P-HRL: An Adaptive and Flexible Prediction-based Hierarchical Reinforcement Learning for Robot Soccer

Anonymous submission

Abstract

Robot soccer as a complex mixed cooperative-competitive task presents many challenges to multi-agent reinforcement learning (MARL). Firstly, due to the sparse rewards and the difficulty of measuring each agent's contribution to reward, it is difficult for MARL to learn effective offensive tactics; Secondly, MARL needs to learn soccer tactics and robot motion control simultaneously, so it results in inefficiency in training, making it difficult to adapt to new opponents. Although MARL has made impressive achievements in many robot tasks, MARL has shown a limited ability to handle such complex and long-horizon tasks, especially where other adversary agents are present and rewards are sparse. This is because the exploration efficiency of MARL of learning tasks when rewards are complex and sparse is low. To address these challenges, we propose Prediction-based Hierarchical Reinforcement Learning(P-HRL), a new framework for robot soccer. P-HRL consists of a high-level controller for giving soccer tactics and a low-level controller for robot motion control. Experimental results demonstrate that P-HRL beats baseline MATD3 in robot soccer and achieves a better end-to-end performance. When learning online, P-HRL has a faster learning speed to play against new opponents.

Introduction

Robot soccer have been developed to facilitate research in the field of robot control and to help deploy multi-agent systems (MAS) from virtual environments to the real world(Antonioni et al. 2021). Several leagues on robot soccer have been successfully organized in recent years, such as RoboCup[??], IEEEVSS[??] and many research problems based on robot soccer have received extensive attention, e.g., team coordination[??], navigation[??].

Multi-agent reinforcement learning(MARL) have achieved outstanding success on cooperative problems such as video games(Vinyals et al. 2019; Zhang et al. 2020; Berner et al. 2019; Wang et al. 2020) and robot control[??]. MARL is a method that controls the actions of multiple agents based on environmental feedback to maximize the joint action return by continuously interacting with the environment. MARL has achieved state-of-the-art performance in many competitive video games (e.g., StarCraftII[??], Dota2[??]) and beaten human players. Thus, the development of a MARL-based approach in robot soccer has the potential to outperform other non-reinforcement learning approaches[??][??]. However, unlike video games, MARL needs to control real entities(i.e. robots) instead of virtual entities(i.e. units in video games) in robot soccer. Controlling entities in continuous action spaces rather than discrete action spaces in competitive games presents many new challenges for MARL. We present the following challenges for the MARL method commonly used in continuous action spaces (e.g., MADDPG(Lowe et al. 2017), MATD3(Ackermann et al. 2019).) in robot soccer.

First, the sparsity of rewards and the long-term credit assignment problem(Vezhnevets et al. 2017) [???] make it difficult for MARL to learn to collaborate. Scoring goals in soccer games is highly sparse (e.g., on average, only two or three goals are scored in a game). MARL learns actions by using rewards to estimate expected returns[??][??]. Sparse rewards can cause MARL to misestimate the expected return and lead MARL to take inappropriate actions to control the robot. Moreover, the scores does not explicitly show the contribution of each robot to the goal, which leads MARL to misestimate the impact of each robot's actions on the event of scoring a goal. This results in unpredictable variance between MARL's assignment of expected returns and the correct assignment. Previous work[??] addressed the problem of reward sparsity by reward shaping. But reward shaping does not solve the problem of long-term credit allocation in robot soccer. This is because the player's contribution to goals in soccer games needs to be context-dependent. Any inappropriate consideration of the context also causes the reward to misestimate the contribution of each player to the goal.

Second, high-dimensional and continuous state-action spaces make it difficult for MARL to adapt to new opponents. Unlike other robot control tasks in stationary environments[??][??], changes in the environment in robot soccer are caused not only by the actions of each robot controlled by MARL, but also by robots that are not controlled by MARL (i.e., the opponent's robots). This results in the optimal policy likewise changing as the opponent robot policy changes. This leads MARL to explore a larger state-action space to find the optimal policy. Previous work[??][??] in video games guides MARL to find optimal policies by means of training with opponents with domain expertise (e.g., human players). This approach requires training with a large number of opponents to fully explore the state-action space. However, in robot soccer, there is a higher dimensional state-action space and greater training costs, which make it difficult to use this approach in robot soccer.

Inspired by hierarchical reinforcement learning(HRL), this paper aims to design a *coach* to guide MARL for robot motion control at the level of soccer tactics and help MARL to adapt to new opponents. We propose Prediction-based Hierarchical Reinforcement Learning(P-HRL), which consists of two hierarchical parts: a coach for soccer tactics and a robot controller for robot motion control. Coach decomposes the soccer game into a series of subtasks, and the robot controller controls the motions of robots to complete these subtasks. For the first challenge, P-HRL decomposes the process of playing soccer into a number of ordered subtasks, which provide different goals and rewards for each robot. It compensates for the sparsity of rewards of goal scoring, and converts the long-term credit assignment process into a short-term, per-robot credit assignment process. For the second challenge, the hierarchical structure of P-HRL allows the coach to focus only on learning soccer tactics and the robot controller to focus only on learning robot control. This allows the coach to train in a low-dimensional space, and the robot controller only needs to train for the subtasks assigned by the coach.

We test the P-HRL by simulating 3x3 robot soccer games under IEEE VSS rules[??]. The results demonstrate that P-HRL has better performance than the state-of-the-art MARL methods. The main contributions of this paper can be summarized as follows:

(1) To the best of our knowledge, this is the first work to use the HRL framework in robot soccer to control robot soccer in a more soccer tactically targeted way than MARL.

(2) We design multiple key performance indicators(KPIs) for robot soccer such as ball possession rate. These KIPs are designed with reference to the relevant indicators of human soccer, which allow for a more professional and systematic evaluation of P-HRL performance in robot soccer.

(3) P-HRL has higher scoring, possession and passing numbers compared to the state-of-the-art baseline MATD3. And P-HRL's scores on goals rose faster and had better overall performance on several KPIs compared to MATD3 when playing against new opponents in online learning.

Background

Multi-agent Reinforcement Learning

Hierarchical Reinforcement Learning

Prediction-based Hierarchical Reinforcement Learning(P-HRL)

Figure 1 shows the temporal process of P-HRL. P-HRL is inspired by HRL[??][??][??][??] and the key idea of P-HRL is to design a coach to guide the robot controller at the level of soccer tactics. To achieve this idea, we design two types of subtasks: offensive subtask and defensive subtask. Coach breaks the soccer game process into a combination of these two subtasks, and the robot controller is aimed to complete those subtasks.

Problem Formalization

The entire process of robot soccer can be formalized as a partially observable, goal-conditioned Markov Decision Process (MDP)[??]. Specifically, it is formally defined as a tuple $< \mathcal{S}, \mathcal{O}, \mathcal{G}, \mathcal{U}, P, R, \gamma >$, where $\mathcal{S}$ is state space. $s_t \in \mathcal{S}$ is the state at each time step t. These states usually consist of position vectors and speed vectors for robots and balls. Limited by the robot hardware, robots cannot observe all the information in the state(e.g., the speed vectors of the opposing robots is usually not available), so each robot can only receive the partially observable observations $o_t^i \in \mathcal{O}$,where $i = 1, \ldots, n$, n is the number of our robots. We define $o_t = \bigcup_{i=1}^{n} o_t^i$ is all the observations that our robots obtain from the environment. $g_t \in \mathcal{G}$ is the subtask. $\mathbf{u} = \left(u_t^1, ..., u_t^n\right)$ is the joint action, where u_t^i is the action of the i-th robot. $P\left(s_{t+1} | s_t, \mathbf{u}_t\right)$ is the state transition function. $\mathcal{R}$ is the reward function, which consists of the subtask reward r_{g_t} and the external reward r_t. γ is the discount factor. A complete robot soccer game is defined as:

$$P\left(\tau\right) = P\left(s_0\right) \prod_{t=0}^{T} P\left(\mathbf{u}_t | s_t\right) P\left(s_{t+1} | s_t, \mathbf{u}_t\right) \qquad (1)$$

where $\tau = \{s_0, \mathbf{u}_0, s_1, \mathbf{u}_1, ..., s_T, \mathbf{u}_T\}$ is the trajectory of a game. s_0 is the initial state. It is the state at the beginning of the game. s_T is the terminal state and defined as the state when a goal is scored or the state when the game reaches the end time step $t = T_{max}$. At each time step t, the coach starts (or continues) a subtask g_t according to the coach policy $\pi^c\left(o_t\right)$ and gives a subtask reward r_{g_t}. The robot controller gives an joint action $\mathbf{u}_t \sim \pi^r\left(\mathbf{u}_t \mid o_t^i, g_t, r_{g_t}, r_t\right)$ to control the robot movement. Then, the environment transfers to state s_{t+1} according to $P\left(s_{t+1} | s_t, \mathbf{u}_t\right)$. The aim of the robot soccer is to maximize goal scores, which is defined by maximizing discounted accumulated rewards $E\left[\sum_{t=0}^{T} \gamma r_t\left(s_t, \mathbf{u}_t\right)\right]$.

Coach

Method Overview Formally, coach is represented as a tuple $\langle \mathcal{G}, r_g, I_g, \beta_g \rangle$, where $g \in \mathcal{G}$ is subtask. r_g is subtask reward, and it is used to train the agent in robot controller to complete the subtask g. I_g and β_g is the initiation condition and the termination condition of the subtask g, respectively. They are used to start or terminate a subtask g.

We divide subtasks into two types $\mathcal{G} = \left\{g^A, g^D\right\}$, where g^A is offensive subtask and g^D is defensive subtask. Considering the most general soccer tactics, when our robots are in possession, the optimal tactic is obviously to attack and shoot. When opposing robots are in possession, the optimal tactic is obviously to defend to prevent the opponent from scoring. Thus, we define the offensive subtask as the subtask that helps to score when our robots are in possession and defensive subtasks as the subtask that help reduce scores lost when opposing robots are in possession.

In offensive subtask, passing the ball is an effective way to secure possession to ensure that the opponent team does not have the opportunity to attack. Thus, coach encourage passing between robots to attack with more flexibility. To

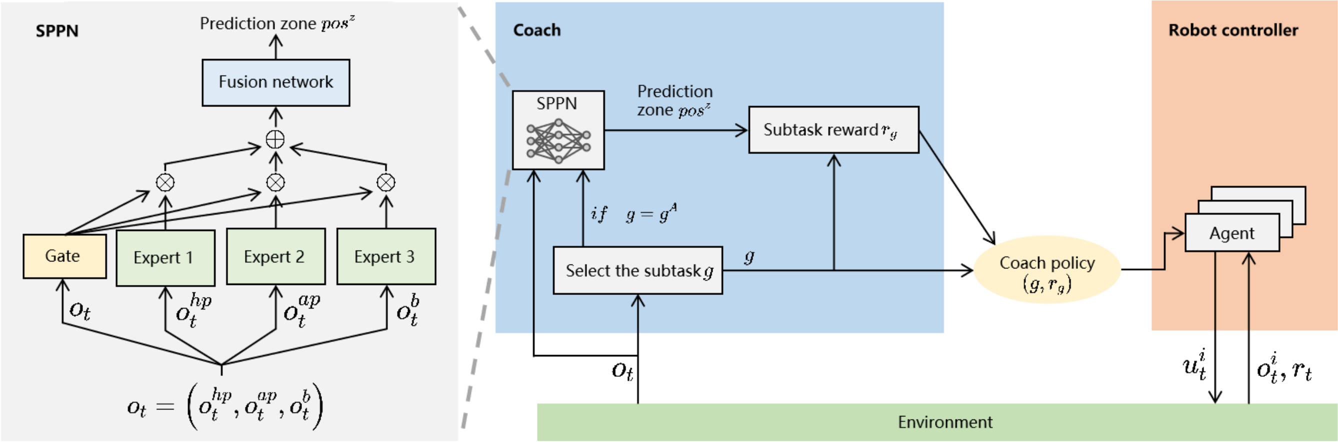


Figure 1: Illustration of the P-HRL. At each time step t, after coach receives observation o_t, it starts (or continues) a subtask g and calculates the subtask reward r_g according to subtask initiation condition I_g and termination condition β_g. Then, each agent in the robot controller will output an action u_t^i to control the corresponding robot based on coach policy π^c, observation o_t^i and reward r_t.

realize the this idea, we propose the soccer position prediction network (SPPN) which is described in following session. SPPN will predict the zone where the ball will come up after a fixed number t_g of time steps. Our robot closest to the ball is defined as an attacker, and the rest of our robots are defined as supporters. Attacker's goal is to attack with the ball, and supporter's goal is to go to prediction zone and wait to be passed. It is worth noting that in this process, once the ball is intercepted by the opposing robot (i.e. the opposing robots are in possession), the offensive subtask will be terminated immediately, and the defensive subtask will be started. In defensive subtask, because the opposing robots are in possession, we encourage all of our robots to intercept the ball. According to the process of the these subtasks, subtask reward r_g is defined as encouraging the robot to go to the target position (i.e. the position of the ball or the position predicted by the SPPN):

$$r_g = \frac{pos^r - pos^g}{\left\|pos^r - pos^g\right\|_2} \cdot v^r$$
$$where \quad pos^g = \begin{cases} pos^b & \text{if } g = g^A \text{ and robot is attacker.} \\ pos^z & \text{if } g = g^A \text{ and robot is supporter.} \\ pos^b & \text{if } g = g^D \end{cases} \tag{2}$$

where, pos^r, pos^z and pos^b is the position vectors of the robot, ball and centroid of the prediction zone respectively; v^r is the robot speed vector. Table 1 shows the definition of the initiation condition I_g and the termination condition β_g. Algorithm 1 describes the process of coach.

Soccer Position Prediction Network To achieve the football position prediction we mentioned in the previous section, we propose SPPN. SPPN is inspired by MoE[??] and is a neural network for prediction.

As mentioned in previous session, the robot soccer presents many challenges to soccer position prediction. Firstly, the data in observation is heterogeneous. The data in the observation can be divided into three types: data related to the home team player (i.e. our robots), data related to the away team player (i.e. the opposing robots), and data related to the ball. The correlation between the first two types of data is weak, but the correlation with the third type of data is strong. Secondly, because the opponents are not fixed, the network needs to have a fast convergence rate.

To address these challenges, we propose SPPN. Figure **??** shows the pipeline of SPPN. SPPN consists of the following three components:

- **Expert network:** To effectively solve the problem that data is heterogeneous, we use three expert networks in SPPN. Each expert focused on modeling only one type of data. We divided the data into the three types and input the divided data into three expert networks.
- **Gating network:** Gating network is used to gate the output of expert networks. This network structure can model heterogeneous data more efficiently.
- **Fusion network:** Fusion network is used to map the output features of the expert network with the soccer position prediction results. To speed up the convergence rate of SPPN, we divide the entire football field into several rectangular zones of equal size, and use SPPN to predict the zones rather than specific positions.

Given a observation $o^H = \left(o^{hp}, o^{ap}, o^b\right)$ obtained by a high-level controller, where o^{hp} is the part of the observation related to the home team players, o^{ap} is the part of the observation related to the away team players, and o^b is the part of the observation related to the ball, the soccer position prediction result of SPPN can be formulated as below:

$$\begin{gathered} z = h\left(\sum_{i=1}^{3} g\left(o^H\right) f_i\left(o^i\right)\right) \\ where \quad o^1 = o^{hp}, \quad o^2 = o^{ap}, \quad o^3 = o^b, \\ g\left(o^H\right) = softmax\left(f\left(o^i\right)\right) \end{gathered} \tag{3}$$

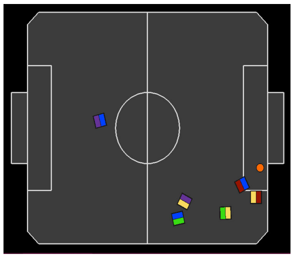


Figure 2: rSoccer VSSMA Environment

where, $f_i(\cdot)$ is expert network, $g(\cdot)$ is gating network, and it maintains the weight matrix of the expert networks through a fully connected layer $f(\cdot)$, $h(\cdot)$ is fusion network.

Robot Controller

In robot controller, we use MADDPG and the framework of centralized training and decentralized execution to implement robot control. Specifically, each agent controls the motion of one robot separately. Consider a game with N robots per team, where each agent i uses an actor network μ_i with parameter θ^{μ_i} and a critic network Q_i with parameter θ^{Q_i}. The aim of the robot controller is to complete subtask g and maximize goal scores. Thus, critic network Q_i is written as:

$$Q_i\left(o_t, \mathbf{u}_t\right) = \max\left(E\left[\sum_{j=0}^{T} \gamma\left(r^i_{g_{t+j}} + r_t\right) \mid g_t, o_t, \mathbf{u}_t\right]\right) \tag{4}$$

We update the parameter θ^{Q_i} of the critic network Q_i by using the tuple $(o, g, r_g, \mathbf{u}, r, o')$ in the replay buffer $\mathcal{D}$ to minimize the loss:

$$\begin{aligned} \mathcal{L}\left(\theta^{Q_i}\right) &= E_{o,g,r_g,\mathbf{u},r,o' \sim \mathcal{D}}\left[\left(Q_i\left(o, \mathbf{u}, g \mid \theta^{Q_i}\right) - y\right)^2\right], \\ where \quad y &= r + r^i_g + \gamma Q'_i\left(o', \mathbf{u}', g \mid \theta^{Q'_i}\right)\big|_{u'_j = \mu'_j(o^j)} \end{aligned} \tag{5}$$

Where $Q'_i\left(o', \mathbf{u}', g \mid \theta^{Q'_i}\right)$ is the target critic network with parameter $\theta^{Q'_i}$, which is periodically updated by parameter θ^{Q_i}. Each action u'_j in the joint action $\mathbf{u}'$ is given by the target actor network $\mu'_j\left(o^j\right)$, respectively, and the target actor network with parameter $\theta^{\mu'_i}$ is updated periodically by the parameter θ^{μ_i}. Using this critic network Q_i, the gradient of actor network μ_i used to optimize the deterministic policy of agent i can be written as:

$$\begin{aligned} \nabla_{\theta^{\mu_i}} J\left(\theta^{\mu_i}\right) = \\ E_{o,u^{j \neq i},g,r_g \sim \mathcal{D}}[\nabla_{u^i} Q_i\left(o, \mathbf{u}, g \mid \theta^{Q_i}\right)\big|_{u^i = \mu_i(o^i, g)} \\ \nabla_{\theta^{\mu_i}} \mu_i\left(o^i, g \mid \theta^{\mu_i}\right)] \end{aligned} \tag{6}$$

Training Method

Evaluation

Our experiments are based on rSoccer(Martins et al. 2022) - a framework for studying reinforcement learning for small and very small robot soccer. Specifically, we use the the IEEE VSSS (Very Small Size Soccer) multi-agent environment to simulate a robot soccer scenario (See Figure 2). IEEE VSSS is a robotic soccer competition in which two teams of three robots compete against each other . According to the rules, the robots are controlled remotely by computers without human intervention; The robot are fixed-size(0.08m*0.08m) with wheels but no hardware to dribbling or kicking the ball. The size of the soccer field is 1.5m*1.3m.

In our experiments, we basically followed the rSoccer (Bassani et al. 2020) in reward shaping. For better application to our experiments, we changed the sub-reward for movement toward the ball to movement toward the target, and added a speed penalty to encourage continuous movement of the robot. The detailed reward settings are defined as follows.

R_g is the goal scoring rewards and is defined as:

$$R_g = \begin{cases} 1 & \text{if the team scores a goal.} \\ 0 & \text{if no goal scored.} \\ -1 & \text{if the opponent scores a goal.} \end{cases} \tag{7}$$

R_m is a reward that encourages the robot to move to the target position. R_m in timestep ts is defined as:

$$R_m = \parallel r, t \parallel_{ts} - \parallel r, t \parallel_{ts-1} \tag{8}$$

where $\|\cdot\|$ is the euclidean distance. r and t are the positions of the robot and target.

R_{bd} is a reward for the direction of ball's movement. It is positive when the ball is taken toward the opponent's goal line and negative toward its own goal line. R_{bp} in timestep ts is defined as:

$$R_{bd} = bd_{ts} - bd_{ts-1} \tag{9}$$

$$bd = \frac{\frac{\|g_t, b\| - \|g_o, b\|}{l} - 1}{2} \tag{10}$$

where g_t and g_o are the positions of the center of the team goalpost and opponent goalpost respectively, b is the position of the ball, and l is the total length of the field, which is 1.5 in our experiment.

The energy penalty R_e is a negative reward used to penalize the robot's wheel speed, which is defined as:

$$R_e = -\left(|v_l| + |v_r|\right) \tag{11}$$

where v_l and v_l are the linear velocities of the left and right wheels respectively.

The speed penalty R_s is a negative reward when the robot is considered stationary, used to penalize the robot for being lazy or getting stuck, which is defined as:

$$R_s = \begin{cases} 0 & \text{if } s \geq ts \\ -1 & \text{if } s < ts \end{cases} \tag{12}$$

where s is the linear speed of the robot and ts is the threshold value that determines whether the robot is stationary or not, which is set to 0.01 in our experiment.

For each agent, the final reward at each time step is calculated as a weighted sum of the above rewards:

$$R = w_g R_g + w_m R_m + w_{bp} R_{bp} + w_e R_e + w_s R_s \quad (13)$$

where the weights are based on the default settings of rSoccer and, for those values that are not available, are determined by trial and error. In our experiments, $w_g = 50$, $w_m = 0.2$, $w_{bp} = 0.8$, $w_e = 2 \times 10^{-6}$, $w_s = 0.5$.

The evaluation focused on these questions:

- RQ1: How is P-HRL compared to baseline in terms of end to end performance?
- RQ2: How well does the P-HRL adapt to the new environment (new opponent) compared to baseline?
- RQ3: How do the individual components (SPPN, dynamic subtask assignment) contribute to the overall system?

Evaluation Metrics

The number of goals scored is undoubtedly the most direct indicator to victory. However, it could not be enough to assess the performance of two teams by score alone. In many cases, both teams have the same or similar number of goals, or even no goals at all, which makes it impossible to compare the two teams. For a more comprehensive evaluation, based on the common metrics used in human soccer analysis we defined the following key performance indicators(KPIs) applicable to robot soccer:

Ball possession rate The last robot to control the ball is defined to have possession of the ball. In particular, if at least one robot from both sides is close to ball and competing for the ball simultaneously, it is considered that no one has possession of the ball. Historically, it has been believed that higher possession is associated with scoring advantage (Lago and Martín 2007; Lago-Ballesteros and Lago-Peñas 2010; Parziale and Yates 2013). In particular, we use the clock time method to measure a team's possession rate. For a team, possession rate can be calculated as the sum of the time that all robots on the team have possession of the ball as a percentage of the total time of the game.

Number of Passing Passing is defined as the transfer of possession from one player to another player of the same team. For a team, the number of passing is counted as the sum of the number of passing made by all players. Passing brings an advantage in that the team secures possession of the ball while bringing the ball to a good position to attack. The number of passing is a key indicator of the coordination between players.

Number of interception Interception is defined as the transfer of possession from one player to an opponent's player. For a team, the number of interception is counted as the sum of the number of interception made by all players. Frequent interceptions means that players are more motivated to compete for ball possession, bringing more scoring opportunities for the team.

End-to-end Performance

To test the end-to-end performance, we use MATD3(Ackermann et al. 2019), an MARL algorithm used in many studies for multi-agent cooperation tasks similar to robot soccer. In the environment mentioned above, P-HRL controls one team of three robots and MATD3 controls another team of three robots for soccer matches. The results of the competition are used to evaluate the end-to-end performance of the P-HRL.

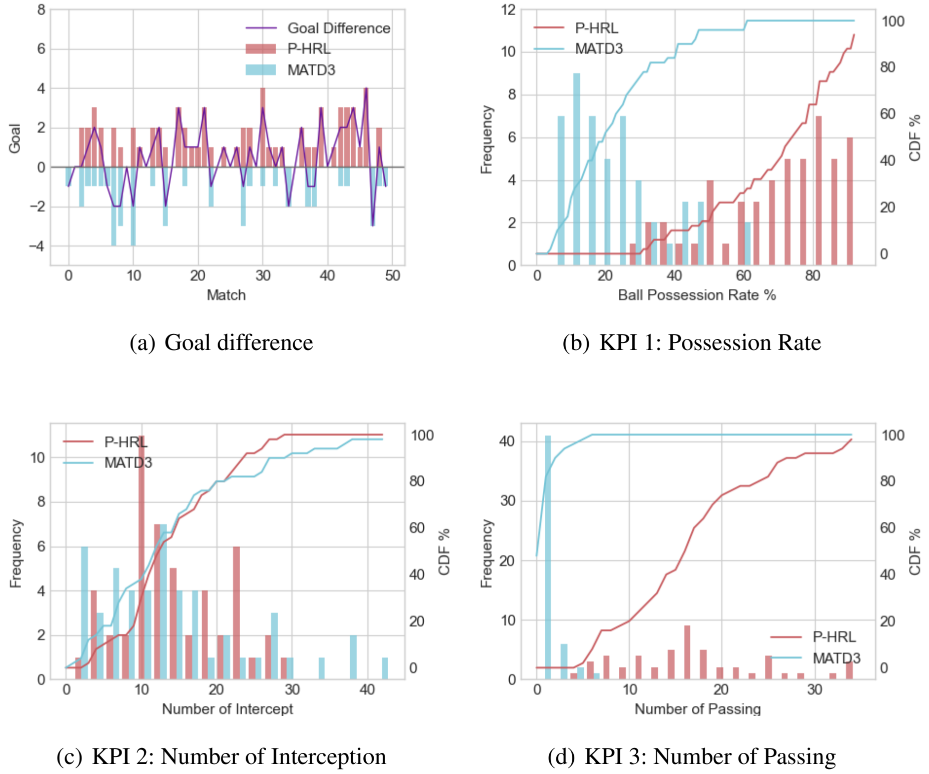
Table 1: Hyperparameters

Name	*Value*
Critic hidden layers size	64
Critic learning rate	1×10^{-4}
Actor hidden layers size	64
Actor learning rate	1×10^{-4}
Batch size	1024
Discount Factor (gamma)	0.95
Soft Update (tau)	0.01
Initial Noise	0.2
Noise decay rate	5×10^{-7}
Minimum noise	0.05

We trained P-HRL and MATD3 respectively in the above environment, using rSoccer built-in algorithm as the opponent. Unless otherwise specified, All reinforcement learning-related training parameters in are consistent, see Table 1. The training lasts until the average reward of each step does not rise with the episode (about 10000k steps), and the changes in the reward and goal scored of the two methods during the training process are shown in Figure **??**.

We do further training to ensure that the model is adequately trained in case the build-in opponents in rSoccer might not be competitive enough to reach the full potential of the model. Inspired by Robust Adversarial Reinforcement Learning(RARL) proposed by Pinto et al.(2017), we further train P-HRL with MATD3 in a mutual confrontation using a RARL-like approach. Based on the converged model output from the previous experiment as the starting point, the two sides of the competition are set to P-HRL and MATD3 respectively, alternately training two sets of agents: In the first stage, P-HRL is trained and parameters are updated, including the coach and the agents, while keeping the MATD3 strategy unchanged; In the next stage, P-HRL's policies remain unchanged and MATD3's policies are learned. Repeat this sequence until convergence.

For evaluation,the robots controlled by both algorithms play 50 matches in the environment as described above, each match lasting 2000 time steps. The results of the matches are shown in Figure 3. Out of the total 50 matches, P-HRL won 26 matches, MATD3 won 13 matches and tied 11 matches. Figure 3(a) show that P-HRL scored more goals than MATD3. Figure 3(b),3(c),3(d) show the differences in their KPIs. P-HRL outperform MATD3 in ball possession rate and number of passing. P-HRL has an average of 70.25% possession and 14.32 passes per game; MATD3 has an average of 17.14% possession and 1.92 passes per game. In terms of interceptions, P-HRL and MATD3 averages 14.40 and 14.44 respectively, with no significant differ-



(a) Goal difference

(b) KPI 1: Possession Rate

(c) KPI 2: Number of Interception

(d) KPI 3: Number of Passing

Figure 3: Results of P-HRL vs. MATD3 over 50 matches. In (a), the red bar and blue bar are the goal scoring of P-HRL and MATD3 respectively, and the purple dash line is the goal difference between the two (P-HRL minus MATD3). In (b-d), the histograms show the distribution of the average possession rate, interception, and passing per match for P-HRL and MATD3; the dash lines are the cumulative distribution function(cdf).

ences found.

Ability against different opponents

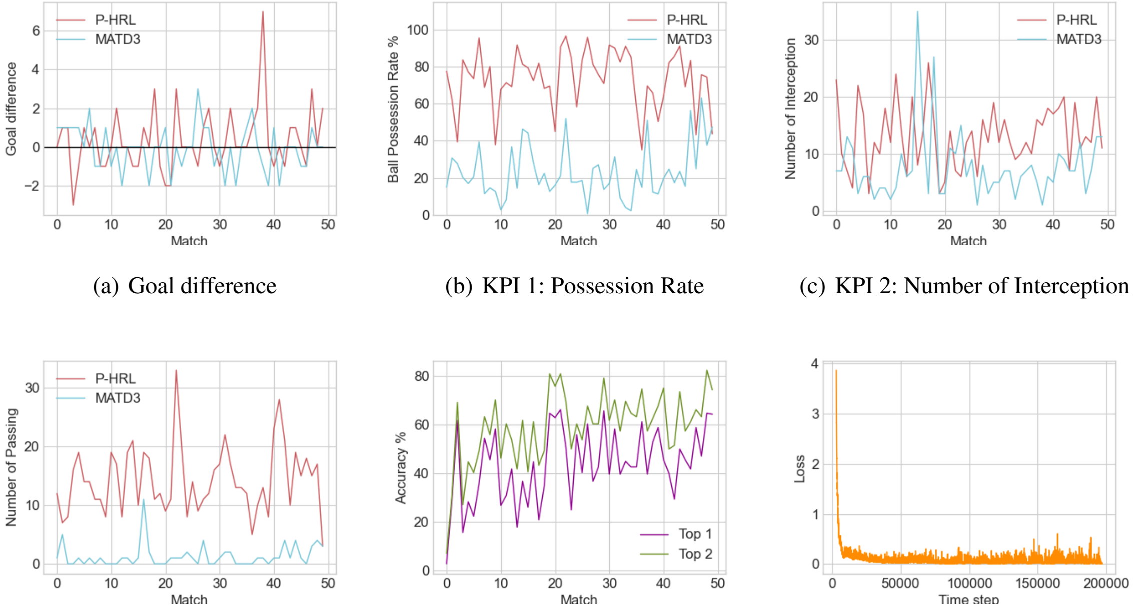
We investigated the performance of the model against different opponents. This simulates that in real robot training, the opponent's strategy is unknown during the training phase. Once on the field, the robots are supposed to fine-tune their strategies to counter their opponents.

We use MADDPG (Lowe et al. 2017) , a commonly used MARL algorithm, as the opponent. Use the P-HRL and MATD3 models trained in the previous chapter as starting points. Let the two play against MADDPG separately to evaluate their adaptability. Note that neither model has been trained with MADDPG as the opponent, therefore, it is safe to say that they both play against a new opponent.

A total of 50 matches were played, each lasting 2000 time steps. In this process, MATD3 got the entire network trained, while P-HRL got only the coach trained, with the robot controllers unchanged. In the respective 50 matches, P-HRL won 18 and tied 14; MATD3 won 15 and tied 15. The results in Figure 4(a) show that the goal difference of P-HRL is higher than MATD3 and tends to increase slightly over the 50 episodes, pwhile no significant trend is seen for MATD3. One possible reason for the increase could be the convergence of SPPN. Figure 4(e),4(f) show the loss and accuracy of SPPN during training. It can be observed that the loss decreases rapidly in the first period and almost converges within 40,000 time steps. The accuracy of prediction increase within 50 matches and reaches a maximum of 66.17% top1 accuracy and 82.35% top2 accuracy, which is close to end-to-end performance model with long training time (average of 67.49% top1 accuracy and 83.00% top2 accuracy over 50 matches). Figure 4(b),4(c),4(d) show that P-HRL has better overall performance than MATD3 in ball possession rate, number of interception and number of passing. P-HRL has an average of 73.18% possession, 13.16 interceptions and 14.32 passes per match; MATD3 had an average of 23.10% possession, 7.72 interceptions and 1.28 passes per match.

Ablation Study

SPPN To compare the effect of SPPN in the overall model, we kept the robot controllers unchanged and replaced the SPPN with the trained neural network(NN) model and the random nearest model, respectively. The random nearest model selects two random adjacent zones of the current zone as the prediction results. The original model and the replaced model each played 50 matches with baseline, each lasting



(a) Goal difference (b) KPI 1: Possession Rate (c) KPI 2: Number of Interception

(d) KPI 3: Number of Passing (e) SPPN Accuracy (f) SPPN loss

Figure 4: Comparative results of P-HRL and MATD3 against MADDPG in 50 matches with learning. In (a-d), the red and blue lines represent the goal difference(P-HRL minus MADDPG, MATD3 minus MADDPG), possession rate, number of interception, and number of passing of P-HRL and MATD3 against MADDPG over the number of matches. (e-f) show the prediction accuracy and loss of the SPPN module of P-HRL with respect to the number of matches.

Table 2: Coach

Prediction Net	*Team Score : Opponent Score*	*Acc Top 1*	*Acc Top 2*
SPPN	**2.40±1.78 : 1.48 ± 1.05**	**0.70±0.06**	**0.85±0.04**
NN	2.15±1.22 : 1.55 ± 1.31	0.59±0.06	0.80±0.05
Random Nearest	1.85±1.35 : 2.15 ± 1.39	0.06±0.11	0.12±0.06

2000 time steps. The results of goal difference and SPPN prediction accuracy for both are shown in Table 2. The results show that the SPPN model outperforms NN and random nearest model in terms of accuracy of both goal difference(average 0.92 goal difference) and prediction accuracy(average 70% top1 accuracy and 85% top2 accuracy).

Dynamic Subtask Assignment Method To compare the effect of dynamic subtask assignment method in the overall model, we keep the robot controllers unchanged and replace the current setting with two different fixed subtask assignment settings: offensive setting and defensive setting. Regardless of the ball possession, in the offensive setting, coach will only assign offensive subtasks; in the defensive setting, the robot closest to the ball is assigned offensive subtask and the remaining two are assigned defensive subtask. Three models of different settings each played 50 matches of 2000 time steps with baseline. The results of goal difference and KPIs for both are shown in Table 3. The results show that dynamic setting perform better on goal scored(average 0.92 goal difference) and possession rate(average 72.91 ball possession rate). Offensive setting showed some advantage in average number of passing(42.2) and interception(58.70), however, both sides scored significantly fewer goals than the other settings(average 1.85 goals for the team and 1.20 for the opponent). A possible reason for this is that the high number of passing and interception reflects the intense competition and frequent turnover of ball possession, resulting in difficulties for both sides to eventually score goals.

Conclusion

References

Ackermann, J.; Gabler, V.; Osa, T.; and Sugiyama, M. 2019. Reducing overestimation bias in multi-agent domains using double centralized critics. arXiv:1910.01465.

Antonioni, E.; Suriani, V.; Riccio, F.; and Nardi, D. 2021. Game strategies for physical robot soccer players: a survey. *IEEE Transactions on Games*, 13(4): 342–357.

Table 3: Subtask Assignment

Subtask Assignment Setting	*Team Score : Opponent Score*	*Possession Rate %*	*Interception*	*Passing*
Dynamic Setting	**2.40 ±1.78 : 1.48 ± 1.05**	**72.91±9.82**	39.88 ±9.07	56.90 ±15.40
Offensive Setting	1.85±0.93 : 1.20 ± 1.23	72.33±9.82	**42.20 ±11.30**	**58.70 ±22.35**
Defensive Setting	2.35±1.31 : 1.55 ± 1.43	70.83±7.78	42.00 ±11.55	53.45 ±14.25

Bassani, H. F.; Delgado, R. A.; Junior, J. N. d. O. L.; Medeiros, H. R.; Braga, P. H.; Machado, M. G.; Santos, L. H.; and Tapp, A. 2020. A Framework for Studying Reinforcement Learning and Sim-to-Real in Robot Soccer. *arXiv preprint arXiv:2008.12624*.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680.

Lago, C.; and Martín, R. 2007. Determinants of possession of the ball in soccer. *Journal of sports sciences*, 25(9): 969–974.

Lago-Ballesteros, J.; and Lago-Peñas, C. 2010. Performance in team sports: Identifying the keys to success in soccer. *Journal of Human kinetics*, 25(1): 85–91.

Lowe, R.; Wu, Y. I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30.

Martins, F. B.; Machado, M. G.; Bassani, H. F.; Braga, P. H.; and Barros, E. S. 2022. rSoccer: A Framework for Studying Reinforcement Learning in Small and Very Small Size Robot Soccer. In *Robot World Cup*, 165–176. Springer.

Parziale, E. J.; and Yates, P. A. 2013. Keep the ball! The value of ball possession in soccer. *Reinvention: an International Journal of Undergraduate Research*, 6(1): 1–24.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, 2817–2826. PMLR.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, 3540–3549. PMLR.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, T.; Gupta, T.; Mahajan, A.; Peng, B.; Whiteson, S.; and Zhang, C. 2020. Rode: Learning roles to decompose multi-agent tasks. arXiv:2010.01523.

Zhang, T.; Xu, H.; Wang, X.; Wu, Y.; Keutzer, K.; Gonzalez, J. E.; and Tian, Y. 2020. Multi-agent collaboration via reward attribution decomposition. arXiv:2010.08531.