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 long-term credit assignment problem, it is difficult for MARL to learn to collaborate in robot soccer. Secondly, high-dimensional and continuous state-action spaces make it difficult for MARL 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-term tasks, especially where other opponents are present and rewards are sparse. To address these challenges, we propose Prediction-based Hierarchical Reinforcement Learning (P-HRL) for robot soccer. P-HRL consists of a coach for soccer tactics and a robot controller for robot motion control. To professionally evaluate the performance of P-HRL, we design various key performance indicators (KPIs) for robot soccer. Experimental results demonstrate that P-HRL has a better performance than the state-of-the-art baseline MATD3. In matches against MATD3, P-HRL has 52% win rate and 26% loss rate. In matches against the new opponent, P-HRL has 36% win rate and 34% loss rate, compared to 30% win rate and 40% loss rate for MATD3.

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; Asada et al. 2019). In robot soccer, MAS controls the robots on the team to cooperate and score goals. Several leagues on robot soccer have been successfully organized in recent years, such as RoboCup, IEEE Very Small Size Soccer (IEEE VSSS) and many research problems based on robot soccer have received extensive attention, such as path planning(Cooksey and Veloso 2017; Macenski et al. 2020).

Multi-agent reinforcement learning (MARL) have achieved outstanding success on cooperative problems such as video games(Zhang et al. 2020; Wang et al. 2020) and robot control(Perrusquía, Yu, and Li 2021). MARL controls the actions of multiple agents based on reward to maximize the return by continuously interacting with the environment. The unsupervised learning enables MARL to achieve end-to-end autonomous learning and allows MARL to continuously improve its performance without increasing its reliance on domain knowledge (Buşoniu, Babuška, and Schutter 2010). Thus, the development of a MARL-based approach in robot soccer has the potential to outperform other non-reinforcement learning approaches(Akiyama et al. 2018; Chen et al. 2019). We present the following challenges to MARL in robot soccer.

First, the sparsity of rewards and the long-term credit assignment problem(Vezhnevets et al. 2017; Harutyunyan et al. 2019) make it difficult for MARL to learn to collaborate. MARL can only score goals in soccer games after a long time of appropriate decision making, which makes MARL get sparse positive rewards. MARL learns actions by using rewards to estimate expected returns(Mnih et al. 2013). Sparse rewards can cause MARL to misestimate the expected return and lead MARL to take inappropriate actions to control the robot.

Moreover, long-term credit assignment exacerbates MARL's inappropriate control of robots. It is impossible to judge the contribution of each player to the goal by only scoring, which leads MARL to misestimate the expected return of each robot's actions on the process of scoring a goal. This results in unpredictable variance between MARL's assignment of expected returns to each robot and the correct assignment. Previous work(Abreu, Reis, and Cardoso 2019; de Medeiros, Marcos, and Yoneyama 2020) addressed the problem of reward sparsity by reward shaping. But reward shaping does not solve the problem of long-term credit assignment in robot soccer because of the difficulty in trade-off of different rewards. The contribution of players to a goal is related to many factors such as possession and positions of the opposing players. Any inappropriate quantitative trade-off of these factors 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 robotic cooperative tasks(Mordatch and Abbeel 2018; Todorov, Erez, and Tassa 2012), the unknown and potential behavior of the opponent robots leads to stochasticity in the environment. This resulted in MARL need to explore the entire state-action space rather than partial state-action space to find the optimal policy. However, MARL faces the problem of the curse of dimensionality (Nguyen, Nguyen, and Nahavandi 2020): The dimensionality of the state-action space increases exponentially with

respect to the degrees of freedom and the number of robots. Thus, it impossible for MARL to fully explore the entire state-action space.

Inspired by human soccer, 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. The *coach*, just like the head coach or manager of a human soccer team, plans strategies for the players on the field to beat the opponent. Hierarchical reinforcement learning (HRL) provides a framework for improving MARL through *coach*. HRL learns the same task at multiple time scales, and the policies at each time scales jointly determine the behavior of the agent(Pateria et al. 2021).

In this paper, 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 compensates for the sparsity of rewards of goal scoring, and provide different tasks and corresponding rewards for each robot. 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(Bassani et al. 2020). 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 KPIs 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 better performance than the state-of-the-art MARL methods. In matches against MATD3, P-HRL has 52% win rate and 26% loss rate. P-HRL had an average ball possession rate of 70.25%, compared to 17.14% for MATD3. In matches against the new opponent, P-HRL has 36% win rate and 34% loss rate, compared to 30% win rate and 40% loss rate for MATD3.

Background

Multi-agent Reinforcement Learning

Learning and acting in continuous action spaces is a key challenge for MARL applications in the real world. In reinforcement learning, (Watkins 1989) proposed a method to learn the value of an action using a Q-function to approximate the expected return and a greedy selection of the optimal policy. (Mnih et al. 2013) proposed the use of multilayer perceptrons, which is called deep Q-networks (DQN), to approximate the Q-function. In addition, they use a replay buffer to update the parameters in the DQN. (Hasselt 2010) proposed to use two Q-functions for learning separately to optimize the problem of overestimation of the Q-function, which is called Double Q-learning. In 2015, (Lillicrap et al. 2015) proposed deep deterministic policy gradient (DDPG) for continuous and high action spaces, which is an actor-critic approach based on DQN. It uses deterministic policy gradients (Silver et al. 2014) and target networks to update actor policies.

MADDPG (Lowe et al. 2017) is an approach to extend DDPG from a single-agent setting to a multi-agent setting. It requires that each agent's critic network has access to all agents' actor networks at training, but not needed at execution. MATD3 (Ackermann et al. 2019) is an approach as TD3 (Fujimoto, Hoof, and Meger 2018) based on MADDPG to optimize the Q-function overestimation bias.

Prediction-based Hierarchical Reinforcement Learning (P-HRL)

Figure 1 shows the temporal process of P-HRL. 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 $< \mathcal{S}, \mathcal{O}, \mathcal{G}, \mathcal{U}, P, \mathcal{R}, \gamma >$, where $\mathcal{S}$ is state space. At each time step t, $s_t \in \mathcal{S}$ is the state. The state s_t 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. $g \in \mathcal{G}$ is the subtask. $\mathbf{u_t} = \left(u_t^1, ..., u_t^n\right)$ 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 and is the probability of the succeeding state s_t at time step $t+1$ after taking joint action $\mathbf{u}_t$ in state s_t at time step t. $r_t = \left(r_t^1, ..., r_t^n\right)$ is the external reward from the environment, where r_t^i is the external reward of the i-th robot. γ is the discount factor. A complete robot soccer game is defined as:

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

where $\tau = \left\{s_0, \mathbf{u}_0, s_1, \mathbf{u}_1, ..., s_T, \mathbf{u}_T\right\}$ 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

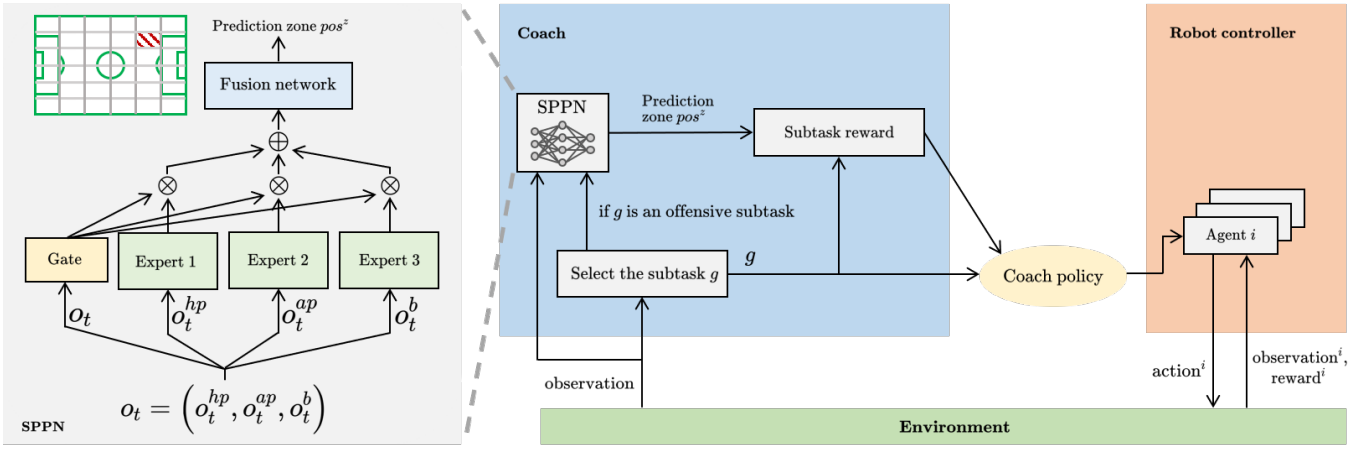


Figure 1: Illustration of the P-HRL. At each time step t, coach receives observation and starts (or continues) a subtask g. If g is a new offensive subtask, the coach calculates the reward based on the predicted zone's centroid pos^z of the SPPN. In SPPN, the observation of the home team player o_t^{hp} , away team player o_t^{ap} and ball o_t^b are input separately to different expert networks. Then, the coach calculates the corresponding subtask reward based on the selected subtask. In the robot controller, Each agent i outputs an action^i to control the corresponding robot based on coach policy, observation^i and reward^i.

when a goal is scored or the state when the game reaches the end time step $t = T_{max}$. The aim of the robot soccer is to maximize goal scores, which is defined by maximizing discounted accumulated rewards $\mathbb{E}\left[\sum_{t=0}^{T} \gamma r_t\left(s_t, \mathbf{u}_t\right)\right]$.

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.

The basic principle of all soccer tactics is to guarantee possession of the ball because the team that has possession of the ball has the opportunity to attack and score, otherwise it can only play passive defense. Thus, we divide subtasks into two types $\mathcal{G} = \{g^A, g^D\}$, where g^A is offensive subtask and g^D is defensive subtask. When our team gets possession of the ball, the coach starts an offensive subtask. When the opponent gets possession of the ball, the coach starts a defensive subtask. The offensive subtask aims to secure possession of the ball and score a goal, and the defensive subtask aims to get possession of the ball to prevent the opponent's attack. Specifically, in the defensive subtask, coach encourages all of our robots to intercept the ball to get possession of the ball. In the offensive subtask, our robot which is closest to the ball is called the attacker, and the rest of our robots are called supporters. The coach encourages the attacker to attack and score. The aim of supporters is to protect the possession of the ball. The coach encourages the supporter to go to the zone where the ball will be located and become the attacker to continue the attack. Moreover, if the opposing robot gets possession of the ball during this process, the supporter can immediately perform a defensive subtask to intercept the ball.

At each time step, the coach assigns subtasks and corresponding subtask rewards to each robot to guide the robot controller complete the subtasks. To predict the zone where the ball will be located, we propose the soccer position prediction network (SPPN) which is described in the following session. Subtask reward for the i-th robot r_g^i is defined as encouraging the i-th robot to go to the target position (i.e. the position of the ball or the zone predicted by the SPPN):

$$r_g^i = \frac{pos^r - pos^g}{\|pos^r - pos^g\|} \cdot v^r \tag{2}$$

$$where \quad pos^g = \begin{cases} pos^b & \text{if } g = g^A \text{ and robot is attacker.} \\ pos^z & \text{if } g = g^A \text{ and robot is supporter.} \\ pos^b & \text{if } g = g^D \end{cases}$$

where $\| \cdot \|$ is the euclidean distance, pos^r and pos^b is the position vector of the i-th robot and the ball respectively. v^r is the i-th robot speed vector. These vectors are included in the observation o_t obtained by the coach at time step t. pos^z is the centroid position vector of the prediction zone. Table 1 shows the definition of initiation conditions I_g and termination conditions β_g. It is worth noting that after T_p time steps, the offensive subtask is restarted if our robots have possession of the ball. SPPN will predict new zones to ensure that our robots can keep the attack.

Soccer Position Prediction Network To achieve the zoned soccer prediction we mentioned in the previous section, we propose SPPN. SPPN is inspired by Mixture of Experts (Eigen, Ranzato, and Sutskever 2013).

Robot soccer presents a challenge to soccer position prediction. The data in the observation can be divided into three types: data related to the home team player (i.e. our

Table 1: Definition of initiation and termination conditions

	Offensive subtask g^A	Defensive subtask g^D
Initiation condition I_g	Our robots are in possession.	Opposing robots are in possession.
Termination condition β_g	(1) Opposing robots are in possession. (2) After T_p time steps.	Our robots are in possession.

robots), data related to the away team player (i.e. the opposing robots), and data related to the ball. In soccer games, the position of the soccer is determined by the actions of both team players. However, a single network structure cannot effectively model the data under observation because the data of the home player and the data of the away player are independent. To address these challenges, we propose soccer position prediction network(SPPN).

As a resuult, we propose SPPN which consists of the following three components:

Expert network: To effectively solve the problem of data independence in the observation, 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 allows modeling the complex relationship between the three types of data in the observation.

Fusion network: Fusion network is used to map the output features of the gating network with the zoned soccer prediction results.

We divide the soccer field into several equal rectangular zones, and SPPN is used to predict the zone where the ball will be located after T_p time steps. The SPPN section in Figure 1 shows the network structure and an example of a zoned soccer prediction result.

Formally, given a observation $o_t = \left(o_t^{hp}, o_t^{ap}, o_t^b\right)$ obtained by coach at time step t, where o_t^{hp}, o^{ap} and o^b is the part of the observation related to the home team players, away team players and ball, respectively. The prediction zone of SPPN can be formulated as below:

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

where $f_i\left(\cdot\right)$ is the i-th expert network, $g\left(\cdot\right)$ is the gating network, $f\left(\cdot\right)$ is a single fully connected layer, $h\left(\cdot\right)$ is the fusion network, and pos^z is the position vector of the centroid of the prediction zone. The expert networks and the fusion network are stacked by fully connected layers.

Robot Controller

In robot controller, we modify MADDPG(Lowe et al. 2017) to be used in the hierarchy structure of P-HRL and also use the centralized training with decentralized execution method. Agent i controls the motion of the i-th robot. 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:

$$Q_i\left(o_t, \mathbf{u}_t\right) = \mathbb{E}\left[\sum_{j=0}^{T} \gamma\left(r_{g_{t+j}}^i + r_t\right) \mid g, o_t, \mathbf{u}_t\right] \tag{4}$$

where $r_{g_{t+j}}^i$ is the subtask reward of agent i at time step $t+j$ and T is the duration of the subtask. 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:

$$\mathcal{L}\left(\theta^{Q_i}\right) = \mathbb{E}_{o,g,r_g,\mathbf{u},r,o'\sim\mathcal{D}}\left[\left(Q_i\left(o, \mathbf{u}, g \mid \theta^{Q_i}\right) - y\right)^2\right],$$
$$where \quad y = r^i + r_g^i + \gamma Q_i'\left(o', \mathbf{u}', g \mid \theta^{Q_i'}\right)|_{u_j'=\mu_j'(o^j)} \tag{5}$$

where r^i and r_g^i are the external reward and subtask reward for the agent i, respectively. $Q_i'\left(o', \mathbf{u}', g \mid \theta^{Q_i'}\right)$ 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 $\mu_j'\left(o^j\right)$, and the target actor network with parameter $\theta^{\mu_i'}$ is updated periodically by the parameter θ^{μ_i}. Using the critic network Q_i, the gradient of actor network μ_i used to optimize the deterministic policy of agent i can be written as:

$$\bigtriangledown_{\theta^{\mu_i}} J\left(\theta^{\mu_i}\right) = \mathbb{E}_{o,u^{j\neq i},g,r_g\sim\mathcal{D}}[\bigtriangledown_{\theta^{\mu_i}} \mu_i\left(o^i, g \mid \theta^{\mu_i}\right) \bigtriangledown_{u^i} Q_i\left(o, \mathbf{u}, g \mid \theta^{Q_i}\right)|_{u^i=\mu_i(o^i,g)}] \tag{6}$$

Training Method

To allow the robot controller to focus on completing the subtasks assigned by the coach, P-HRL uses parallel training of the coach and robot controller. Algorithm 1 describes the training parallel for P-HRL.

To make the SPPN in coach and the robot controller can be trained simultaneously, we use two replay buffer $\mathcal{D}_1$ and $\mathcal{D}_2$. In an episode, coach selects a subtask g (lines 5 to 12). Then, the robot controller controls robots to complete the subtask (lines 14 to 15). Every T_p time steps, P-HRL stores the observation o_t into replay buffer $\mathcal{D}_1$ for the training of SPPN (lines 16 to 18). P-HRL uses the samples in $\mathcal{D}_1$ to construct the training set and the labels to perform parameter updates on the SPPN (lines 20 to 21). P-HRL performs parameter updates on the robot controller using samples from $\mathcal{D}_2$ and deterministic policy gradient (lines 22 to 23).

Evaluation

Our experiments are based on rSoccer(Martins et al. 2022) - a framework for studying reinforcement learning for small

Algorithm 1: Parallel training for P-HRL

Initialize replay buffer $\{\mathcal{D}_1, \mathcal{D}_2\}$, SPPN in coach, actor networks and critic networks in robot controller.
for episode = 1 to M **do**
  Rest the environment.
  for t = 0 to $T_{max} - 1$ and not $occur\_goal\_scoring$ **do**
    if I_{g^A} is True **then**
      $g \leftarrow g^A$
      $\beta_g \leftarrow \beta_{g^A}$
      Use SPPN to predict zone pos^z.
    else
      $g \leftarrow g^B$
      $\beta_g \leftarrow \beta_{g^B}$
    end if
    while β_g is False **do**
      Sample a joint action $\mathbf{u}$ from robot controller.
      Execute the joint action $\mathbf{u}$, calculate subtask reward r_g using Eq 2, observe the external reward r and next observation o'.
      if t mod Tp=0 **then**
        Store observation o in replay buffer $\mathcal{D}_1$.
      end if
      Store $(o, g, r_g, \mathbf{u}, r, o')$ in replay buffer $\mathcal{D}_2$.
      Randomly sample mini-batches from $\mathcal{D}_1$.
      Update SPPN by samples and cross-entropy loss function.
      Randomly sample mini-batches from $\mathcal{D}_2$.
      Update robot controller by samples, Eq 5 and Eq 6.
    end while
  end for
end for

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 (0.08 m $\times$ 0.08 m) with wheels but no hardware to dribbling or kicking the ball. The size of the soccer field is 1.5 m $\times$ 1.3 m.

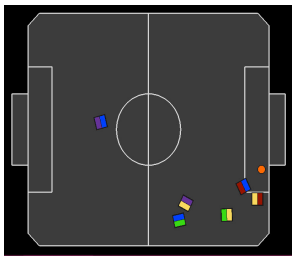


Figure 2: rSoccer VSSMA Environment

In our experiments, we basically followed the reward shaping in rSoccer (Bassani et al. 2020). Exceptionally, we changed the 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:

$$R_m = \| r, t \|_{ts} - \| r, t \|_{ts-1} \tag{8}$$

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:

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

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 opponent compared to baseline?
- RQ3: How do the coach 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 evaluate simply by the score. In many cases, both teams have the same or a similar number of goals, or even no goals at all, making it impossible to compare the performance of the two teams. 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 as having possession of the ball. In particular, if at least one robot from both teams approach ball and compete 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 Passes Passing is defined as the transfer of possession from one player to another player of the same team. The number of passes made by a team is the sum of all passes 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 passes 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 interceptions is counted as the sum of interceptions made by all players. Frequent interceptions mean 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 state-of-the-art actor-critic-based algorithm that has been used in many multi-agent cooperative tasks similar to robot soccer. In the above environment, 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.

We trained P-HRL and MATD3 respectively in the above environment, using rSoccer built-in algorithm as the opponent. Unless otherwise specified, all RL-related training parameters in this evaluation section are consistent, see Table 2. The training lasts until the average episode reward does not rise with the episode (about 1×10^6 steps), and the changes in the reward and goal difference of the two methods during the training process are shown in Figure 3(e),3(f).

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.

Table 2: 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

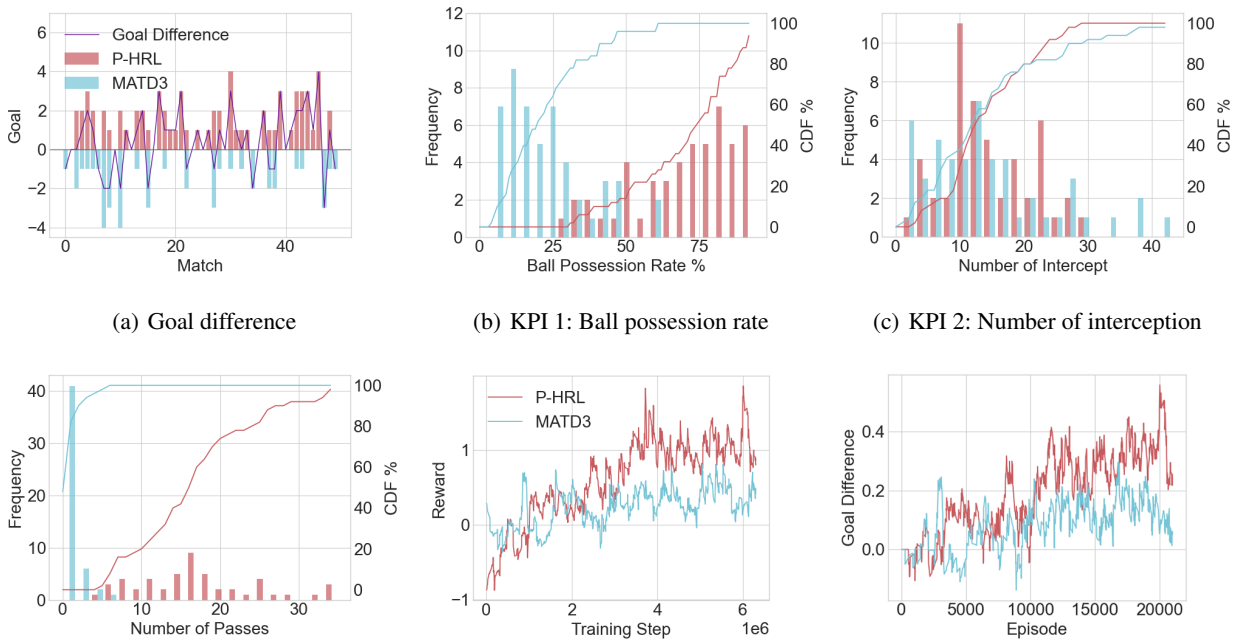
For evaluation, the robots controlled by both algorithms play 50 matches in the environment as described above, each match lasting 2000 time steps. The results of the matches are shown in Figure 3. Out of the total 50 matches, P-HRL won 26 matches, MATD3 won 13 matches and tied 11 matches. Figure 3(a) show that P-HRL scored more goals than MATD3. Figure 3(b), 3(c), 3(d) show the differences in KPIs. P-HRL outperform MATD3 in ball possession rate and number of passes. 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 difference found.

Ability against new opponents

We investigated the performance of the model against new 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 widely used actor-critic-based MARL algorithm, as the new 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 timesteps. 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,



(a) Goal difference (b) KPI 1: Ball possession rate (c) KPI 2: Number of interception

(d) KPI 3: Number of passes (e) Reward in training (f) Goal difference in training

Figure 3: (a) Results of P-HRL vs. MATD3 over 50 matches. The purple dash line represents the difference between the two (P-HRL minus MATD3). (b-d) Histogram shows the distribution of the average KPI per match; the dash line is the cumulative distribution function (CDF).(e-f) Change of reward and goal difference with training. Smoothed by exponential moving average (EMA) with exponential smoothing constant K=0.95.

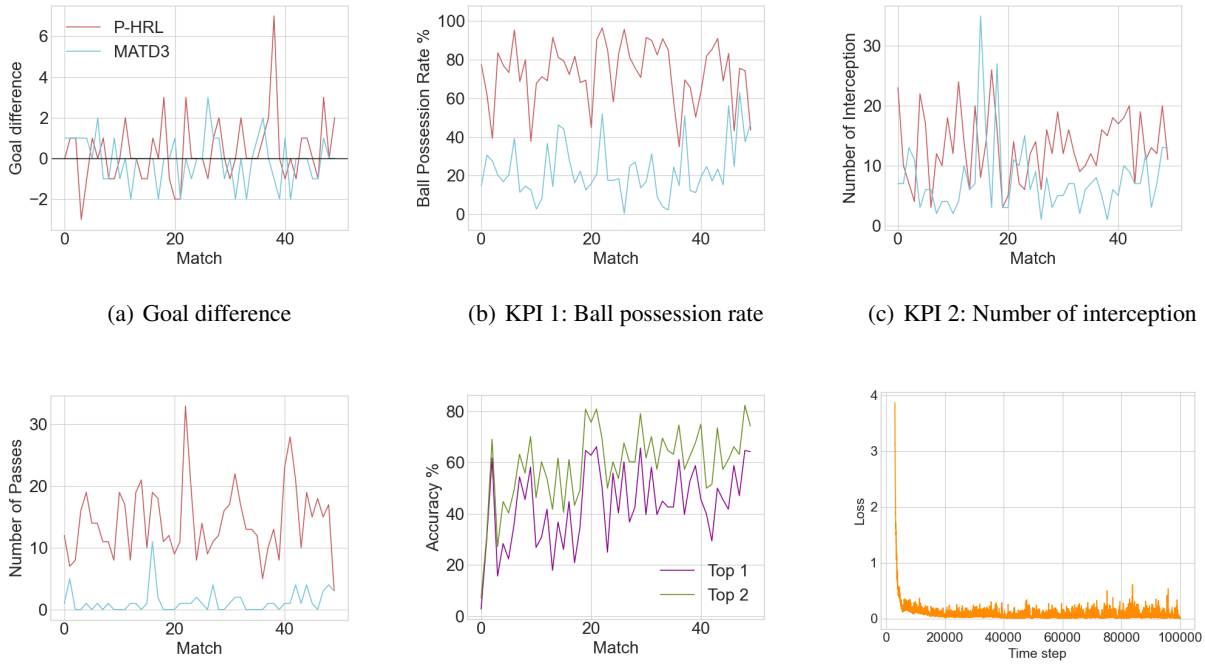
P-HRL won 18 and tied 14; MATD3 won 15 and tied 15. The results in Figure 4(a) show that the goal difference of P-HRL is higher than MATD3 and tends to increase slightly over the 50 episodes, while no significant trend is seen for MATD3. One possible reason for the increase could be the convergence of 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 converges approximately within 20000 time steps. The accuracy of prediction increases within 50 matches and reaches a maximum of 66.17% top1 accuracy and 82.35% top2 accuracy, which is close to the model tested in end-to-end performance experiment 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, the number of interceptions and the number of passes. P-HRL had 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

To understand the effectiveness of coach 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 structure of the NN coach is similar to SPPN, but with only one expert in the prediction network. The observed data are not segmented and are all fed into the unique expert. 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 2000 time steps. The results of goal difference and SPPN prediction accuracy for both are shown in Table 3. 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).

Conclusion

In this paper, we identify two challenges for MARL applications in robot soccer: sparsity of reward and the difficulty of adapting to new opponents due to high and continuous state action spaces. We propose P-HRL, a hierarchical approach in which coach decomposes the soccer game into a series of subtasks, and the robot controller controls the motions of robots to complete these subtasks. It is the first hierarchical reinforcement learning method applied to robot soccer. Experiments demonstrate that P-HRL has better end-to-end performance than baseline, including more goals scored and better KPIs, and has a stronger ability to adapt to new opponents.



(a) Goal difference (b) KPI 1: Ball possession rate (c) KPI 2: Number of interception

(d) KPI 3: Number of passes (e) SPPN accuracy (f) SPPN loss

Figure 4: 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) and three KPIs 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.

Table 3: Match results for different types of coach (vs. MATD3 in 50 matches)

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

References

Abreu, M.; Reis, L. P.; and Cardoso, H. L. 2019. Learning high-level robotic soccer strategies from scratch through reinforcement learning. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 1–7. IEEE.

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

Akiyama, H.; Nakashima, T.; Fukushima, T.; Zhong, J.; Suzuki, Y.; and Ohori, A. 2018. Helios2018: Robocup 2018 soccer simulation 2D league champion. In *Robot World Cup*, 450–461. Springer.

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.

Asada, M.; Stone, P.; Veloso, M.; Lee, D.; and Nardi, D. 2019. Robocup: A treasure trove of rich diversity for research issues and interdisciplinary connections [tc spotlight]. *IEEE Robotics & Automation Magazine*, 26(3): 99–102.

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*.

Buşoniu, L.; Babuška, R.; and Schutter, B. D. 2010. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, 183–221.

Chen, Z.; Zhang, H.; Guo, D.; Jia, S.; Fang, X.; Huang, Z.; Wang, Y.; Hu, P.; Wen, L.; Chen, L.; et al. 2019. Champion team paper: Dynamic passing-shooting algorithm of the RoboCup soccer SSL 2019 champion. In *Robot World Cup*, 479–490. Springer.

Cooksey, P.; and Veloso, M. 2017. Intra-robot replanning to enable team plan conditions. In *2017 IEEE/RSJ Interna-*

tional Conference on Intelligent Robots and Systems (IROS), 1113–1118. IEEE.

de Medeiros, T. F.; Marcos, R. d. A.; and Yoneyama, T. 2020. Deep Reinforcement Learning Applied to IEEE Very Small Size Soccer Strategy. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, 1–6. IEEE.

Eigen, D.; Ranzato, M.; and Sutskever, I. 2013. Learning factored representations in a deep mixture of experts. arXiv:1312.4314.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Harutyunyan, A.; Dabney, W.; Mesnard, T.; Gheshlaghi Azar, M.; Piot, B.; Heess, N.; van Hasselt, H. P.; Wayne, G.; Singh, S.; Precup, D.; et al. 2019. Hindsight credit assignment. *Advances in neural information processing systems*, 32.

Hasselt, H. 2010. Double Q-learning. *Advances in neural information processing systems*, 23.

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.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. arXiv:1509.02971.

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.

Macenski, S.; Martín, F.; White, R.; and Clavero, J. G. 2020. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2718–2725. IEEE.

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.

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

Mordatch, I.; and Abbeel, P. 2018. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Nguyen, T. T.; Nguyen, N. D.; and Nahavandi, S. 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9): 3826–3839.

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.

Pateria, S.; Subagdja, B.; Tan, A.-h.; and Quek, C. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5): 1–35.

Perrusquía, A.; Yu, W.; and Li, X. 2021. Multi-agent reinforcement learning for redundant robot control in task-space. *International Journal of Machine Learning and Cybernetics*, 12(1): 231–241.

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

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*, 387–395. PMLR.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033. IEEE.

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.

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.

Watkins, C. J. C. H. 1989. Learning from delayed rewards.

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.