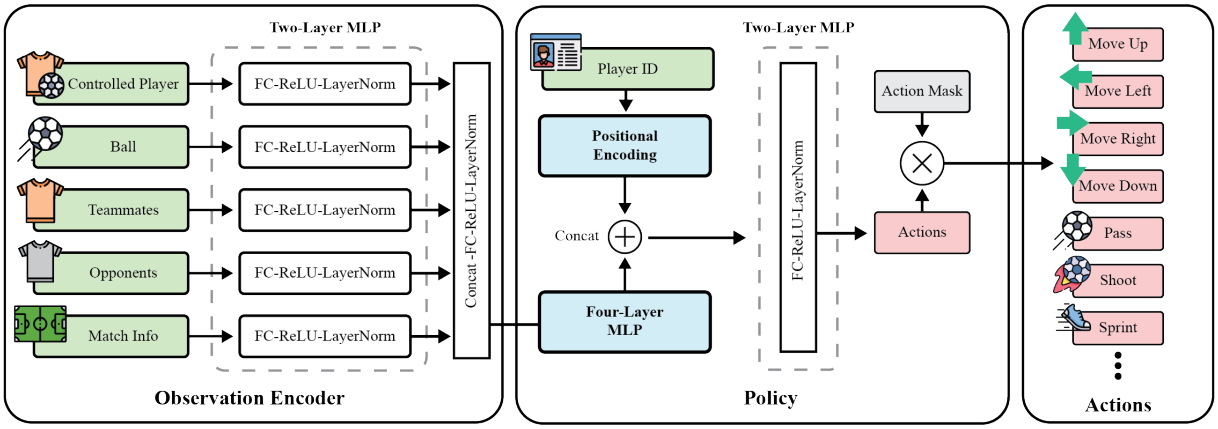


Figure 2: An overview of the architecture of our modified TiZero actor component, with our modifications highlighted in blue. We replace the original LSTM in TiZero with a 4-layer MLP and the original player-ID MLP encoder with fixed positional encodings. This network is shared among agents and by passing their player-related information beside the global ones, the network differentiates among individual agents.

5 Experiments

In this section, we present our training setup, evaluation settings, and results for TiZero and our augmented version using RND. We first describe our curriculum learning and self-play setup, followed by the settings of our training experiments and evaluations. We then present results for the training experiments and analyze the sample efficiency of each method. Finally, we evaluate the generalization of the best trained models against a HAI baseline and provide detailed gameplay statistics and analysis.

Note that our main goal in this work is to improve the sample efficiency of TiZero as a representative MARL method. We conduct training experiments on both the original TiZero and TiZero augmented with RND within a fixed data sample budget. Therefore, since we are not concerned with improving the gameplay quality of the final models, we do not run the methods for 40 days as was done in the original TiZero paper [12]. However, we still carry out gameplay evaluations against a baseline HAI to test the generalization ability of our resulting models.

5.1 Experimental Setup

All training experiments use 5 different seeds and are carried out on machines equipped with an Intel Xeon Gold 6146 CPU @ 3.20GHz (12 cores, 24 logical processors), 256GB of DDR4 RAM, and an NVIDIA GeForce GTX 1080 Ti GPU with 11GB of VRAM.

Curriculum and Self-Play Design. Following TiZero, we employ 10 scenarios with increasing difficulties in our curriculum, where in each scenario, the agents play against the last version of the trained policy in the previous scenario (and GRF's heuristic AI in the first scenario). In each scenario, the agents progress to the next one when they reach a win-rate threshold assigned to the curriculum scenario.

In our curriculum design, we do not use a constant win-rate threshold as done in TiZero. Instead, we set a starting win-rate threshold in the curriculum stage at 55% (slightly above 50% for confidence) for scenario 1 and increase it linearly to 75% by scenario 8. We do so because we found an uneven difficulty gap between curriculum scenarios, specifically from scenario 7 to 8, during the curriculum learning stage of TiZero. We believe the agents spend an unnecessarily long time on simpler scenarios in TiZero's curriculum design, potentially overfitting to basic strategies (e.g., moving into the goal with the ball) that are not particularly effective at later scenarios. Furthermore, we lower the maximum passing win-rate threshold in the curriculum stage from TiZero's 80% to 75%, aiming to maintain a similar performance level while reducing training time requirements. Since the original TiZero paper does not specify a win-rate threshold for their curriculum, we assumed it matched the 80% used in their *Challenge* and *Generalize* stages.

Each curriculum scenario consists of an 11-vs-11 football game with offside disabled. The game lasts at most 500 environment steps but ends immediately if a foul is committed, a player goes out of bounds, or a team scores and is deemed the winner of the scenario. In all cases of termination, the environment is reset. Moreover, the players' positioning is adjusted in each scenario following TiZero's approach [12]. However, in our design, we manually set the GRF player strength factors to `[0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.75, 0.95]` from scenario 1 to 10. We take this approach since we observe in TiZero that a linear increase in GRF's individual player strength factors (affecting their reaction times) does not provide a linear difficulty increase, resulting in uneven difficulty jumps between curriculum scenarios.

After the curriculum stage, the method enters the self-play stage where it alternates between *Challenge* and *Generalize* phases. For the *Challenge* phase, we use a normal football scenario including offside and ending after 3000 environment steps, no longer ending on one team scoring but continuing until the end of the match. The opponent is chosen from a pool of previously trained policies using the approach of OpenAI 5 [16], and the agents move to the *Generalize* phase only after passing a fixed win-rate threshold. We set this win-rate threshold to 75%, as opposed to the 80% threshold believed to be used in TiZero, aiming to reduce the required training time while retaining a reasonable performance level. We use the

Table 1: Sample-efficiency metrics for the curriculum, *Challenge*, and *Generalize* stages of training in TiZero and TiZero-RND. Interquartile mean values with standard deviations over 5 seeds are used, and elapsed time is in hours. N/A indicates that the value for the corresponding row and column is not available due to the method not reaching the respective training stage.

Method	Passed Curriculum Scenarios ↑	*Curriculum* Rollouts ↓	*Curriculum* Elapsed Time ↓	Passed *Challenge* Phases ↑	*Challenge* Rollouts ↓	*Challenge* Elapsed Time ↓	Passed *Generalize* Phases ↑	*Generalize* Rollouts ↓	*Generalize* Elapsed Time ↓
TiZero [12]	**10.00 ± 0.00**	275.30 ± 19.57	5.63 ± 0.38	**1.00 ± 0.00**	2940.33 ± 330.98	60.15 ± 6.19	N/A	N/A	N/A
TiZero-RND (Ours)	**10.00 ± 0.00**	**261.87 ± 50.58**	**5.36 ± 1.06**	0.5 ± 0.5	**2297.00 ± 486.19**	**47.02 ± 9.88**	**0.25 ± 0.43**	**978.0 ± 111.0**	**19.78 ± 2.24**

same game scenario and threshold for the *Generalize* phase, and the opponent is chosen from the pool of previous policies via the approach of AlphaStar [22]. After passing this stage, training goes back to the *Challenge* phase where it continues the alternation between *Generalize* and *Challenge* phases until manually stopped. Here, training can be stopped if convergence is observed in the performance (e.g. average win-rate) or, if existing, a maximum number of training steps is reached.

Heuristic Football Game-AI Baseline. As for the scripted game-AI comparison baseline, we have chosen the built-in, scripted football game-AI provided for the GRF environment in the framework presented by Song et al. [20]. This baseline is essentially a rule-based bot proposed by Konings [8], and it accepts a difficulty level which can be between 0 and 1, where a higher difficulty level corresponds to a faster decision-making and reaction time. Additionally, the non-active players are controlled by another rule-based bot which follows standard, basic football behavior patterns, such as moving towards the ball when not possessing it. This built-in AI can be replaced by another mechanism or disabled in GRF. Here, we utilize it as a baseline for our agents to play against.

Experiments and Baseline Evaluations. We run training experiments on the two methods that we study in this work: TiZero without any intrinsic/exploration reward (TiZero) and TiZero augmented with RND (TiZero-RND). Each model is trained for 170M environment steps, and the intrinsic/exploration reward term is added after 700 rollouts for network warm up [5], where a rollout refers to a fixed, limited number of transitions. This corresponds to 0.14×10^8 environment steps, as each rollout is performed by a batch of 40 workers, with each worker performing 500 environment steps. On average, a single training session runs for about a week.

Furthermore, we evaluate the generalization of the best trained models from TiZero and TiZero augmented with RND by having them play a number of football games against a HAI baseline provided by GRF and described in Section 3.2. It should be noted that the agents optimize against this baseline HAI only in the first curriculum scenario and spend most of the training in playing against versions of themselves [12]. Therefore, we are effectively evaluating the generalization of the methods under study. For these evaluations, the football scenario used is the same as the one for *Challenge* and *Generalize* phases, i.e., a normal football match scenario. We report gameplay performance metrics for each method averaged over 50 evaluation matches, repeating the experiment with 5 different seeds. We use the HAI baseline with the medium strength level.

Lastly, we perform an ablation study of the effect of the fixed positional encoding for player ID embedding compared to TiZero's approach. In the original design, one-hot encoding is used for a given player ID that is then input to a fully-connected layer, outputting a number of latent values equal to the length of the original one-hot encoding [12]. As we replace this part of the architecture with a fixed positional encoding, we perform training experiments on TiZero-RND using the original MLP-based encoding for the player IDs and compare the results to the ones for TiZero-RND with positional encoding used. We thereby analyze the impact of using the fixed positional encoding approach on training sample efficiency and stability compared to the original MLP encoding.

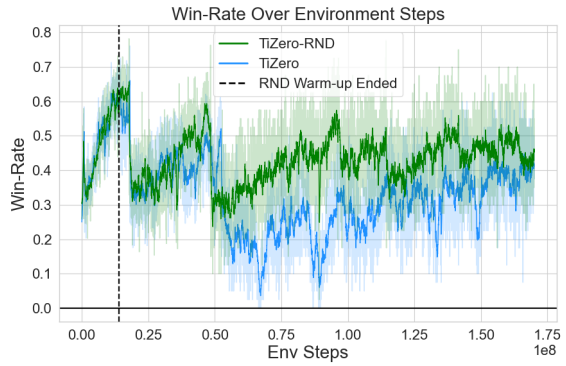
Appendix C presents a comprehensive list of the hyperparameters used in our experiments. In addition, Appendix A reports supplementary experiments that use an alternative exploration strategy based on a different form of intrinsic reward. Specifically, we replace the RND module with the Self-supervised Intrinsic Reward (SSIR) [5], and we refer the reader to the appendix for a detailed description of this methodology and its corresponding results.

5.2 Results

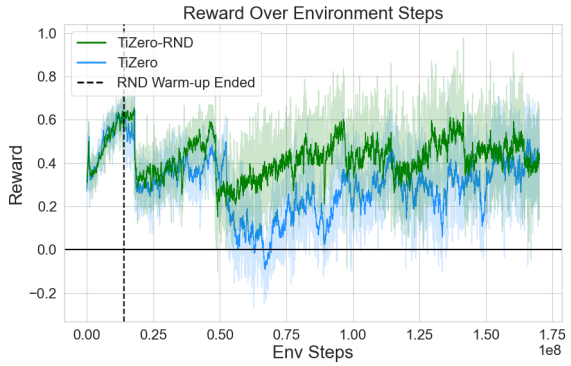
We first discuss the sample efficiency of the methods during training and then evaluate the gameplay generalization of the best models of each method against a scripted heuristic AI as a baseline (see Section 5.1). Furthermore, we analyze the effect of using the fixed positional encoding for player ID embedding compared to TiZero's original approach via an ablation study.

Training Sample Efficiency. We train each of the methods as described in Section 5.1. We summarize the results with respect to sample-efficiency metrics in Table 1, namely the number of training phases passed, the elapsed time and the number of rollouts required, on average, to pass a particular phase. We have derived these metrics for the curriculum, *Challenge*, and *Generalize* stages. It should be noted that progress in the training process is mainly implied by the number of training phases passed, as the goal of the agents is to achieve given win-rate thresholds at each phase to proceed to the next.

From Table 1, we see that both TiZero and TiZero-RND pass all of the curriculum scenarios. Both exhibit a similar amount of elapsed time and number of rollouts in the curriculum phase as well, while TiZero-RND appears to require slightly fewer rollouts here. More specifically, TiZero-RND achieves a 4.8% improvement considering the average elapsed time required for each method to pass a curriculum scenario. In the *Challenge* phase, TiZero passes more stages on average; however, TiZero-RND demands a lower number of rollouts and elapsed time to go through the *Challenge* stages. Similarly to the curriculum phases, in these stages TiZero-RND achieves an improvement rate of 21.8% when comparing each method's average elapsed time per *Challenge* stage.



(a) Win-Rate Comparison During Training



(b) Reward Comparison During Training

Figure 3: Comparison of win-rate and reward achieved by the best-performing TiZero-RND (green) and the best-performing standard TiZero (blue) experiments during training. The solid points represent the exponential moving average, and the shaded points are the raw values.

Furthermore, we observe that standard TiZero does not complete any stages in the *Generalize* phase, whereas TiZero-RND does. Additionally, TiZero-RND attains its number of *Generalize* stages with a number of rollouts and elapsed time which is lower than the ones needed in the *Challenge* phase for both standard TiZero and TiZero-RND. Considering that the *Generalize* phase is a difficult phase where the models are faced with versions of themselves against whom they had previously demonstrated worse performance, we believe TiZero-RND has better utilized its training data within our fixed training budget to allow for a better generalization. We mainly attribute this better utilization of training data to its more efficient exploration of the game states via the added RND term during training.

In conclusion, TiZero-RND improves sample efficiency per training stage based on the number of rollouts and elapsed time it requires to pass through a single stage, specifically by 13.3% considering both curriculum and Challenge phases, and it also improves generalization over long training experiments compared to the original TiZero, as it manages to succeed and adapt better against opponents that previously yielded poor performance on average.

Lastly, Fig. 3 shows comparison plots of the win-rate and reward values achieved by the best-performing TiZero-RND and the best-performing TiZero experiments (among their respective 5 seeds). In the figure, both methods exhibit a similar trend throughout training with peaks followed by sudden drops, indicating a curriculum or stage change. This is while the trajectory within each given training phase (i.e., after drops) remains upward. Nevertheless, TiZero-RND appears to achieve a higher win-rate and reward values throughout most of the training session. Additionally, while both methods experience drops especially when the training stage changes (i.e., often due to increased difficulty), TiZero-RND's drops seem to be less pronounced compared to the ones for standard TiZero. For instance, between the 50 and 100 million environment steps marks, the win-rate for standard TiZero drops near zero at times, while TiZero-RND's value remains stable between 0.3 and 0.6. We believe these results indicate a steadier learning performance for TiZero-RND, possibly due to its enhanced exploration.

Generalization vs. Baseline Heuristic AI. To further evaluate the gameplay performance and generalization to less encountered opponents, we run 50 games against the baseline HAI (see Section 5.1) for the top-performing models of the original TiZero and TiZero-RND. We report football-specific gameplay metrics in Table 2 to assess the overall behavioral quality of the models, including the percentage of good and bad passes and shots, total possession, and interception counts. Specifically, according to GRF, a good pass is one that successfully reaches a teammate, while a bad pass fails to do so. Similarly, a good shot results in a goal, whereas a bad shot is one towards the goal which does not score.

In passing, both methods attempt a comparable number of total passes, while TiZero achieves a higher percentage of good passes. However, a significant divergence is observed in the shooting behaviors; while TiZero-RND attempts much fewer total shots on average, it demonstrates a higher accuracy and precision as it attains a higher percentage of good shots and a lower rate of bad shots. Additionally, in possession, both methods showcase a comparable performance, with standard TiZero achieving slightly more possession on average. Defensively, TiZero-RND proves more effective at intercepting the ball, whereas standard TiZero is better at retaining possession without getting intercepted.

Notably, TiZero-RND showcases significantly lower variance across most metrics; for instance, its standard deviation for the number of shots and possession is roughly five to six times lower than that of standard TiZero. Therefore, we observe that TiZero-RND showcases a generalized gameplay similar to standard TiZero while reaching a higher accuracy in shooting. Here, higher shooting accuracy indicates more chances to score a goal. It further appears to have converged to a more certain and stable behavior as demonstrated by the lower standard deviation across reported metrics.

Effect of Fixed Positional Encoding on Player ID Embedding. Here, we ablate the original one-hot MLP encoding for the player ID against our approach of using a fixed positional encoding instead. We therefore compare TiZero augmented with RND, which showed better sample efficiency in our experiments above, in two versions: one using the original MLP encoding for player IDs (i.e., ablated),

Table 2: Gameplay metrics against a baseline heuristic AI (HAI), which is less-seen during training, for TiZero and TiZero-RND. Interquartile means and standard deviations are computed over 50 games per best model, aggregated over 5 seeds, and the baseline is used on "medium" difficulty. Good and bad passes and shots are calculated as percentages of the total metric value.

Method	Total Passes	Good Pass % ↑	Bad Pass % ↓	Total Shots	Good Shot % ↑	Bad Shot % ↓	Total Possession	Interception ↑	Get Intercepted ↓
TiZero [12]	**31.32 ± 21.82**	**65.85**	**33.30**	**0.66 ± 0.68**	5.42	92.77	**646.1 ± 218.3**	13.2 ± 1.2	**12.2 ± 2.7**
TiZero-RND (Ours)	28.66 ± 6.35	60.61	40.09	0.11 ± 0.12	**18.18**	**72.73**	619.7 ± 37.2	**13.6 ± 0.2**	13.0 ± 0.8

Table 3: Ablating the fixed positional encoding for player ID embedding and using the original MLP approach. Interquartile mean values with standard deviations over 5 seeds are used, and elapsed time is in hours.

Method	Passed Curriculum Scenarios ↑	*Curriculum* Rollouts ↓	*Curriculum* Elapsed Time ↓	Passed *Challenge* Phases ↑	*Challenge* Rollouts ↓	*Challenge* Elapsed Time ↓	Passed *Generalize* Phases ↑	*Generalize* Rollouts ↓	*Generalize* Elapsed Time ↓
TiZero-RND (w/o PE)	**10.00 ± 0.00**	352.90 ± 76.15	7.29 ± 1.59	0.40 ± 0.49	**1739.50 ± 69.75**	**36.21 ± 1.26**	**0.40 ± 0.49**	2568.00 ± 566.00	53.19 ± 12.00
TiZero-RND (w/ PE, Ours)	**10.00 ± 0.00**	**261.87 ± 50.58**	**5.36 ± 1.06**	**0.5 ± 0.5**	2297.00 ± 486.19	47.02 ± 9.88	0.25 ± 0.43	**978.0 ± 111.0**	**19.78 ± 2.24**

and another using fixed positional encoding (i.e., as described in Section 4.2). To compare, we run training experiments using these two versions in a similar fashion to our previous experiments for TiZero and TiZero-RND in Table 1.

The sample-efficiency metrics for our training experiments using the ablated and unmodified versions of TiZero-RND are summarized in Table 3. Examining the table, it can be seen that both versions pass all of the curriculum scenarios in their experiments. However, TiZero-RND with positional encoding appears to require slightly less time and number of rollouts to pass the curriculum part of training compared to the ablated version. Regarding the Challenge phases, although the positional-encoding variant passes slightly more phases, the ablated TiZero-RND requires fewer rollouts and time to attain the Challenge phases. On the other hand, this relation is reversed for the Generalize phase; the ablated version goes through more Generalize phases, while TiZero-RND with positional encoding demands substantially fewer rollouts and shorter elapsed time to complete its phases.

Overall, aggregating the average time per phase across all stages in Table 3, we can see that the ablated TiZero-RND requires on average ~ 108 hours in total to reach a fixed number of environment steps (170M as described in Section 5.1), whereas the positional-encoding version takes ~ 82 hours. Additionally, considering the number of phases passed, neither method clearly outperforms the other, as they both demonstrate comparable progress throughout the training stages. Furthermore, by utilizing a fixed positional encoding, we reduce the total number of learnable parameters in the non-ablated version. Consequently, based on this architectural efficiency, together with the significantly lower wall-clock time to achieve similar results, we prefer TiZero-RND with positional encoding in our ablation study.

6 Limitations

In this section, we discuss some limitations of our work. We do not do an exact one-to-one comparison with TiZero as presented by Lin et al. [12]. While the method is outlined in significant detail in the original paper, an implementation for the method and its training scheme is not provided at the time of writing, and so, we had to make assumptions on many smaller details in our re-implementation. Additionally, we had to modify the method to bring its computational costs within our time and resource budget.

Furthermore, we do not train the model for 40 days as is done for the original TiZero, and so, we can not guarantee that the results presented here would necessarily extend to such a prolonged training period. However, such lengthy training sessions are anyway not feasible in conventional game-AI development. We also do not carry out an extensive hyperparameter analysis; instead, we chose hyperparameters based on preliminary experiments. We consider this sufficient for the purposes of this paper.

Lastly, our approach can be applied to other similar MARL frameworks, but the anticipated sample-efficiency improvements in domains other than football-like environments are not guaranteed and require further validation and experiments. We view this as a promising direction for future work.

7 Conclusion

In this work, we have targeted the need for better sample efficiency in multi-agent reinforcement learning methods within a simulated football context. We let TiZero represent a state-of-the-art example of such, which requires excessively long training times to achieve good gameplay performance. To address this, we first modified the architecture of TiZero for computational efficiency. Then, we proposed an augmented version of it by adding a multi-agent random network distillation exploration term to the original reward function. This was motivated by our initial hypothesis that poor sample efficiency can be remedied by better exploration mechanisms. We then carried out training experiments to compare sample efficiency. We saw that TiZero with random network distillation improves sample efficiency by an average of 13.3% in a multi-stage training setup and enhances adaptability to challenging scenarios compared to original TiZero. Lastly, we carried out gameplay evaluations of the learned policies against a heuristic AI baseline, less-seen in training, to test generalization. Our evaluations showed that the random network distillation version achieves a higher accuracy in shooting compared to standard TiZero, as well as higher stability and certainty in its learned behavior.

References

[1] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. 2024. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

[2] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. In *International Conference on Learning Representations*.

[3] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. 2004. Intrinsically Motivated Reinforcement Learning. *Advances in Neural Information Processing Systems* (2004).

[4] Wojciech M Czarnecki et al. 2020. Real World Games Look Like Spinning Tops. *Advances in Neural Information Processing Systems* 33 (2020).

[5] Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. 2022. Exploration-guided reward shaping for reinforcement learning under sparse rewards. *Advances in Neural Information Processing Systems* 35 (2022).

[6] Joris Dinneweth, Abderrahmane Boubezoul, René Mandiau, and Stéphane Espié. 2022. Multi-agent reinforcement learning for autonomous vehicles: a survey. *Autonomous Intelligent Systems* 2, 1 (16 Nov 2022).

[7] Mikhail Jacob, Sam Devlin, and Katja Hofmann. 2020. "it's unwieldy and it takes a lot of time"—challenges and opportunities for creating agents in commercial games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 16. 88–94.

[8] Bastiaan Konings. 2024. BazkieBumpercar/GameplayFootball. https://github.com/BazkieBumpercar/GameplayFootball original-date: 2017-11-02T15:29:41Z.

[9] Aleksandar Krnjaic et al. 2024. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

[10] Karol Kurach et al. 2020. Google Research Football: A Novel Reinforcement Learning Environment. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (2020). doi:10.1609/aaai.v34i04.5878

[11] Raul Lara-Cabrera, M. Nogueira-Collazo, Carlos Cotta, and Antonio Fernández-Leiva. 2015. Game artificial intelligence: Challenges for the scientific community. *CEUR Workshop Proceedings* 1394 (2015).

[12] Fanqi Lin, Shiyu Huang, Tim Pearce, Wenze Chen, and Wei-Wei Tu. 2023. TiZero: Mastering Multi-Agent Football with Curriculum Learning and Self-Play. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*.

[13] Michael L Littman. [n. d.]. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*.

[14] Siqi Liu et al. 2022. From motor control to team play in simulated humanoid football. *Science Robotics* 7, 69 (2022). doi:10.1126/scirobotics.abo0235

[15] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*.

[16] OpenAI et al. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. arXiv:1912.06680 [cs.LG]

[17] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations*.

[18] Alessandro Sestini, Joakim Bergdahl, Jean-Philippe Barrette-LaPierre, Florian Fuchs, Brady Chen, Micheal Jones, and Linus Gisslén. 2025. Human-Like Goalkeeping in a Realistic Football Simulation: a Sample-Efficient Reinforcement Learning Approach. *arXiv preprint arXiv:2510.23216* (2025).

[19] Alessandro Sestini, Linus Gisslén, Joakim Bergdahl, Konrad Tollmar, and Andrew David Bagdanov. 2022. Automated gameplay testing and validation with curiosity-conditioned proximal trajectories. *IEEE Transactions on Games* 16, 1 (2022), 113–126.

[20] Yan Song, He Jiang, Haifeng Zhang, Zheng Tian, Weinan Zhang, and Jun Wang. 2024. Boosting Studies of Multi-Agent Reinforcement Learning on Google Research Football Environment: The Past, Present, and Future. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 10 pages.

[21] Ashish Vaswani et al. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.

[22] Oriol Vinyals et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019).

[23] Chen Wang et al. 2024. AI-Olympics: Exploring the Generalization of Agents through Open Competitions. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*. doi:10.24963/ijcai.2024/1040

[24] Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. 2022. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature* 602, 7896 (2022), 223–228.

[25] Linjie Xu, Zichuan Liu, Alexander Dockhorn, Diego Perez-Liebana, Jinyu Wang, Lei Song, and Jiang Bian. 2024. Higher Replay Ratio Empowers Sample-Efficient Multi-Agent Reinforcement Learning. In *2024 IEEE Conference on Games (CoG)*.

[26] Zelai Xu, Yancheng Liang, Chao Yu, Yu Wang, and Yi Wu. 2023. Fictitious Cross-Play: Learning Global Nash Equilibrium in Mixed Cooperative-Competitive Games. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*.

[27] Chao Yu et al. 2024. The surprising effectiveness of PPO in cooperative multi-agent games. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

[28] Chen Zhang, Qiang He, Zhou Yuan, Elvis S Liu, Hong Wang, Jian Zhao, and Yang Wang. 2024. Advancing DRL agents in commercial fighting games: Training, integration, and agent-human alignment. *arXiv preprint arXiv:2406.01103* (2024).

[29] Chen Zhang, Huan Hu, Yuan Zhou, Qiyang Cao, Ruochen Liu, Wenya Wei, and Elvis S Liu. 2024. Training Interactive Agent in Large FPS Game Map with Rule-enhanced Reinforcement Learning. In *2024 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[30] Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas Schön, and Per Mattsson. 2025. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. *Advances in Neural Information Processing Systems* (2025).

Table 4: Sample-efficiency metrics for TiZero-SSIR under the curriculum, *Challenge*, and *Generalize* stages of training, besides the same resulting metrics for TiZero and TiZero-RND for comparison. Interquartile mean values with standard deviations over 5 seeds are used, and elapsed time is in hours. N/A indicates that the value for the corresponding row and column is not available due to the method not reaching the respective training stage.

Method	Passed Curriculum Scenarios ↑	*Curriculum* Rollouts ↓	*Curriculum* Elapsed Time ↓	Passed *Challenge* Phases ↑	*Challenge* Rollouts ↓	*Challenge* Elapsed Time ↓	Passed *Generalize* Phases ↑	*Generalize* Rollouts ↓	*Generalize* Elapsed Time ↓
TiZero-SSIR (Ours)	8.00 ± 0.00	**91.30 ± 3.91**	**1.80 ± 0.05**	N/A	N/A	N/A	N/A	N/A	N/A
TiZero [12]	**10.00 ± 0.00**	275.30 ± 19.57	5.63 ± 0.38	**1.00 ± 0.00**	2940.33 ± 330.98	60.15 ± 6.19	N/A	N/A	N/A
TiZero-RND (Ours)	**10.00 ± 0.00**	261.87 ± 50.58	5.36 ± 1.06	0.5 ± 0.5	**2297.00 ± 486.19**	**47.02 ± 9.88**	**0.25 ± 0.43**	**978.0 ± 111.0**	**19.78 ± 2.24**

Table 5: Gameplay metrics against a baseline heuristic AI (HAI), which is less-seen during training, for TiZero-SSIR, TiZero, and TiZero-RND. Interquartile means and standard deviations are computed over 50 games per best model, aggregated over 5 seeds, and the baseline is used on "medium" difficulty. Good and bad passes and shots are calculated as percentages of the total metric value.

Method	Total Passes	Good Pass % ↑	Bad Pass % ↓	Total Shots	Good Shot % ↑	Bad Shot % ↓	Total Possession	Interception ↑	Get Intercepted ↓
TiZero-SSIR (Ours)	5.48 ± 4.47	38.50 %	61.50 %	0.20 ± 0.34	0.00 %	100.00 %	**1008.9 ± 248.4**	13.7 ± 0.6	**7.7 ± 2.1**
TiZero [12]	**31.32 ± 21.82**	**65.84 %**	**34.16 %**	**0.66 ± 0.68**	5.45 %	94.55 %	646.1 ± 218.3	13.2 ± 1.2	12.2 ± 2.7
TiZero-RND (Ours)	28.08 ± 5.69	59.86 %	40.14 %	0.11 ± 0.12	**22.72 %**	**77.28 %**	614.2 ± 42.2	**13.9 ± 0.2**	13.3 ± 1.2

A Additional Methodology: A Self-Supervised Intrinsic Reward as an Exploration Mechanism

Here, we present an additional methodology on a different approach we also investigate, with the same goal of improving sample efficiency through a varied and hopefully better exploration mechanism. More specifically, instead of using random network distillation and its added term, we look into self-supervised reward shaping and its application as an exploration mechanism here. As a more recent and dynamic concept, self-supervised reward shaping utilizes the agent's interactions with the environment and the original reward as self-supervisory signals to learn an additional, denser reward. A notable work is the exploration-guided reward shaping framework, which integrates learned intrinsic rewards with exploration bonuses to improve learning [5]. Additionally, the authors provide a derivation and formulation for their learned intrinsic reward, from which we take inspiration here for a SSIR in our multi-agent framework.

A.1 Methodology of SSIR

Aiming to improve the exploration signal, we augment TiZero's original reward (see Section 3.2) with an SSIR term, inspired by the work of Devidze et al. [5]. For our implementation, we use a 2-layer MLP with ReLU activations for the SSIR network. The input to this network has the same shape as for our actor (see Fig. 2), and the output has the same shape as the number of actions, but with tanh activations. We utilize the network output corresponding to each agent's taken action as the agent's SSIR term for that transition [5].

While Devidze et al. [5] present a single-agent formulation of this intrinsic reward, we first calculate the values for each agent as described above, and then, we average the intrinsic reward of the individual agents and set that as our global SSIR, adhering to the CTDE paradigm (see Section 2.2). Our resulting SSIR-based reward function is:

$$R_{\text{SSIR}}(t) = R_{\text{TiZero}}(s_t, a_t, s_{t+1}) + \alpha \frac{1}{N} \sum_{i=1}^{N} r_\phi(o_t^i, a_t^i), \tag{7}$$

where $R_{\text{TiZero}}(s_t, a_t, s_{t+1})$ is the original TiZero reward function [12], α is a hyper-parameter, N is the number of agents, and $r_\phi(o_t^i, a_t^i)$ is the SSIR term computed for each agent by a neural network with parameters ϕ. To optimize r_ϕ, we use the empirical update rule for ϕ from Devidze et al. [5] and apply gradient descent by considering the simplified update term as a loss function.

A.2 SSIR in Training and Evaluation Against Baseline

In a similar fashion to Section 5, we run training experiments using TiZero with the provided SSIR formulation, and we analyze the results besides those provided for TiZero and TiZero-RND to study the effects of the self-supervised intrinsic reward on training sample efficiency. We also evaluate the best policies trained by TiZero-SSIR against our HAI comparison baseline using the same setup and process used for evaluating the policies trained by TiZero and TiZero-RND in Section 5.

The training experiment setup is the same as the one provided in Section 5.1, and we aggregate the results for TiZero-SSIR (i.e., TiZero with the self-supervised intrinsic reward) from 5 seeds. The training results for this new method, besides those for TiZero and TiZero-RND, are provided in Table 4. Looking at the training results, we see that TiZero-SSIR passes slightly fewer curriculum scenarios than TiZero and TiZero-RND on average. Additionally, it requires fewer rollouts to complete each curriculum scenario it does pass compared to the other two methods. However, we also observe that this method does not manage to pass any Challenge or Generalize stages, meaning it gets stuck in the curriculum phase. It potentially overfits to simple strategies such as going into the goal with the ball instead of passing and shooting which is more

effective in later scenarios. Therefore, considering the results of TiZero-SSIR, we can see that while certain forms of reward shaping, such as RND, improve sample efficiency, others may not bring such enhancements, although they may aim to provide a form of additional exploration.

Furthermore, the gameplay evaluation results for TiZero-SSIR against the baseline, averaged over 50 games and repeated for 5 seeds, are shown in Table 5. The results for the evaluation of TiZero and TiZero-RND are also brought alongside the new metrics for ease of comparison.

Considering the gameplay evaluation metrics in Table 5, the models learned by TiZero-SSIR show a more possessive strategy with the highest total possession among all and getting intercepted almost half the time compared to the other methods. Additionally, these models carry out the fewest number of passes, and the majority of these passes are inaccurate. Regarding shots, the models produced by TiZero-SSIR do not manage to execute any good shots, and thus, do not score any goals. These together indicate that the TiZero-SSIR approach seems to often reach an extreme policy with regards to possession, which is aligned with the training experiment results provided above; the method does not pass the more difficult curriculum scenarios that may require a strategy generalizing better to more complex scenarios.

B TiZero's Actor and Critic Observation Vectors

Here, we detail the elements of the observation vectors used by the actor and critic in our implementation of TiZero [12]:

- Controlled player's ID (*only actor*), represented by 1 value;
- Controlled player's information (*only actor*). This part includes sticky actions, current player position, current direction, tired factor, yellow card, red card, offside, relative ball position, distance to ball, relative teammate positions, distance to teammates, relative opponent positions, and distance to opponent, for a total of 87 values;
- Ball information (*only actor*). This part includes the ball position, ball direction, ball-owning team, ball rotation, ball-owning player, current player information, ball-owning player position, ball-owning player direction, relative position of the ball owner, distance to the ball owner, and ball owner's information, for a total of 57 values.
- Teammates' information (*shared for actor and critic*). This part includes teammate positions, teammate directions, teammate tired factors, teammate yellow cards, teammate red cards, and teammate offsides, for a total of 88 values.
- Opponents' information (*shared for actor and critic*). This part includes opponent positions, opponent directions, opponent tired factors, opponent yellow cards, opponent red cards, and opponent offsides, for a total of 88 values.
- Current match information (*shared for actor and critic*). This part includes the game mode, goal differences, and remaining environment steps, for a total of 9 values.
- Ball information (*only critic*). This part includes the ball position, ball direction, ball rotation, and the ball-owning team, for a total of 12 values.
- Ball-owning player as a one-hot encoded player index (*only critic*), for a total of 23 values.

C Experimental Configurations and Hyperparameters

Here, we present the configurations and hyperparameters used for our training experiments. Tables 6, 7, and 8 provide the training configurations for TiZero-RND, TiZero, and TiZero-SSIR, respectively.

Attribute	Value
Observation Encoder (Actor & Critic)	
Input MLP Layers	2
Input MLP Hidden Units	64
Match Info MLP Hidden Units	9
Final FC Hidden Units	256
Actor Policy Network	
Main MLP Layers	4
Main MLP Hidden Units	256
Positional Encoding Dimension	256
Pre-Action MLP Layers	2
Pre-Action MLP Hidden Units	256
Activation Function	ReLU
Critic Network	
Final Encoder Layers	2
Final Encoder Hidden Units	256
Activation Function	ReLU
Random Network Distillation (RND)	
Predictor Network Layers	4
Predictor Network Hidden Units	128
Target Network Layers	2
Target Network Hidden Units	64
Output Embedding Size	4
Learning Rate	0.0005
Update Frequency (Rollouts)	1
Term Coefficient	0.0001
Start After (Rollouts)	700
Training Parameters	
Discount Factor (γ)	0.999
GAE Lambda (λ)	0.995
PPO Epochs	2
Num. Mini-batches	1
Clipping Parameter (ϵ)	0.2
Value Loss Coefficient	1.0
Entropy Coefficient	0.01
Use Orthogonal Initialization	True
Initialization Gain	0.01
Use Feature Normalization	True

Table 6: Configurations and hyperparameters for TiZero-RND's training experiments.

Observation Encoder (Actor & Critic)	
Input MLP Layers	2
Input MLP Hidden Units	64
Match Info MLP Hidden Units	9
Final FC Hidden Units	256
Actor Policy Network	
Main MLP Layers	4
Main MLP Hidden Units	256
Positional Encoding Dimension	256
Pre-Action MLP Layers	2
Pre-Action MLP Hidden Units	256
Activation Function	ReLU
Critic Network	
Final Encoder Layers	2
Final Encoder Hidden Units	256
Activation Function	ReLU
Training Parameters	
Discount Factor (γ)	0.999
GAE Lambda (λ)	0.995
PPO Epochs	2
Num. Mini-batches	1
Clipping Parameter (ϵ)	0.2
Value Loss Coefficient	1.0
Entropy Coefficient	0.01
Use Orthogonal Initialization	True
Initialization Gain	0.01
Use Feature Normalization	True

Table 7: Configurations and hyperparameters for TiZero's training experiments.

Attribute	**Value**
Observation Encoder (Actor & Critic)	
Input MLP Layers	2
Input MLP Hidden Units	64
Match Info MLP Hidden Units	9
Final FC Hidden Units	256
Actor Policy Network	
Main MLP Layers	4
Main MLP Hidden Units	256
Positional Encoding Dimension	256
Pre-Action MLP Layers	2
Pre-Action MLP Hidden Units	256
Activation Function	ReLU
Critic Network	
Final Encoder Layers	2
Final Encoder Hidden Units	256
Activation Function	ReLU
Intrinsic Reward Settings	
Intrinsic Reward Network Layers	2
Intrinsic Reward Network Hidden Units	128
Intrinsic Reward Network Output Dimension	4
Learning Rate	0.0005
Clipping Parameter	0.01
Training Parameters	
Discount Factor (γ)	0.999
GAE Lambda (λ)	0.995
PPO Epochs	2
Num. Mini-batches	1
Clipping Parameter (ϵ)	0.2
Value Loss Coefficient	1.0
Entropy Coefficient	0.01
Use Orthogonal Initialization	True
Initialization Gain	0.01
Use Feature Normalization	True

Table 8: Configurations and hyperparameters for TiZero-SSIR's training experiments.