When to Align: Dynamic Behavior Consistency for Multiagent Systems via Intrinsic Rewards

Kunyang Lin, Yufeng Wang, Peihao Chen, Runhao Zeng, Yinjie Lei, *Senior Member, IEEE*, Siyuan Zhou, Qing Du, Mingkui Tan, and Chuang Gan

***Abstract*—In multiagent systems, learning optimal behavior policies for individual agents remains a challenging yet crucial task. While recent research has made strides in this area, the issue of when agents should maintain consistent behaviors with one another is still not adequately addressed. This article proposes a novel approach to enable agents to autonomously decide whether their behaviors should align with those of their peers by leveraging intrinsic rewards to optimize their policies. We define behavior consistency as the divergence between the actions taken by two agents given the same observations. To encourage agents to be aware of each other's behaviors, we propose dynamic consistency-based intrinsic reward (DCIR), which guides agents in determining when to synchronize their behaviors. In addition, we introduce a dynamic scaling network (DSN) that provides learnable scaling factors at each time step, enabling agents to dynamically decide the extent of rewarding consistent behavior. Our method is evaluated on environments including Multiagent Particle, Google Research Football, and StarCraft II Micromanagement. Experimental results demonstrate its effectiveness in learning optimal policies.**

***Index Terms*—Behavior consistency, intrinsic reward, multiagent reinforcement learning (MARL), reinforcement learning.**

Received 22 November 2024; revised 27 April 2025 and 5 July 2025; accepted 7 August 2025. Date of publication 10 September 2025; date of current version 3 December 2025. This work was supported in part by the Joint Funds of the National Natural Science Foundation of China under Grant U24A20327, in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2019B010155001, in part by the National Natural Science Foundation of China (NSFC) under Grant 62202311, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515011512, and in part by the Key Scientific Research Project of the Department of Education of Guangdong Province under Grant 2024ZDZX3012. *(Kunyang Lin and Yufeng Wang are co-first authors.) (Corresponding author: Mingkui Tan.)*

Kunyang Lin, Peihao Chen, Qing Du, and Mingkui Tan are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: imkunyanglin@gmail.com; phchencs@gmail.com; duqing@scut.edu.cn; mingkuitan@scut.edu.cn).

Yufeng Wang is with the School of Future Technology, South China University of Technology, Guangzhou 510641, China, and also with the Department of Networked Intelligence, Peng Cheng Laboratory, Shenzhen 518052, China (e-mail: yufeng6568@gmail.com).

Runhao Zeng is with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, China (e-mail: runhaozeng.cs@gmail.com).

Yinjie Lei is with the College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China (e-mail: yinjie@scu.edu.cn).

Siyuan Zhou is with The Hong Kong University of Science and Technology, Hong Kong (e-mail: szhoubr@connect.ust.hk).

Chuang Gan is with UMass Amherst, Amherst, MA 01003 USA (e-mail: ganchuang1990@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2025.3598301, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2025.3598301

I. Introduction

MULTIAGENT reinforcement learning (MARL) has been evidenced as an important technique in a wide range of practical real-world tasks. These tasks are set in multiagent systems with the goal of cooperation, such as robotic control [1], [2], [3], [4], video games [5], [6], [7], and autonomous vehicles [8], [9], [10]. In MARL, each agent is a reinforcement learning system with its own perception and decision-making capabilities, and multiple agents are required to learn and optimize their behavior strategies by interacting with the environment to achieve a common goal.

As the number of agents grows from one to multiple, it is challenging for each agent to find its optimal behavior. This is particularly obvious in many real-world scenarios where extrinsic rewards are sparse and delayed, exacerbating the difficulty of MARL. Deducing the reward of each agent is crucial to encourage the agents to organize their behaviors for successful collaboration [11]. While reward shaping [12] requires heavy and careful manual work, previous works are striving to cope with this challenge by assigning intrinsic rewards for each agent [11], [13], [14], [15], [16], [17]. These intrinsic reward methods can be roughly categorized into three groups: consistency-oriented methods, diversity-oriented methods, and exploration-driven methods. Consistency-oriented methods (e.g., ELIGN [11] and IAM [18]) promote behavioral alignment by encouraging agents to act consistently with teammates, thereby improving stability. However, these methods generally assume that consistency is always desirable, which may not hold in dynamic environments. Diversity-oriented methods (e.g., LIIR [13] and CDS [19]) stimulate agents to behave differently from others, often to promote coverage, robustness, or opponent disruption. While diversity improves exploration and specialization, these methods may ignore situations where behavioral alignment is beneficial. Exploration-driven methods [15], [16], [20] focus on novelty-seeking by rewarding agents for visiting unfamiliar states, either from an individual or team perspective. However, they do not consider whether agents should coordinate or differentiate their actions. Despite these advances, a fundamental question remains underexplored: *when should agents align their behaviors, and when should they diverge?* Current methods typically adopt a static strategy: either always aligning or always diversifying, regardless of context. This rigid design may limit agents' adaptability in tasks that require dynamic coordination, such as dividing subgoals or escaping from adversaries. To this end, we equip agents with the ability to dynamically adjust their alignment decisions based

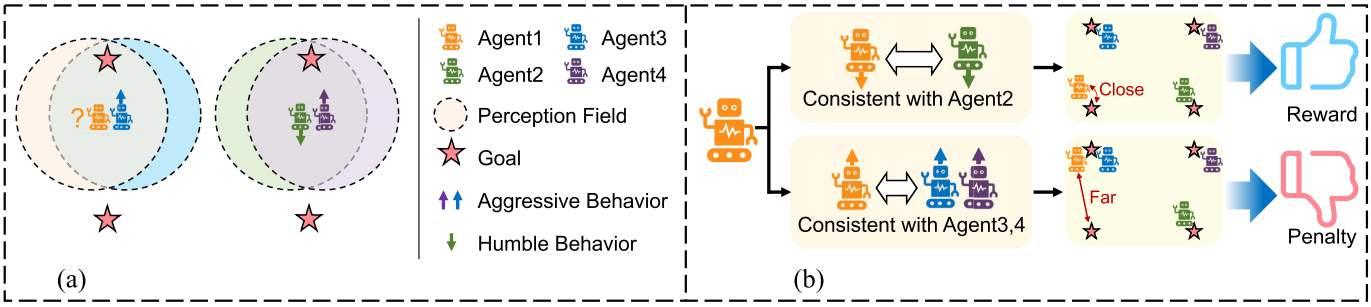

Fig. 1. (a) In a multiagent cooperation task, four agents must cooperate to reach four goals simultaneously as soon as possible. In the context that *Agent3* and *Agent4* take aggressive behavior to directly walk toward the goal that they have observed, *Agent1* and *Agent2* should share consistent humble behavior to explore more targets. (b) Our dynamic consistency-based intrinsic reward (DCIR) awards *Agent1* if it behaves consistently with *Agent2* while punishing it if it behaves consistently with *Agent3* and *Agent4*.

on task context and peer behavior, thereby improving overall cooperation efficiency and flexibility.

To illustrate the importance of both behavior consistency and inconsistency, we consider a collaborative navigation scenario, where four agents aim to collectively reach four goals. As shown in Fig. 1, Agent2 and Agent4 share similar observations, wherein they both observe the same goal. However, to facilitate efficient collaborative task completion and avoid conflicts, it is recommended that they behave inconsistently. Expressly, Agent4 should adopt a more aggressive approach by navigating directly toward the target, whereas Agent2 should behave more humbly and continue exploring. In a parallel scenario, Agent3 exhibits similar aggressive behavior as Agent4. Therefore, Agent1 should maintain behavior consistent with Agent2 by also adopting a humble exploration behavior toward other goals. Consequently, it is intuitive to reward Agent1 for adopting the same behavior as Agent2 and penalize it for imitating the behaviors of Agent3 or Agent4.

In light of this observation, we consider exploring a multiagent system where each agent is incentivized to dynamically and selectively behave consistently with other agents. To formulate behavior consistency, inspired by agent play style diversity [21], we expose two agents to the same observation and assess the KL divergence of their predicted action distributions. Intuitively, the action distribution reflects an agent's intention. If two agents have consistent behavior, it suggests they share similar intentions when facing the same situation. We exploit such consistency in behavior as intrinsic rewards to encourage agents to selectively adhere to the behavior of other agents. However, deciding when to behave consistently with other agents is non-trivial. To address this challenge, we introduce a dynamic scaling network (DSN) that calculates learnable scale factors for each agent at every time step to dynamically guide whether to award consistent behavior and the magnitude of intrinsic rewards. The decent combination of behavior consistency and scale factor constitutes our proposed dynamic consistency-based intrinsic reward (DCIR). DCIR alleviates the dynamic behavior consistency issue in existing methods mentioned above by a large margin (i.e., from 75% to 88% in the consistent behavior case and from 69% to 97% in the inconsistent behavior case).

Our contributions are summarized as follows.

1) We study the MARL problem from the perspective of behavior consistency between agents. To this end, we propose to assess behavior consistency among multiple agents under the same environmental conditions, formulating it as the divergence in output action distributions between agents given the same observation.
2) We propose to design an intrinsic reward with dynamic consistency to evaluate the goodness of an agent's behavior. This reward dynamically encourages agents to behave consistently or inconsistently with each other, to maximize extrinsic return and thereby coordinate their behaviors.
3) Extensive experiments rigorously validate the superior efficacy of DCIR, demonstrating that DCIR outperforms five baseline methods on three multiagent benchmarks consistently, including both cooperative and competitive scenarios, within the Multiagent Particle environment [22], Google Research Football environment [23], and StarCraftII [24].

II. Related Works

A. Multiagent Reinforcement Learning

The multiagent environment, where multiple agents are present for interaction and action, is more typical in practical applications [25], [26], [27], [28], [29], [30], [31], [32] compared with the single-agent environment. However, MARL methods are more challenging due to the dependencies and conflicts between multiple agents as well as the dynamic and unstable environment [33], [34]. To address this issue, one prominent approach is independent reinforcement learning (IRL) [35], [36], where each agent treats other agents as part of the environment and learns its policy independently. Though simple and intuitive, IRL ignores the dependencies between agents, often leading to suboptimal outcomes. To capture interagent dependencies, centralized training and decentralized execution (CTDE) [22], [34], [37], [38], [39], [40], [41], [42] is an effective and widely applied framework. CTDE trains a centralized critic network that observes all agents' joint state and action information during training. During execution, each agent acts independently based on its own observations, leading to decentralized decision-making. CTDE has been widely used in MARL, such as RMIX [43], QMIX [44], COMA [45], VDN [46], and QTRAN [47]. In this work, we also follow the CTDE paradigm in our proposed method.

B. Behavior Modulation in Multiagent Systems

Recent work explores strategies to modulate agent behaviors in multiagent systems, aiming to improve coordination,

diversity, and exploration. ELIGN [11] promotes behavioral consistency by aligning agents with neighbors' expectations, enhancing system predictability. Ding et al. [48] disentangle probing and adaptation phases for rapid teammate adjustment, while Wang et al. [49] leverage multigraph neural networks to enrich local observations and team information. For diversity, LIIR [13] and CDS [19] assign intrinsic rewards to stimulate agents differently, and Li et al. [50] introduce subgroup-specific encoders via information bottlenecks, diversifying both objectives and architectures. Exploration-driven methods include Dong et al. [51], who propose a variational framework to trigger exploration, and approaches like Stadie et al. [15] and ActiveCamera Chen et al. [52], which reward agents for self-novelty, while Iqbal and Sha [16] incentivize exploration of states novel to the team. Zhaikhan and Sayed [20] use Q value variance among neighbors for efficient exploration, and Hou et al. [53] propose min–max intrinsic motivation to balance individual surprise and social influence. Adaptive and opponent-aware methods, such as GIIR [54], AIIR-MIX [55], and LAZIES [17], further refine intrinsic reward design by balancing cooperation, individuality, and opponent influence. However, these paradigms only enforce consistent or diverse behaviors within episodes, whereas real-world scenarios demand agents to autonomously decide when to act consistently; lacking such dynamic consistency impairs collaboration and adaptability.

III. Preliminaries

MARL problem can be formulated as a decentralized partially observable Markov decision process (DPOMDP): $<N, \mathcal{S}, \mathcal{O}, \mathcal{U}, \mathcal{T}, r_{\text{ex}}>$ [56] for N agents in an environment with state space $\mathcal{S}$. The observation space for the agents is $\mathcal{O} = \{\mathcal{O}^1, \ldots, \mathcal{O}^N\}$ and the action space is $\mathcal{U} = \{\mathcal{U}^1, \ldots, \mathcal{U}^N\}$. At time step t, each agent $i \in \{1, 2, \ldots, N\}$ obtains its own observation $o_t^i \in \mathcal{O}^i$ and performs an action $\boldsymbol{u}_t^i \in \mathcal{U}^i$ through the policy network $\boldsymbol{\pi}_i^{\theta_i} : \mathcal{O}^i \times \mathcal{U} \to [0, 1]$, which is parameterized by θ_i. For convenience, we omit θ_i in the subsequent description. The environment changes to the next state according to the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{U} \to \mathcal{S}$ with the current state and each agent's actions and returns a shared extrinsic reward $r_{\text{ex}} : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$. The learning objective of the multiagent problem is to learn the policy network $\boldsymbol{\pi}_i$ of each agent parameterized by θ_i to maximize the total expected return: $R = \sum_{t=0}^{T} \gamma_t r_{\text{ex}}$, where $\gamma \in [0, 1]$ is the discount factor. One of the challenges in MARL is the sparse and delayed extrinsic rewards, which make it difficult for agents to optimize the policies on the huge action space.

IV. Proposed Method

We seek to learn the agents to dynamically adjust their behaviors consistent with teammates via our proposed intrinsic reward. In this section, we present a formal description of DCIR that aims to encourage the agents to adaptively adjust their behavior based on behavior consistency with other agents for finishing multiagent tasks efficiently. We begin by defining behavior consistency between two agents. Next, we introduce the DCIR framework and its dynamic adjustment approach for each agent. Finally, we formulate an optimizing algorithm for the learning objective.

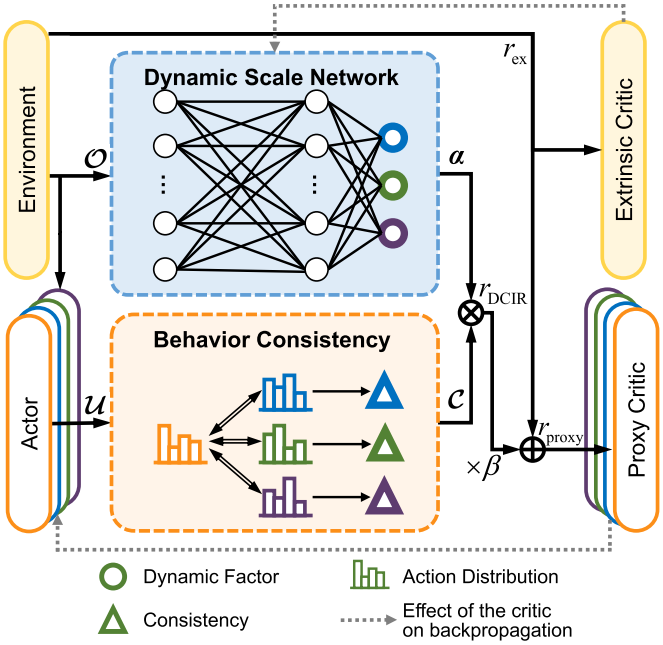

Fig. 2. Schematic of the overall DCIR framework. We adopt a bi-level actor–critic framework. Each agent has its own actor and proxy critic (denoted by different colors). The centralized extrinsic critic is used for updating the proposed DSN. Taking the **orange agent** as an example, at each time step, the Behavior Consistency module calculates its behavior consistency $\mathcal{C}$ with the other agents. The DSN outputs the dynamic adjustment factor $\boldsymbol{\alpha}$. Then, the intrinsic reward r_{DCIR} is obtained from multiplying the $\mathcal{C}$ and $\boldsymbol{\alpha}$. Last, r_{DCIR} is added to the extrinsic reward r_{ex} for proxy value.

A. Behavior Consistency Between Agents

In evaluating the behavior consistency between two agents, our approach involves examining their respective action outputs when confronted with similar states. As illustrated in Fig. 2, behavior consistency is defined as the degree of similarity in the distribution of actions for two agents given an identical observation. Specifically, each agent i has its own observation denoted by o_t^i at time step t. Since the action space size of each agent is represented as n, we denote the action distribution of policy of Agent i as $\boldsymbol{u}_t^i = \boldsymbol{\pi}_i(o_t^i) = \{u_{t,1}^i, \ldots, u_{t,n}^i\}$. The behavior consistency between agent i and its cooperative agent j is defined by the following KL divergence:

$$\mathcal{C}_{i,j,t} = \sum_{k=1}^{n} u_{t,k}^{i,j} \log \frac{u_{t,k}^{i,j}}{u_{t,k}^{i}}. \tag{1}$$

Here, $u_{t,k}^{i,j}$ represents the probability of agent j selecting action k when it observes the same observation as agent i (i.e., o_t^i) at time step t. A larger KL divergence (i.e., a larger $\mathcal{C}$) indicates a greater difference in the behaviors taken by the two agents. On the other hand, a smaller KL divergence (i.e., a smaller $\mathcal{C}$) indicates more consistency in the behaviors taken by the two agents. Note that in a continuous action setting, we can still perform KL divergence calculations by replacing summation with an integral.

B. Intrinsic Reward for Dynamic Consistency

Building upon the above analyses, a successful collaboration requires each agent to dynamically behave consistently

or inconsistently with other agents. Ideally, the environment should assign a high reward to the agent i when its optimal behavior for the task at time step t is a consistent behavior w.r.t. to agent j. Therefore, an agent's reward at each step is directly related to the difference between its behavior and the behavior of other agents, which is represented by (1). To promote the adaptive adjustment of the dominance of consistent and diverse behaviors based on collaboration progress with agent j, we propose dynamic consistency-based intrinsic reward (DCIR) for the ith agent as follows:

$$r_{\text{DCIR}}^{i,t} = \sum_{j \in \mathcal{N}(i)} \alpha_j^i \times \mathcal{C}_{i,j,t} \tag{2}$$

where α_j^i is a value that can be negative to encourage consistent behavior between agent i and agent j, and be positive α_j^i to reward inconsistent behavior. It is viewed as an instance of meta learning [57], [58] to indirectly maximize extrinsic returns by modulating how policy adjustments propagate to reward outcomes. This approach quantifies how strongly agents prioritize behavioral predictability over adaptive collaboration in decentralized decision-making.

C. Dynamic Scale Network for Reward Learning

As aforementioned, factor α is expected to dynamically adjust the intrinsic reward. To achieve this goal, we propose a dynamic scale network (DSN) to parameterize the learnable α, as shown in Fig. 2. The DSN is modeled by a multilayer perceptron (MLP) with layers, each with ReLU activations except the last layer. For a multiagent system with N agents, each agent has its own DSN which outputs a vector of length $(N-1)$ for the behavior consistent with other $(N-1)$ agents, respectively. Formally, we compute

$$\boldsymbol{\alpha}^i = \text{DSN}\left(o^1, \ldots, o^N\right) \tag{3}$$

where $\boldsymbol{\alpha}^i = \{\alpha_j^i \mid j \in \mathcal{N}(i)\}$ and $\mathcal{N}(i)$ denotes the set of agents excluding agent i. As illustrated in Fig. 2, the output $\boldsymbol{\alpha}$ is used to multiply the behavior consistency $\mathcal{C}$ for DCIR in (2).

D. Learning With Soft Actor–Critic

1) Soft Actor–Critic as Learning Paradigm: As illustrated in Fig. 2, we adopt a bi-level actor–critic framework to optimize the agent policies. Each agent has its own actor and proxy critic (denoted by different colors). Soft actor–critic (SAC) algorithm [59] serves as the policy optimization method. SAC is an off-policy RL algorithm with the actor–critic framework. It combines maximizing entropy learning to bring better exploration ability. During the training, the policy parameter θ_i of agent i is optimized by minimizing the objective J_{θ^i}

$$J_{\theta^i} = \mathop{\mathbb{E}}_{o_t^i \sim D^i} \left[\mathop{\mathbb{E}}_{\boldsymbol{u}_t^i \sim \boldsymbol{\pi}_{\theta_i}(\cdot|_t)} \left[\omega \log \boldsymbol{\pi}_{\theta_i}\left(\boldsymbol{u}_t^i \mid o_t^i\right) - \min_{\phi_i} Q_{\phi_i}^{\text{Soft}}\left(o_t^i, \boldsymbol{u}_t^i\right) \right] \right] \tag{4}$$

where o_t^i is the observation of agent i uniformly sampled from the replay buffer D^i at time step t; $\boldsymbol{\pi}_{\theta_i}(o_t)$ is the predicted action probability distribution by policy $\boldsymbol{\pi}_{\theta_i}$ input by observation o_t; and ω is the entropy temperature coefficient. It determines the importance of entropy maximization in the objective function; $Q_{\phi_i}^{\text{Soft}}(o_t^i, \boldsymbol{u}_t^i)$ is the output action value of the soft value function with parameter ϕ_i. In order to alleviate the overestimating problem, two value networks with the same structure are constructed, and the smaller output value of the two networks is chosen during training, that is,

$$Q_{\phi_i}^{\text{Soft}}\left(o_t^i, \boldsymbol{u}_t^i\right) = \min\left\{Q_{\phi_i^1}^1\left(o_t^i, \boldsymbol{u}_t^i\right), Q_{\phi_i^2}^2\left(o_t^i, \boldsymbol{u}_t^i\right)\right\}. \tag{5}$$

Note that our DCIR can apply to any MARL paradigm besides SAC.

Algorithm 1 Training Paradigm for DCIR

Require: The reply buffer $\mathcal{D}$, the policy of the agent $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, the Dynamic Scale Network $\text{DSN}_{\boldsymbol{\eta}}$.

Initialize the N agent parameters $\boldsymbol{\theta}$ and DSN parameters $\boldsymbol{\eta}$ randomly.
Initialize the replay buffer $\mathcal{D}$.
while not converge **do**
  Collect samples $\left\{o_t^i, o_{t+1}^i, r_{\text{ex}}^t, \boldsymbol{u}_t^i, \left\{\boldsymbol{u}_t^{i,j} \mid j \in \mathcal{N}(i)\right\}\right\}$ from environments and populate them into $\mathcal{D}$.
  for agent $i = 1, \ldots, N$ **do**
    Uniformly sample $\left\{o_t^i, o_{t+1}^i, r_{\text{ex}}^t, \boldsymbol{u}_t^i, \left\{\boldsymbol{u}_t^{i,j} \mid j \in \mathcal{N}(i)\right\}\right\}$ from $\mathcal{D}$.
    Compute **Behavior Consistency** $\mathcal{C}$ by Equation (1).
    Combine scale factor α ouput from **Dynamic Scale Network** with $\mathcal{C}$ to compute **DCIR** $r_{\text{DCIR}}^{i,t}$ according to Equation (2) and Equation (3).
    Calculate proxy reward $r_{\text{proxy}}^{i,t}$ via Equation (6).
    Update the sample to $\left\{o_t^i, o_{t+1}^i, r_{\text{ex}}^t, r_{\text{proxy}}^{i,t}, \boldsymbol{u}_t^i, \left\{\boldsymbol{u}_t^{i,j} \mid j \in \mathcal{N}(i)\right\}\right\}$.
    Update θ_i using Soft Actor-Critic algorithm.
    Update η_i using the extrinsic actor-critic optimization method based on the chain derivation rule.
    Update i^{th} proxy critic using the $r_{\text{proxy}}^{i,t}$ and the estimated value of the next state by TD learning.
  end for
  Update extrinsic critic using the r_{ex}^t and the estimated value of the next state by TD learning.
end while

2) Learning Objectives: For agent i at each time step t, we calculate the DCIR using (2) and obtain a proxy reward by combining the extrinsic reward (r_{ex}^t) with $r_{\text{DCIR}}^{i,t}$

$$r_{\text{proxy}}^{i,t} = r_{\text{ex}}^t + \beta \times r_{\text{DCIR}}^{i,t} \tag{6}$$

where β is a scaling hyper-parameter to constrain the range of two rewards at the beginning, providing a better initialization and ensuring that the two rewards are roughly of the same order of magnitude. This is a common practice in training, which will make the network converge and learn more easily. Here, as outlined problem definition, the extrinsic reward r_{ex}^t is sparse and delayed since the environment feedbacks it regarding the completion of the task. The policies are optimized by SAC to maximize $r_{\text{proxy}}^{i,t}$. Then, $r_{\text{proxy}}^{i,t}$ is used to update the proxy critic via temporal difference (TD) [60] learning. The updated proxy critic outputs the proxy Q value used to modify the actor parameters of the agent. An overview of the optimization process is presented in Algorithm 1.

To maximize the standard accumulated discounted team return from the environment, we adopt the learning paradigm from LIIR [13] to train the DSN parameters using the extrinsic-level actor–critic framework. This framework shares the same actors as the proxy actor–critic framework but maintains a centralized extrinsic critic. We utilize the chain rule to connect the impact of the DSN parameter (i.e., η_i) changes on the objective J^{ex} of the extrinsic-level actor–critic in the updated actor parameter (i.e., θ_i') as

$$\nabla_{\eta_i} J^{\text{ex}} = \nabla_{\theta_i'} J^{\text{ex}} \nabla_{\eta_i} \theta_i'. \tag{7}$$

We detail the optimizing method of DSN in Section IV-E.

E. Optimizing Method of DSN

The dynamic scale network (DSN) is parameterized by η. As aforementioned, each agent has its own DSN to output the dynamic scaling factor α for behavior consistency with other team agents. The parameters of DSN should be optimized in the direction of increasing extrinsic rewards. To achieve this goal, we adopt an extra-level actor–critic framework as in LIIR [13].

To streamline our analysis, we represent the policy as $\boldsymbol{\pi}_{\theta_i}$ with parameters θ_i for agent i. We adopt the policy gradiant [61] method as the objective function of the policy in an extra-level actor–critic framework

$$J^{\text{ex}} = \mathbb{E}_{o_t, \boldsymbol{u}_t \sim \mathcal{D}} \left[\log \boldsymbol{\pi}_{\theta_i} \left(\boldsymbol{u}_t^i \mid o_t^i \right) A^{\text{ex}} \left(o_t, \boldsymbol{u}_t \right) \right]. \tag{8}$$

Here, the advantage A^{ex} is estimated as $A^{\text{ex}}(o_t, \boldsymbol{u}_t) = r_{\text{ex}}(o_t, \boldsymbol{u}_t) + V^{\text{ex}}(o_{t+1}) - V^{\text{ex}}(o_t)$ following [13], [62], [63], where o_t is the vector that involves all the observations of the agents at time step t. We denoted V^{ex} as the extrinsic value which is estimated by the extrinsic critic and o_{t+1} as the next successive observations of the agents.

Given the updated policy $\boldsymbol{\pi}_{\theta_i'}$ with parameters θ_i' of agent i and its own DSN (i.e., DSN_{η_i}), we use the chain rule to build the connection between η_i and J^{ex} as follows [13]:

$$\nabla_{\eta_i} J^{\text{ex}} = \nabla_{\theta_i'} J^{\text{ex}} \nabla_{\eta_i} \theta_i' \tag{9}$$

where the first term $\nabla_{\theta_i'} J^{\text{ex}}$ is formulated as

$$\nabla_{\theta_i'} \log \boldsymbol{\pi}_{\theta_i'} \left(\boldsymbol{u}_t^i \mid o_t^i \right) A^{\text{ex}} \left(o_t, \boldsymbol{u}_t \right). \tag{10}$$

Here, we reuse the samples generated by θ_i with the importance sampling method [64].

As aforementioned, the updated parameter θ_i' comes from the following stochastic gradient method:

$$\begin{aligned} &\theta_i - \xi \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \Big[\nabla_{\theta_i} \mathbb{E}_{\boldsymbol{u}_t^i \sim \boldsymbol{\pi}_{\theta_i}(\cdot \mid o_t^i)} \big[\omega \log \boldsymbol{\pi}_{\theta_i} \left(\boldsymbol{u}_t^i \mid o_t^i \right) \\ &\qquad - \min_{\theta_i} Q_{\phi_i}^{\text{Soft}} \left(o_t^i, \boldsymbol{u}_t^i \right) \big] \Big] \\ &\approx \theta_i - \xi \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \Big[\nabla_{\theta_i} \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top \big[\omega \log \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right) \\ &\qquad - Q_{\phi_i'}^{\text{Soft}} \left(o_t^i \right) \big] \Big] \end{aligned} \tag{11}$$

where ξ is the learning rate, and ω is the entropy temperature coefficient.

To this end, the second term $\nabla_{\eta_i} \theta_i'$ is computed as

$$\begin{aligned} &\nabla_{\eta_i} \theta_i' \\ &= \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\eta_i} \Big[\theta_i - \xi \nabla_{\theta_i} \Big[\omega \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top \log \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right) \\ &\qquad - \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top Q_{\phi_i'}^{\text{Soft}} \left(o_t^i \right) \Big] \Big] \\ &= \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\eta_i} \Big[-\xi \nabla_{\theta_i} \omega \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top \log \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right) \\ &\qquad + \xi \nabla_{\theta_i} \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top Q_{\phi_i'}^{\text{Soft}} \left(o_t^i \right) \Big] \\ &= \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\eta_i} \Big[\xi \nabla_{\theta_i} \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top Q_{\phi_i'}^{\text{Soft}} \left(o_t^i \right) \Big] \\ &= \xi \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\theta_i} \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top \nabla_{\eta_i} Q_{\phi_i'}^{\text{Soft}} \left(o_t^i \right) \\ &= \xi \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\theta_i} \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top \nabla_{\phi_i'} Q_{\phi_i'}^{\text{Soft}} \left(o_t^i \right) \nabla_{\eta_i} \phi_i'. \end{aligned} \tag{12}$$

The parameter of proxy critic i is derived as

$$\begin{aligned} &\nabla_{\eta_i} \phi_i' \\ &= \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\eta_i} \Big[\phi_i - \xi \nabla_{\phi_i} \left(Q_{\phi_i}^{\text{Soft}} \left(o_t^i \right) - Q_{\text{target}} \right)^2 \Big] \\ &= -\mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\eta_i} \left[2 \left(Q_{\phi_i}^{\text{Soft}} \left(o_t^i \right) - Q_{\text{target}} \right) \xi \nabla_{\phi_i} Q_{\phi_i}^{\text{Soft}} \left(o_t^i \right) \right] \\ &= -2\xi \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\eta_i} \left(Q_{\phi_i}^{\text{Soft}} \left(o_t^i \right) - Q_{\text{target}} \right) \nabla_{\phi_i} Q_{\phi_i}^{\text{Soft}} \left(o_t^i \right) \\ &= 2\xi \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\eta_i} Q_{\text{target}} \nabla_{\phi_i} Q_{\phi_i}^{\text{Soft}} \left(o_t^i \right) \end{aligned} \tag{13}$$

where $Q_{\text{target}} = r_{\text{ex}}^t + \beta \times r_{\text{DCIR}}^{i,t} + \mathbb{E}_{u_{t+1}^i \sim \boldsymbol{\pi}_{\theta_i'}} [Q_{\phi_i'}^{\text{Soft}}(o_{t+1}^i) - \omega \log(\boldsymbol{\pi}_{\theta_i'}(u_{t+1}^i \mid o_{t+1}^i))]$.

Therefore,

$$\nabla_{\eta_i} \phi_i' = 2\beta\xi \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\eta_i} r_{\text{DCIR}}^{i,t} \nabla_{\phi_i} Q_{\phi_i}^{\text{Soft}} \left(o_t^i \right). \tag{14}$$

Hence, we have

$$\begin{aligned} &\nabla_{\eta_i} \theta_i' \\ &= \xi \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\theta_i} \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top \nabla_{\phi_i'} Q_{\phi_i'}^{\text{Soft}} \left(o_t^i \right) \nabla_{\eta} \phi_i' \\ &= 2\beta\xi^2 \mathbb{E}_{o_t^i \sim \mathcal{D}^i} \nabla_{\theta_i} \boldsymbol{\pi}_{\theta_i} \left(o_t^i \right)^\top \nabla_{\phi_i'} Q_{\phi_i'}^{\text{Soft}} \left(o_t^i \right) \nabla_{\eta_i} r_{\text{DCIR}}^{i,t} \\ &\quad \times \nabla_{\phi_i} Q_{\phi_i}^{\text{Soft}} \left(o_t^i \right) \end{aligned} \tag{15}$$

where the $r_{\text{DCIR}}^{i,t} = \sum_{j \in \mathcal{N}(i)} \alpha_j^i \times \mathcal{C}_{i,j,t}$ and the α_j^i is parameterized by η_i. In this way, we can connect the parameter updating of DSN with the objective J^{ex} to ensure the dynamic scaling factor α is optimized in the direction of increasing extrinsic rewards.

V. Experiment

A. Environments

To evaluate the efficacy of DCIR, we benchmark our method on three popular MARL benchmark environments: Multiagent Particle environment (MPE) [22], Google Research Football [23], and StarCraft II Micromanagement [24].

1) Multiagent Particle Environment (MPE): In this environment, multiple particle agents can move, communicate, observe each other, push other agents, and interact with fixed landmarks inhabiting a 2-D world. The action space for each agent is discrete, i.e., stay or change the velocities of four cardinal directions, while the observation space of each agent is continuous. We adopt a partially observable setting, wherein each agent possesses an observation radius. These particles can perceive the positions and velocities of other agents within the radius, as well as the positions of landmarks falling within the same range.

2) Google Research Football (GRF): Google Research Football (GRF) provides a simulated football environment for training multiagent players to score against adversary players in a 3-D world. The discrete action set consists of 19 actions, including one idle action, eight movement actions, four passing/shooting actions, and six other actions. Each football play is controlled by an agent, observing the ball information, team players' information, adversary players' information, controlled player information, and game mode information.

3) StarCraft II Micromanagement: StarCraft II Micromanagement is a strategy game where each agent has various attributes such as health points, weapon cooldown, unit type, last action, and distance to observed units. Agents partially observe units within their view range, i.e., a circular radius. The action space includes four move directions, a maximum of k attack actions against enemy units, stop, and no operation.

B. Tasks for Demonstration

We evaluate the proposed DCIR on seven multiagent tasks based on the MPE, GRF, and StarCraft II Micromanagement. In order to more comprehensively verify the effectiveness of DCIR, we consider both cooperative and competitive tasks. Note that these tasks are with shared extrinsic task completion rewards only by default.

1) Cooperative Navigation (Coop Nav.): This task is conducted in MPE. In this task, N agents are required to cooperate to reach N goals as fast as possible and are collectively rewarded based on how many goals are occupied.

2) Heterogeneous Navigation (Hetero Nav.): There are N goals that N agents cooperate to reach in MPE. The speeds and sizes of N agents are different. Half of the agents are slow and big, while the other half are fast and small.

3) Physical Deception (Phy Decep.): N agents are rewarded if one of them reaches the goal but penalized if one of their $N/2$ adversaries occupies it. The goal is hidden among N landmarks, known only to agents, not adversaries. Agents must learn to deceive adversaries by covering all the landmarks. This task is based on MPE.

4) Predator-Prey (Pred-Prey): In a randomly generated obstacle-filled environment, N slow adversaries pursue and capture N fast cooperating agents. When an adversary catches an agent, the agent is penalized, and the adversary is rewarded. This task is based on MPE.

5) Keep-Away: In this MPE task, there are N agents and N landmarks. One of the landmarks is the goal, known to the agents. M adversaries are present, and the agents are rewarded for pushing them away from the goal. Adversaries can only deduce the goal based on the agents' behavior.

6) Academy 3 Versus 1 With Keeper (3v1 w/ keeper): This task is undertaken within the GRF environment. Specifically, three agents are assigned to control three individual players. Their objective is to successfully navigate past a defender and a goalkeeper in order to propel the ball into the goal and score.

7) Marines Versus 3 Marines: We explore symmetric battle scenarios in StarCraft II involving 3 Marines on each side, i.e., *3 Marines versus 3 Marines* (3M). In each time step, the agents receive a joint team reward based on their inflicted damage and the damage received from the enemy.

C. Experimental Setups

We implement the experiments in MPE by the tianshou framework [65], the GRF environment by the rllib framework [66], and StarCraft II Micromanagement by the SMAC framework [24]. For a fair comparison, we employ the same experiment setting in MPE and GRF as ELIGN and in StarCraft II Micromanagement as LIIR. Specifically, for MPE, we train the agents for 100-k time steps per epoch over 100 epochs. For GRF, the agents are trained for a total of 5-M time steps, and for StarCraft II Micromanagement, they are trained for 3-M time steps. Here, a time step refers to the process in which each agent performs one action, leading to a change in the environment's state and the generation of one extrinsic reward. Each experiment runs on one NVIDIA A800 GPU. Note that both the dynamic scale network and behavior consistency computation are only involved during the training phase. During testing, only the learned actor networks are executed, without any additional computation or modules.

D. Baselines

Our overall main focus is on evaluating the effectiveness of DCIR in intrinsic reward systems. Toward this goal, we choose five popular and solid baselines under the same setting as our method for a fair and comprehensive evaluation.

1) SPARSE [22]: This SPARSE baseline learns the policy using only sparse extrinsic rewards returned by the environment.

2) EXP-Self [15]: This baseline encourages the agent to explore states that it has not explored before. The intrinsic reward is high for exploring more novel states. It is a classic exploration-based intrinsic reward method in the single-agent domain.

3) EXP-Team [16]: In addition to only rewarding its exploration of states that are novel to itself, the EXP-team also rewards each agent for simultaneously exploring states that are also novel to the rest of the team agents. The rewards are also provided in the form of intrinsic rewards.

4) ELIGN [11]: ELIGN proposes to use intrinsic rewards to reward each agent for making decisions that conform to the predictions of other agents, ensuring that the behavior of each agent is predictable.

5) LIIR [13]: LIIR proposes to motivate agents to perform diverse behaviors by using learnable intrinsic rewards without specific practical meanings.

E. Comparison Results

1) Results on Cooperative Tasks: In Table I, the policy learned with our DCIR beats all the baselines on both *Coop Nav.* task and *Hetero Nav.* task. On *Coop Nav.* task, compared to the exploration-based methods EXP-self and EXP-team, the improvement can be attributed to that instead of blindly pursuing novel states, we encourage agents to selectively adopt exploration behaviors. DCIR also beat ELIGN. ELIGN encourages the behaviors of agents to be predictable and aligned with each other. When the behaviors of all agents become easy to predict, they also tend to be the same. This limits the agents to dynamically adjust intentions according to the current state. Different from ELIGN, our method does not simply request all the agents acting consistently but considers

TABLE I

MEAN TEST EPISODE EXTRINSIC REWARDS AND STANDARD ERRORS ACROSS DIFFERENT METHODS. WE TRAIN ALL ALGORITHMS WITH 5 RANDOM SEEDS FOLLOWING ELIGN. THE NUMBERS IN PARENTHESES REPRESENT THE NUMBER OF AGENTS AND ADVERSARIES, I.E., (AGT # VERSUS ADV #)

Methods	MPE (Cooperative)		MPE (Competitive)			GRF (Competitive)
	Coop Nav. (5v0)	Hetero Nav. (6v0)	Phy Decep. (4v2)	Pred-prey (4v4)	Keep-away (4v4)	3v1 w/ keeper (3v2)
SPARSE	459.92 ± 22.44	616.62 ± 25.30	166.89 ± 27.72	-28.75 ± 7.3	0.752 ± 1.82	0.020 ± 0.001
EXP-self	458.45 ± 19.79	702.73 ± 18.57	146.55 ± 29.05	-25.35 ± 6.16	10.52 ± 5.48	0.024 ± 0.004
EXP-team	497.15 ± 11.47	695.38 ± 12.22	84.66 ± 16.94	-17.21 ± 8.23	1.40 ± 2.06	0.021 ± 0.002
ELIGN	498.24 ± 9.77	646.70 ± 23.25	186.83 ± 21.92	-9.14 ± 5.57	11.29 ± 9.02	0.025 ± 0.001
LIIR	495.50 ± 8.40	660.92 ± 15.71	179.27 ± 21.74	-49.79 ± 11.57	3.45 ± 13.53	0.027 ± 0.003
DCIR (Ours)	$\mathbf{525.92 \pm 8.99}$	$\mathbf{707.52 \pm 16.70}$	$\mathbf{224.49 \pm 15.39}$	$\mathbf{-7.91 \pm 1.88}$	$\mathbf{21.45 \pm 9.75}$	$\mathbf{0.031 \pm 0.002}$

both behaviors dynamically conditioned on the states. This is more obvious in the *Hetero Nav.*, where the agents need more moments of inconsistent behavior because of differences in their properties. Compared to LIIR, DCIR better promotes collaboration and coordination among multiple agents instead of focusing on agents' own learning of an intrinsic reward. Moreover, LIIR directly parameterizes the intrinsic reward without any actual meaning or guidance to learn the rewards, while our DCIR is able to encourage inconsistent behavior by positive values and encourage inconsistent behavior by negative values.

2) Results on Competitive Tasks: We further demonstrate the effectiveness of our proposed DCIR for tasks with adversaries. The results are shown in Table I and Fig. 3. We noticed that DCIR offers a considerable boost (89.99%) over the SoTA baseline in the *Keep-away.* task. We speculate that the role division is more obvious in this task, e.g., deceiving and pushing, which has high requirements for behavior consistency between agents and DCIR can alleviate this. Also in the *3v1 w/ keeper.* task, while the baselines have comparable performance, our DCIR can boost the performance by 24%. This can be explained by the complexity arising from intricate rules, and interferences among defenders and keepers, beyond the scope of mere exploration (EXP-self and EXP-team) or alignment (ELIGN) strategies. In addition, the vast action and state possibilities hinder direct learning of a feasible intrinsic reward (LIIR). Test winning rates in Fig. 3 on StarCraft II Micromanagement further show consistent improvement with faster convergence and higher win rate, demonstrating the efficacy of DCIR in a strategy competitive MARL task.

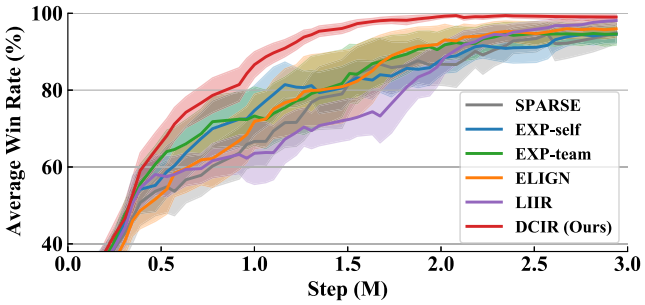

Fig. 3. Average test winning rates versus training steps of various methods on the 3M task in StarCraft II. Our training and testing settings remain the same as in the LIIR [13], and 5 seeds are selected for plotting.

F. Ablation Study

In this section, we proceed to ablate DCIR under MPE on both cooperative task and competitive task, i.e., *Cooperative Navigation* and *Physical Deception*, respectively.

1) Divergence Alternatives: To quantify behavior consistency between agents, we analyze four divergence measures between action probability distributions given identical observations. Our primary analysis employs Kullback–Leibler (KL) divergence due to its information-theoretic properties and sensitivity to distributional differences. We systematically compare against three alternatives.

1) *Binary Divergence:* Assigns +1 for identical actions, −1 otherwise.
2) *Jensen–Shannon (JS) Divergence:* Symmetric measure based on KL divergence [67].
3) *Total Variation (TV) Distance:* L1-norm between distributions [68].

Table II demonstrates KL divergence's superior empirical performance. We theorize this stems from three fundamental properties.

1) *Asymmetric Sensitivity:* KL's asymmetry ($D_{KL}(p\|q) \neq D_{KL}(q\|p)$) aligns with our need to distinguish reference policy (p) from agent policy (q).
2) *Logarithmic Scaling:* KL's $\log(p/q)$ formulation amplifies small probabilistic differences critical for behavior consistency.
3) *Unbounded Nature:* Unlike bounded measures (JS $\in [0, 1]$, TV $\in [0, 2]$), KL's unboundedness better discriminates between near-deterministic policies.

While JS divergence provides symmetry, its midpoint averaging ($\mathrm{M} = (1/2)(P + Q)$ where P means the reference policy distribution, Q means the agent policy distribution, and M is the midpoint mixture in JS divergence) introduces artificial mixture distributions absent in our problem domain. TV distance's L1-norm proves less suitable as it weights all distribution differences equally, unlike KL's emphasis on low-probability tail events through logarithmic scaling. The binary baseline's poor performance confirms our hypothesis that discrete thresholding discards crucial probabilistic information.

2) Effectiveness of DSN: In probing the contributions of DSN, we replace the output of the dynamically adjustable factor α output from DSN with four variants. We first replace α with +1, constructing a variant that encourages an agent to behave inconsistently with any other agent. We thus name it `Inconsistency`. On the contrary, we replace α with −1 to encourage consistent behaviors between agents, and we

TABLE II

ABLATION STUDY ON ALTERNATIVES OF KL DIVERGENCE. WE EXHIBIT THE MEAN TEST EPISODE EXTRINSIC REWARDS AND STANDARD ERRORS ACROSS USING DIFFERENT ALTERNATIVES

Distance Type	Cooperative	Competitive
	Coop Nav. (5v0)	Phy Decep. (4v2)
Binary Divergence	504.60 ± 7.56	205.08 ± 14.46
JS Divergence	516.76 ± 3.98	220.57 ± 13.17
TV Distance	508.27 ± 7.52	214.64 ± 19.43
KL Divergence (Ours)	$\mathbf{525.92 \pm 8.99}$	$\mathbf{224.49 \pm 15.39}$

TABLE III

ABLATION STUDY ON THE EFFECTIVENESS OF DSN. WE EXHIBIT THE MEAN TEST EPISODE EXTRINSIC REWARDS AND STANDARD ERRORS ACROSS USING DIFFERENT ALTERNATIVES

Factor Type	Cooperative	Competitive
	Coop Nav. (5v0)	Phy Decep. (4v2)
Inconsistency	491.51 ± 12.16	197.46 ± 31.92
Consistency	506.45 ± 8.12	206.04 ± 19.93
Shared Factor	511.18 ± 25.13	205.90 ± 21.81
Learnable Paras.	505.29 ± 12.63	208.85 ± 17.20
No Guidance	498.62 ± 5.93	182.68 ± 19.53
DSN (Ours)	$\mathbf{525.92 \pm 8.99}$	$\mathbf{224.49 \pm 15.39}$

denote this variant as `Consistency`. Then, to verify that each agent needs to maintain different behavior consistency with different agents, we design a variant (i.e., `Shared Factor`). The DSN of each agent in this variant only outputs a shared α for the team agents. Besides, we replace α with no-input learnable parameters to justify the necessity of DSN design (i.e., `Learnable Paras.`). Last, to inspect the importance of behavior consistency as the guidance in DCIR, we directly use the α as the intrinsic reward, i.e., the variant `No Guidance`. As illustrated in Table III, either awarding the agents to perform only inconsistent behaviors or consistent behaviors drops the performance. By adding our DSN to learn the dynamic adjustable factor α, the agent can learn to tackle different requirements for behavior consistency under each task. Compared to `Shared Factor`, learning separate α significantly improves the performances. This is because the team agents vary in their policy learning levels and behavior intentions. Therefore, different α needs to be used to encourage different behavior consistency with other team agents. Moreover, the lack of necessary input information and network complexity makes the learnable parameters difficult to distinguish task situations and adjust appropriately. `Learnable Paras.` thereby performs worse than DSN. We also observe that `No Guidance` is not as good as DCIR. We suspect that evaluating the quality of an agent's behavior requires looking at how well it cooperates with other agents rather than just focusing on itself, while a single scalar output is hard to model the relationship among the agents without explicit optimizing guidance and direction.

3) Scalability of DCIR: In this experiment, we investigate the scalability of DCIR when more agents are added to cooperative and competitive tasks. As shown in Table IV, when the number of agents increases, DCIR enjoys superior performance over the five baselines both in the cooperative task and the competitive task. This highlights that DCIR is feasible and effective in the case of a larger number of agents. Besides the results shown in Table IV, we also increase the number of agents to 20 in Coop Nav., and the result demonstrates that DCIR robustly scales to 20 agents, significantly surpassing the baseline ELIGN (with extrinsic reward 4597.65 versus 4128.78). Furthermore, as shown in Fig. 3, in contrast to MPE and GRF benchmarks that use SAC, we utilize the ac [69] algorithm in StarCraft II Micromanagement following LIIR. In Fig. 4, we also incorporate our DCIR into advanced MARL algorithms, i.e., MAPPO [30] and JRPO [31], respectively, in StarCraft II Micromanagement. Consistent improvement highlights the generalization ability and scalability of DCIR.

4) Ablation of Hyperparameter β: The hyperparameter β in 6 is set to 2.5×10^{-2} by default in MPE. We choose it by considering the scale range of DCIR and extrinsic rewards in the initial stage of training to accelerate the convergence of the agent. Below, we conduct an ablation experiment to show how the performance of the Coop Nav (5 v0) task changes with different sizes of β. The results are shown in Table V. From the comparison of the results in the table, any increase or decrease in the β will decrease the task performance. We speculate that at the beginning of training, a suitable weighting factor (i.e., β) can make extrinsic rewards and intrinsic rewards have relatively equal importance, and can accelerate the network's learning speed for both rewards.

G. Symmetry-Breaking Experiments

In this section, we study the challenges of efficiently allocating the subgoals in multiagent collaboration [70]. We consider the symmetry-breaking setting as in ELIGN [11]. Specifically, for *Coop Nav.* and *Hetero Nav.*, we initialize all the agents in the same position and draw the goals randomly around the agents on a circle perimeter with a certain radius, which is equal to the value of the world radius minus the greatest goal size. As for *Phy Decep.*, we place both agents and adversaries at the origin and the landmarks randomly around the agents on a circle perimeter with a certain radius. In *Pred-prey.* task, we initialize the agents in the same position while the adversaries are put randomly in a circle. The agents and adversaries in *Keep-away.* are placed in the same way as in *Pred-prey.*, and the landmarks are initialized on a circle perimeter. Symmetry-breaking setting raises a challenge for the efficient goal allocation of the agents without supervision.

In Table VI, we show the test results under *Symmetry-Breaking* setting in the Multiagent Particle environment. The results show that DCIR significantly beats the baselines, suggesting that DCIR helps the agents to break the deadlock of the same initial position. Through behavior consistency coordination among the agents, each agent can find its own optimal subgoal, thus completing the task efficiently. We also notice that DCIR does not outperform all the baselines in *Phy Decep.*. We hypothesize that this is because this task starts with both the agent and the adversaries in the same position, leading to a phase where their observations align. During this, the agents not only need to perform the dynamic behavior consistency with teammates to occupy the goals but also to deceive adversaries without confusing teammates. This enormously increases the difficulty of behavior consistency decision-making.

TABLE IV
PERFORMANCE WHEN THE NUMBER OF AGENTS INCREASES. WE EXHIBIT THE MEAN TEST EPISODE EXTRINSIC REWARDS AND STANDARD ERRORS ACROSS DIFFERENT METHODS

Method	Cooperative		Competitive	
	Coop Nav. (5v0)	Coop Nav. (10v0)	Phy Decep. (4v2)	Phy Decep. (8v4)
SPARSE	459.92 ± 22.44	1103.87 ± 22.89	166.89 ± 27.72	563.42 ± 70.52
EXP-self	458.45 ± 19.79	1101.15 ± 42.53	146.55 ± 29.05	334.71 ± 97.83
EXP-team	497.15 ± 11.47	1106.48 ± 33.14	84.66 ± 16.94	351.02 ± 84.01
ELIGN	498.24 ± 9.77	1088.42 ± 39.4	186.83 ± 21.92	534.91 ± 48.05
LIIR	495.50 ± 8.40	1003.45 ± 44.1	179.27 ± 21.74	586.83 ± 141.33
DCIR (Ours)	**525.92 ± 8.99**	**1137.48 ± 47.21**	**224.49 ± 15.39**	**622.88 ± 82.96**

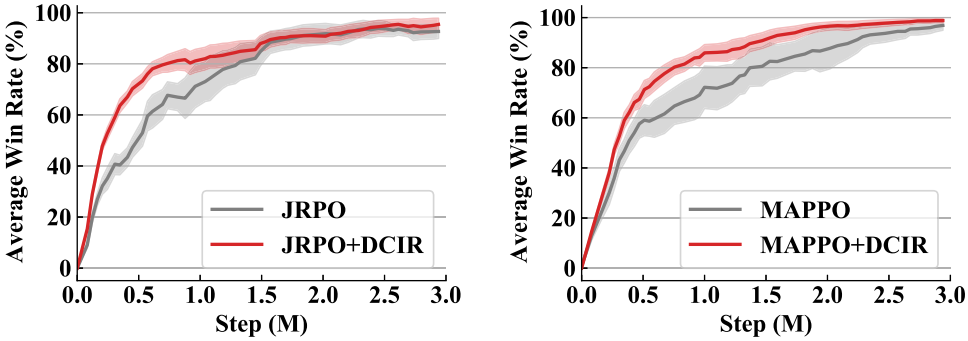

Fig. 4. Average test winning rates versus training steps of incorporating DCIR into MAPPO and JRPO on the 3M task in StarCraft II. Our training and testing settings remain the same as in the LIIR [13], and 5 seeds are selected for plotting.

TABLE V
ABLATION ON THE VALUE OF HYPERPARAMETER β IN 6

β	2.5×10^{-3}	2.5×10^{-2}	2.5×10^{-1}
Reward	504.46 ± 5.70	**525.92 ± 8.99**	501.83 ± 7.27

H. Pilot Study

To inspect how severe the previous methods suffer in the dynamic behavior consistency problem and how much DCIR alleviates this problem, we conduct two pilot studies. We place 5 agents at the origin in MPE. In **Study 1**, we position one target at the origin, while the remaining 4 are randomly distributed around a circle centered at the origin, all within the observable range of agents. The agents need to behave **inconsistently** to go in different directions toward the targets. In **Study 2**, we place all the targets on the right of the origin within the observation range of the agents. In this case, agents need to maintain **consistent** behaviors to go right toward the target. For **Study 1**, we report the proportion of agents that left the origin (with a denominator of 4 since one agent needs to stay at the origin which also has a target). In **Study 2**, we report the proportion of agents that approach the targets. Each variant is repeated 1000 times in different seeds.

In Table VII, compared with the existing SoTA method ELIGN, our delicately designed DCIR works well to encourage agents to perform dynamic behavior consistency at the right time and thus improve task performance. Besides, we visualize the two studies in Figs. 5 and 6, respectively. In **Study 1**, all the agents in our method gather at the origin at time step $t = 1$. Then, they dispersed to search for distant targets, except for *Agent2*●. It indicates that DCIR encourages inconsistent behaviors, causing the other 4 agents to leave the origin to occupy more targets, whereas ELIGN tends to promote consistent behaviors, resulting in all agents leaving the origin. In **Study 2**, all the agents in our method behave consistently to the right in order to occupy the 5 targets, while ELIGN does otherwise. It can be seen that our method also outperforms ELIGN in encouraging consistency.

I. Performance on Continuous Action Settings

To evaluate our DCIR in the continuous actions settings, we change the 2-D space in the *Coop Nav. (5v0)* task to a 1-D space. The agent intelligently moves forward or backwards in a straight line. The forward and backward velocities are continuous. To output the continuous velocity, we first change the output of the network to the mean and standard deviation of a Gaussian distribution. Then, we use reparameterization techniques to sample the velocity value from the Gaussian distribution.

In this way, the behavior consistency is measured by the divergence of two Gaussian distributions. We represent the output Gaussian distribution of agent i as P_i and of agent j as P_j. Supposed that the Gaussian distributions with means (μ_1, μ_2) and standard deviations (σ_1, σ_2), respectively, we use the following formula to calculate the Kullback–Leibler (KL) divergence:

$$KL\left(P_i \| P_j\right) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}. \tag{16}$$

We compare our DCIR with ELIGN under continuous settings, and all experimental parameter settings remained the same as in the article. As shown in Table VIII, our method

TABLE VI

MEAN TEST EPISODE EXTRINSIC REWARDS AND STANDARD ERRORS IN MULTIAGENT PARTICLE ENVIRONMENT UNDER SYMMETRY-BREAKING SETTING. HIGHER VALUES ARE BETTER

Methods	MPE (Cooperative)		MPE (Competitive)		
	Coop Nav. (5v0)	Hetero Nav. (6v0)	Phy Decep. (4v2)	Pred-prey (4v4)	Keep-away (4v4)
SPARSE	328.24 ± 24.17	405.08 ± 21.53	172.87 ± 32.43	-35.40 ± 8.63	1.37 ± 3.48
EXP-self	295.48 ± 20.54	436.17 ± 26.30	202.39 ± 26.06	-11.19 ± 3.65	9.24 ± 8.49
EXP-team	316.33 ± 14.44	422.71 ± 13.24	229.50 ± 28.29	-11.56 ± 6.37	-1.29 ± 1.58
ELIGN	357.40 ± 19.52	417.94 ± 22.29	184.21 ± 23.16	-7.34 ± 5.12	18.71 ± 14.78
LIIR	337.14 ± 10.01	439.07 ± 25.17	184.25 ± 58.98	-52.98 ± 13.55	2.64 ± 17.86
DCIR(Ours)	$\mathbf{360.46 \pm 9.65}$	$\mathbf{461.90 \pm 40.26}$	214.61 ± 31.05	$\mathbf{-7.05 \pm 3.70}$	$\mathbf{23.56 \pm 13.81}$

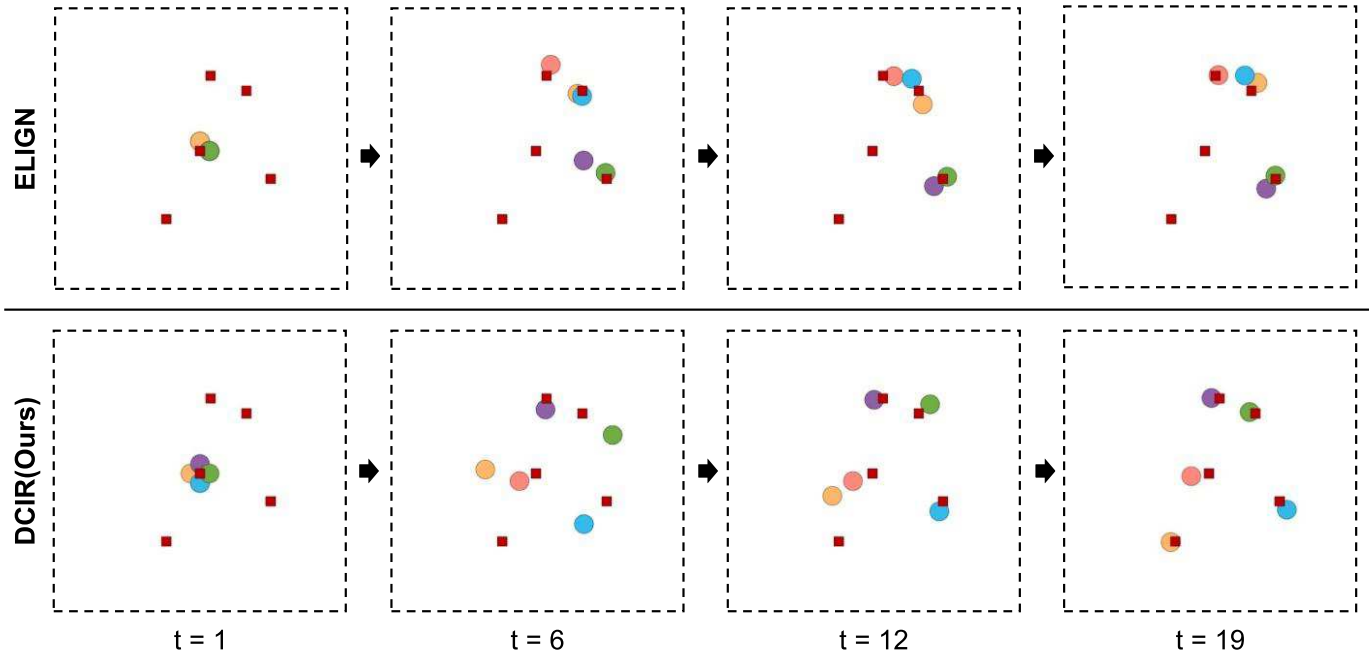

Fig. 5. Visualization of ***Study 1*** in one episode. Goal: ■. *Agent1*:●. *Agent2*:●. *Agent3*:●. *Agent4*:●. *Agent5*:●.

TABLE VII

PILOT STUDIES ON DYNAMIC BEHAVIOR CONSISTENCY

Method	Study 1	Study 2
ELIGN	0.75 ± 0.07	0.69 ± 0.18
DCIR(Ours)	$\mathbf{0.88 \pm 0.08}$	$\mathbf{0.97 \pm 0.02}$

TABLE VIII

MEAN TEST EPISODE EXTRINSIC REWARDS AND STANDARD ERRORS IN MULTIAGENT PARTICLE ENVIRONMENT UNDER CONTINUOUS ACTION SETTING. HIGHER VALUES ARE BETTER

Method	Task	Reward
ELIGN	Coop Nav. (5v0)	199.42 ± 18.13
DCIR(Ours)	Coop Nav. (5v0)	$\mathbf{214.69 \pm 17.09}$

outperforms ELIGN. We believe our performance could be further improved if we carefully design how to design a more suitable metric for behavior consistency. Also, exploring more complex MARL continuous environments which have been not studied much in the community is an interesting research direction. We leave this for future work.

J. Qualitative Results

In this section, we provide a more detailed analysis of the behavior consistency metric $\mathcal{C}$ and dynamic scale factor α of agents with different roles at different task phases for different tasks. For clarity and intuitive understanding, we present and analyze the results qualitatively.

In Fig. 7, we present the extrinsic reward and intrinsic reward, as well as the behavior consistency metric $\mathcal{C}$ and dynamic scale factor α over time in an episode of the *Coop Nav. (5v0)* task.

We take Agent3 as an example to clarify how DCIR dynamically encourages consistency:

- At the beginning, Agent3 has consistent behavior with Agent2 as the $\mathcal{C}$ between Agent3 and 2 is low.
- When both Agent3 and 2 approach a target at t = 3, the α_2^3 of Agent3 to Agent2 changes from negative to positive, thus encouraging it to behave differently from Agent2. This results in Agent3 humbly giving way to Agent2 and

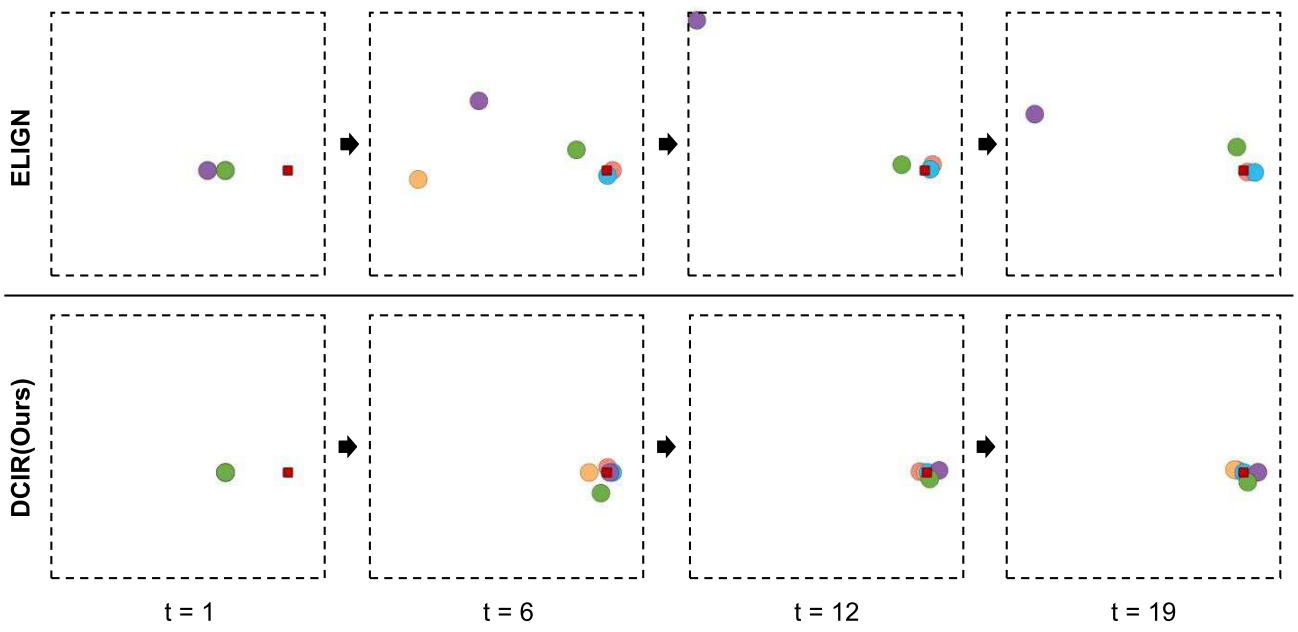

Fig. 6. Visualization of ***Study 2*** in one episode. Goal: ■. *Agent1*:●. *Agent2*:●. *Agent3*:●. *Agent4*:●. *Agent5*:●.

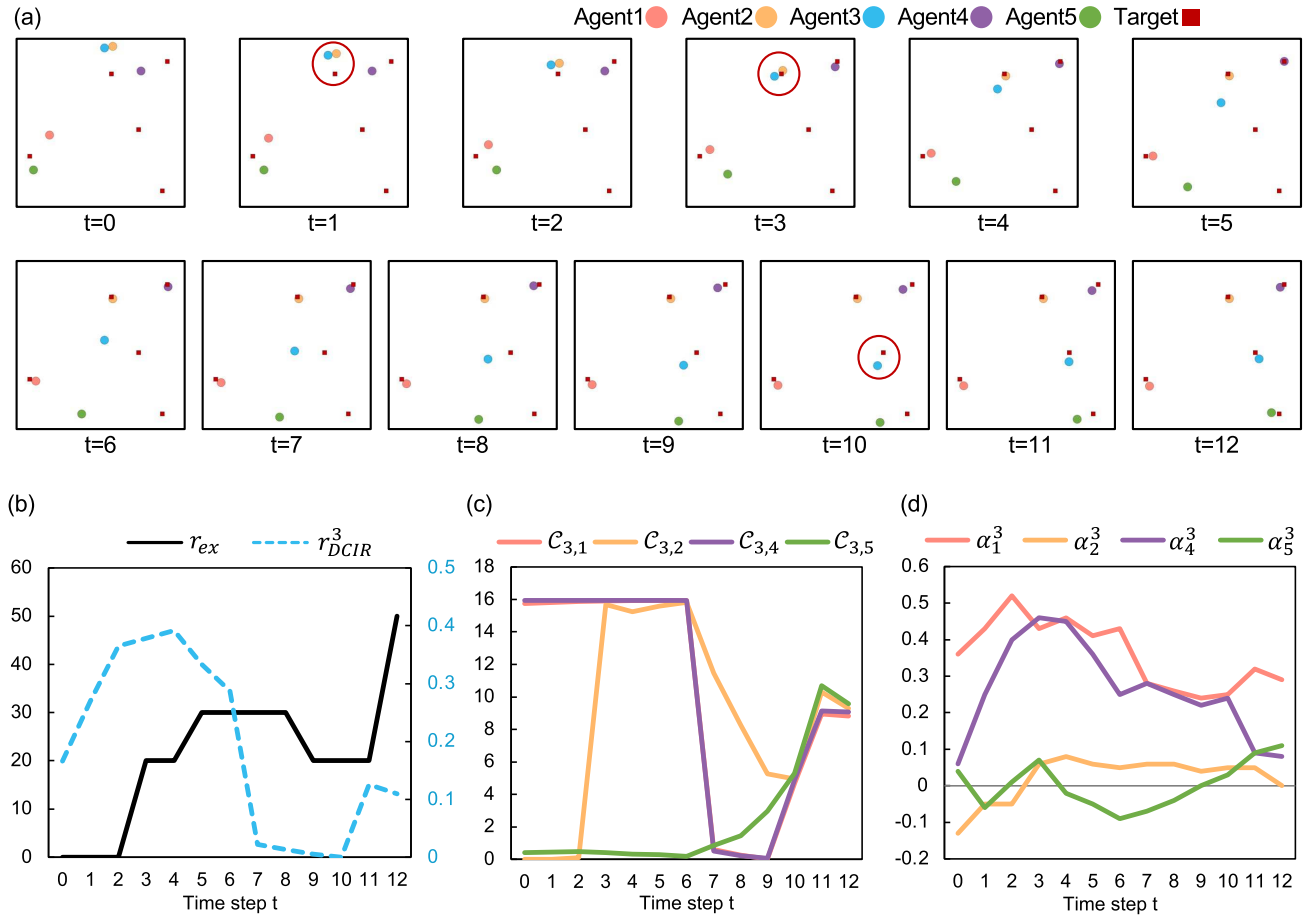

Fig. 7. (a) Visualization of cooperation navigation (5v0) task. (b) Extrinsic and intrinsic reward. (c) Behavior consistency $\mathcal{C}$. (d) Dynamic scale factor α.

continuing to explore. The humble behavior of Agent3 can potentially increase the likelihood of occupying more targets, thus the intrinsic reward is increased.

- Between t = 3 and t = 6, Agent3 is in the exploration stage, which is consistent with Agent5. The higher intrinsic reward indicates continued encouragement of exploration.

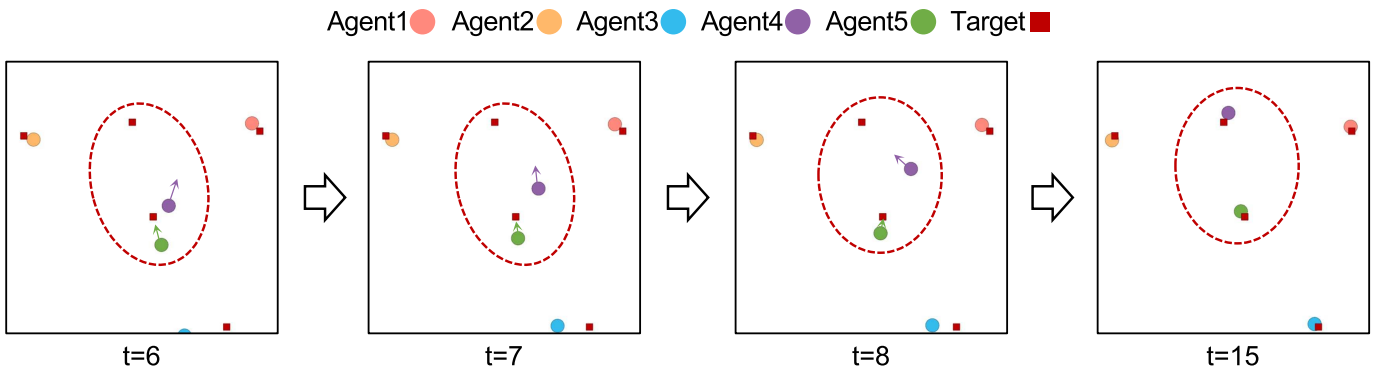

Fig. 8. More visualization of cooperation navigation (5v0) task. Behavior direction of Agent1: →. Behavior direction of Agent4: →.

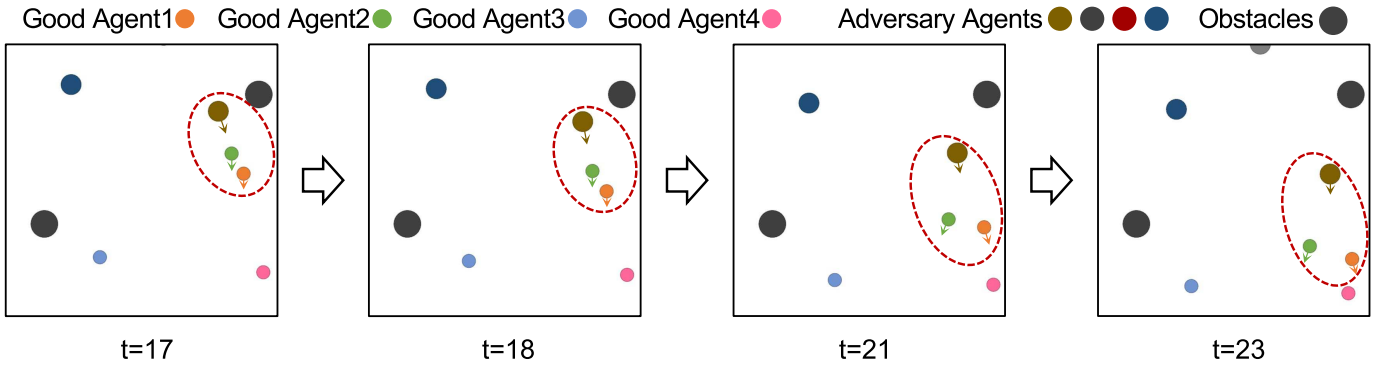

Fig. 9. Visualization of *Predator-prey task (4v4)* task (Case 1). Behavior direction of *Good Agent1*: →. Behavior direction of *Good Agent2*: →. Behavior direction of *Adversary Agent*: →.

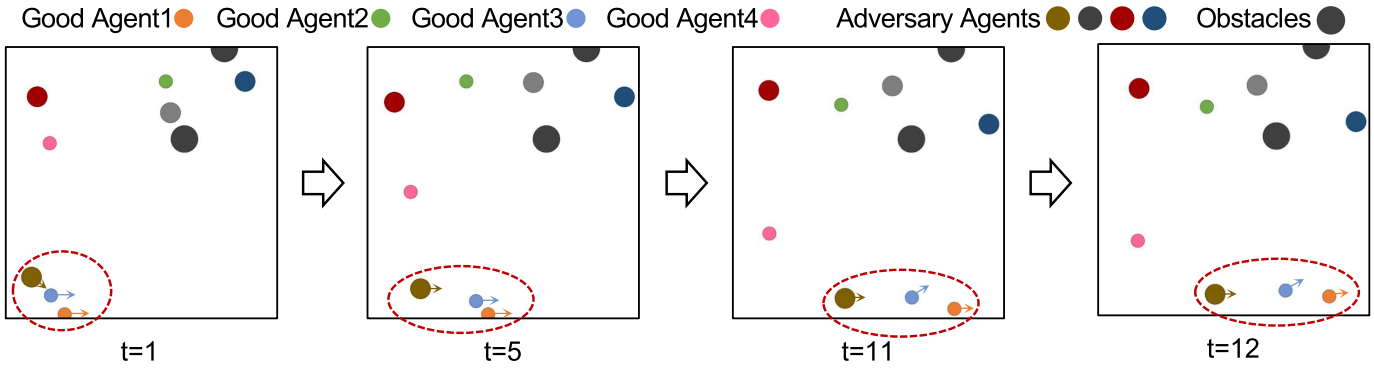

Fig. 10. Visualization of *Predator-prey task (4v4)* task (Case 2). Behavior direction of *Good Agent1*: →. Behavior direction of *Good Agent3*: →. Behavior direction of *Adversary Agent*: →.

- From t = 7 to t = 8, Agent3 finds out the target, but due to its tendency to continue exploring and move away from the target, its intrinsic decreases.
- However, as it is incentivized to approach the behavior of Agents1 and 4 (the α_1^3 and α_4^3 of Agent 3 to Agent1 and 4 gradually decreases), it gradually transitions from exploration to approaching the target and eventually reaches the target, with intrinsic reward increasing.

Moreover, we repeat the visualization experiment across other states from different tasks. The results are shown in Figs. 8–10, respectively. We analyze them below.

In Fig. 8, at t = 6, Agent5 and Agent4 have conflict target. DCIR incentivizes them to behave inconsistently ($\alpha_4^5 = 0.28 > 0$). While Agent5 behave aggressively, Agent4 should walk away. Agent5 should be encouraged to be consistent with Agent3 who also walks to its target ($\alpha_3^5 = -0.38 < 0$). From t = 6 to t = 8, encouraged by DCIR, behavior consistency between Agent5 and Agent3 gradually increases, with $\mathcal{C}_{5,3}$ decreases: 12.09 (t = 6)→ 10.22 (t = 7) → 4.04 (t = 8). All agents finally arrive at all targets at t = 15.

In Fig. 9, from t = 17 to 18, the Adversary is close to Good Agent1 and 2. DCIR incentivizes them to behave consistently ($\alpha_1^2 = -0.07$ at t = 17 and $\alpha_1^2 = -0.05$ at t = 18, both negative), so as to promote escaping pursuit faster. From t = 21 to 23, the Adversary is relatively far from Good Agent1 and 2. DCIR incentivizes them to behave inconsistently ($\alpha_1^2 = 0.03$ at

t = 21 and $\alpha_1^2 = 0.14$ at t = 23, both positive), to reduce the probability of being caught up at the same time.

In Fig. 10, from t = 1 to 5, the Adversary is close to Good Agent3 to 1. DCIR incentivizes them to behave consistently ($\alpha_1^3 = -0.02$ at t = 1 and $\alpha_1^3 = -0.04$ at t = 5, both negative), so as to promote escaping pursuit faster. From t = 11 to 12, the Adversary is relatively far from Good Agent3 to 1. DCIR incentivizes them to behave inconsistently ($\alpha_1^3 = 0.02$ at t = 11 and $\alpha_1^3 = 0.04$ at t = 12, both positive), so as to reduce the probability of being caught up at the same time.

K. Convergence Properties

Our method builds on the foundation of policy gradient algorithms, which are known to converge under standard assumptions such as smoothness and boundedness of the reward function. The intrinsic reward mechanism in DCIR is designed to be bounded and differentiable, ensuring that the overall objective function remains well-behaved. Furthermore, the dynamic scaling factors introduced by the DSN are optimized jointly with the policy, ensuring that the training process remains stable and converges to a locally optimal policy. Empirical results in Figs. 3 and 4 demonstrate the reward convergence of our method.

L. Computational Complexity

The computational overhead of our method comes from the behavior consistency and the dynamic scaling factors. Calculating the behavior consistency of agents requires inputting their own observations into other agents' policies, which is the source of additional computation. However, during the testing stage, we do not need to calculate the intrinsic reward, so the additional computation is 0. In the training stage, since there is no gradient in behavior consistency, the additional computation is relatively small compared to the entire training process. As for the by dynamic scaling factors, which are inferred from DSN, there is also no computational overhead in the testing stage as the intrinsic reward is not calculated. During the training stage, we conducted an experiment to quantify the additional computation time after integrating the DSN. Our results show that in one training iteration, the additional computation of DSN parameter updates accounts for only 0.22% of the total training time. This negligible overhead demonstrates the efficiency of our approach.

VI. Conclusion

In this article, we address the MARL cooperation problem from the perspective of behavior consistency between agents. We propose to assign each agent a dynamic consistency-based intrinsic reward (DCIR) to enhance performance in multiagent tasks. Experimental results show that the proposed method significantly outperforms baseline MARL methods on the Multiagent Particle, Google Research Football, and StarCraft II benchmark environments. Qualitative results also show that the proposed metric correctly defines the behavior consistency and the DSN successfully outputs the appropriate scale factors for incentivizing the agents to perform their optimal behaviors. Future work will focus on these aspects.

1) *Explicit Behavior Representation:* Develop hybrid models that combine symbolic reasoning with deep learning techniques to create interpretable representations of team behaviors. For example, incorporating attention mechanisms or graph-structured models to explicitly capture interagent relationships and decision-making patterns.
2) *Handling More Complex Action Spaces:* Extend our framework to handle more high-dimensional and hierarchical action spaces by designing new reward structures or decomposing tasks into subgoals (e.g., using large language models), enabling agents to coordinate more effectively in complex environments.
3) *Cross-Domain Knowledge Transfer:* Investigate methods for transferring learned behaviors across different domains and tasks with different levels, such as using modular neural architectures or meta-learning frameworks to adapt strategies from one task to another without finetuning.

References

[1] Z. He, L. Dong, C. Song, and C. Sun, "Multiagent soft actor-critic based hybrid motion planner for mobile robots," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10980–10992, Dec. 2023.

[2] C. Song, Z. He, and L. Dong, "A local-and-global attention reinforcement learning algorithm for multiagent cooperative navigation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 6, pp. 7767–7777, Jun. 2024.

[3] L. C. Garaffa, M. Basso, A. A. Konzen, and E. P. de Freitas, "Reinforcement learning for mobile robotics exploration: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3796–3810, Aug. 2023.

[4] B. Peng et al., "FACMAC: Factored multi-agent centralised policy gradients," in *Proc. NeurIPS*, Jun. 2020, pp. 12208–12221.

[5] J. Long, D. Yu, G. Wen, L. Li, Z. Wang, and C. L. P. Chen, "Game-based backstepping design for strict-feedback nonlinear multi-agent systems based on reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 817–830, Jan. 2024.

[6] R. Zhang, Q. Zong, X. Zhang, L. Dou, and B. Tian, "Game of drones: Multi-UAV pursuit-evasion game with online motion planning by deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7900–7909, Oct. 2023.

[7] M. Carroll et al., "On the utility of learning about humans for human-AI coordination," in *Proc. NeurIPS*, 2019, pp. 5175–5186.

[8] Y. Wu, S. Liao, X. Liu, Z. Li, and R. Lu, "Deep reinforcement learning on autonomous driving policy with auxiliary critic network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 7, pp. 3680–3690, Jul. 2023.

[9] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7391–7403, Oct. 2023.

[10] J. Dinneweth, A. Boubezoul, R. Mandiau, and S. Espié, "Multi-agent reinforcement learning for autonomous vehicles: A survey," *Auto. Intell. Syst.*, vol. 2, no. 1, p. 27, Nov. 2022.

[11] Z. Ma, R. Wang, F. Li, M. Bernstein, and R. Krishna, "ELIGN: Expectation alignment as a multi-agent intrinsic reward," in *Proc. NeurIPS*, 2022, pp. 8304–8317.

[12] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, San Francisco, CA, USA, 1999, pp. 278–287.

[13] Y. Du, L. Han, M. Fang, J. Liu, T. Dai, and D. Tao, "LIIR: Learning individual intrinsic reward in multi-agent reinforcement learning," in *Proc. NeurIPS*, vol. 32, 2019, pp. 4403–4414.

[14] M. Jaderberg et al., "Human-level performance in 3D multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, May 2019.

[15] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," 2015, *arXiv:1507.00814*.

[16] S. Iqbal and F. Sha, "Coordinated exploration via intrinsic rewards for multi-agent reinforcement learning," 2019, *arXiv:1905.12127*.

[17] B. Liu, Z. Pu, Y. Pan, J. Yi, Y. Liang, and D. Zhang, "Lazy agents: A new perspective on solving sparse reward problem in multi-agent reinforcement learning," in *Proc. ICML*, Apr. 2023, pp. 21937–21950.

[18] J. Zhang, Y. Zhang, X. S. Zhang, Y. Zang, and J. Cheng, "Intrinsic action tendency consistency for cooperative multi-agent reinforcement learning," in *Proc. AAAI*, vol. 38, 2024, pp. 17600–17608.

[19] C. Li, C. Wu, T. Wang, J. Yang, Q. Zhao, and C. Zhang, "Celebrating diversity in shared multi-agent reinforcement learning," in *Proc. NeurIPS*, 2021, pp. 3991–4002.

[20] A. Zhaikhan and A. H. Sayed, "Graph exploration for effective multiagent Q-learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 3, pp. 5535–5546, Mar. 2025.

[21] R. Charakorn, P. Manoonpong, and N. Dilokthanakul, "Generating diverse cooperative agents by learning incompatible policies," in *Proc. ICLR*, 2023, pp. 1–13.

[22] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. NIPS*, 2017, pp. 6379–6390.

[23] K. Kurach et al., "Google research football: A novel reinforcement learning environment," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, 2020, pp. 4501–4510.

[24] M. Samvelyan et al., "The StarCraft multi-agent challenge," in *Proc. Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2019, pp. 2186–2188.

[25] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[26] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications*. Heidelberg, Germany: Springer, Mar. 2010, pp. 183–221.

[27] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," in *Handbook of Reinforcement Learning and Control*, Cham, Switzerland: Springer, Jun. 2021, pp. 321–384.

[28] S. D. J. McArthur et al., "Multi-agent systems for power engineering applications—Part II: Technologies, standards, and tools for building multi-agent systems," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1753–1759, Nov. 2007.

[29] C. Berner et al., "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[30] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative, multi-agent games," in *Proc. NeurIPS*, 2021, pp. 24611–24624.

[31] F. Lin, S. Huang, T. Pearce, W. Chen, and W.-W. Tu, "TiZero: Mastering multi-agent football with curriculum learning and self-play," in *Proc. AAMAS*, 2023, pp. 67–76.

[32] S. Ding, W. Du, L. Ding, J. Zhang, L. Guo, and B. An, "Robust multi-agent communication with graph information bottleneck optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 5, pp. 3096–3107, May 2024.

[33] F. Alzetta, P. Giorgini, A. Najjar, M. Schumacher, and D. Calvaresi, "In-time explainability in multi-agent systems: Challenges, opportunities, and roadmap," in *Proc. EXTRAAMAS*, 2020, pp. 39–53.

[34] L. Canese et al., "Multi-agent reinforcement learning: A review of challenges and applications," *Appl. Sci.*, vol. 11, no. 11, p. 4948, May 2021.

[35] C. Zhang, S. Jin, W. Xue, X. Xie, S. Chen, and R. Chen, "Independent reinforcement learning for weakly cooperative multiagent traffic control problem," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7426–7436, Aug. 2021.

[36] X. Xu, R. Li, Z. Zhao, and H. Zhang, "Stigmergic independent reinforcement learning for multiagent collaboration," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 4285–4299, Sep. 2022.

[37] J. Foerster and Y. M. Assael, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.

[38] G. Chen, "A new framework for multi-agent reinforcement learning-centralized training and exploration with decentralized execution via policy distillation," in *Proc. AAMAS*, Apr. 2020, pp. 1801–1803.

[39] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 2961–2970.

[40] I.-J. Liu, R. A. Yeh, and A. G. Schwing, "PIC: Permutation invariant critic for multi-agent deep reinforcement learning," in *Proc. CoRL*, 2019, pp. 590–602.

[41] W. Li, X. Wang, B. Jin, D. Luo, and H. Zha, "Structured cooperative reinforcement learning with time-varying composite action space," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8618–8634, Nov. 2022.

[42] T. Hu, B. Luo, C. Yang, and T. Huang, "MO-MIX: Multi-objective multi-agent cooperative decision-making with deep reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12098–12112, Oct. 2023.

[43] W. Qiu et al., "RMIX: Learning risk-sensitive policies for cooperative reinforcement learning agents," in *Proc. NeurIPS*, 2021, pp. 23049–23062.

[44] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. JMLR*, 2020, pp. 7234–7284.

[45] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.

[46] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst.*, 2018, pp. 2085–2087.

[47] H. Wjkde et al., "Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. ICML*, 2019, pp. 5887–5896.

[48] H. Ding et al., "Learning to coordinate with different teammates via team probing," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 6, 2025, doi: 10.1109/TNNLS.2025.3563773.

[49] Y. Wang, J. Wang, R. Zhu, H. Fu, J. Chen, and C. L. P. Chen, "Facilitating multiagent coordination relying on graph information representation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 6, 2025, doi: 10.1109/TNNLS.2025.3575196.

[50] C. Li, T. Wang, C. Wu, Q. Zhao, J. Yang, and C. Zhang, "Celebrating diversity with subtask specialization in shared multiagent reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 2, pp. 2051–2065, Feb. 2025.

[51] S. Dong et al., "WToE: Learning when to explore in multiagent reinforcement learning," *IEEE Trans. Cybern.*, vol. 54, no. 8, pp. 4789–4801, Aug. 2024.

[52] P. Chen et al., "Learning active camera for multi-object navigation," in *Proc. NeurIPS*, 2022, pp. 28670–28682.

[53] Y. Hou et al., "Cooperative multiagent learning and exploration with min–max intrinsic motivation," *IEEE Trans. Cybern.*, vol. 55, no. 6, pp. 2852–2864, Jun. 2025.

[54] H. Wu, H. Li, J. Zhang, Z. Wang, and J. Zhang, "Generating individual intrinsic reward for cooperative multiagent reinforcement learning," *Int. J. Adv. Robotic Syst.*, vol. 18, no. 5, Sep. 2021, Art. no. 17298814211044946.

[55] W. Li, W. Liu, S. Shao, and S. Huang, "AIIR-MIX: Multi-agent reinforcement learning meets attention individual intrinsic reward mixing network," in *Proc. ACML*, 2023, pp. 579–594.

[56] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Cham, Switzerland: Springer, 2016.

[57] M. Andrychowicz et al., "Learning to learn by gradient descent by gradient descent," in *Proc. NeurIPS*, 2016, pp. 3988–3996.

[58] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2016, pp. 1842–1850.

[59] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1856–1865.

[60] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[61] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. NIPS*, 1999, pp. 1057–1063.

[62] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. ICML*, Apr. 2015, pp. 1889–1897.

[63] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[64] Z. Zheng, J. Oh, and S. Singh, "On learning intrinsic rewards for policy gradient methods," in *Proc. NeurIPS*, 2018, pp. 4649–4659.

[65] J. Weng et al., "Tianshou: A highly modularized deep reinforcement learning library," *JMLR*, vol. 23, no. 1, pp. 12275–12280, 2021.

[66] E. Liang et al., "RLlib: Abstractions for distributed reinforcement learning," in *Proc. ICML*, May 2017, pp. 3059–3068.

[67] M. Menéndez, J. Pardo, L. Pardo, and M. Pardo, "The Jensen–Shannon divergence," *J. Franklin Inst.*, vol. 334, no. 2, pp. 307–318, 1997.

[68] R. Beran, "Minimum Hellinger distance estimates for parametric models," *Ann. Statist.*, vol. 5, no. 3, pp. 445–463, May 1977.

[69] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vols. SMC–13, no. 5, pp. 834–846, Sep. 1983.

[70] R. E. Wang, S. Wu, J. A. Evans, J. B. Tenenbaum, D. C. Parkes, and M. Kleiman-Weiner, "Too many cooks: Bayesian inference for coordinating multi-agent collaboration," in *Proc. Hum.-Like Mach. Intell.*, 2020, pp. 152–170.

Kunyang Lin received the B.E. degree in automation science and engineering from South China University of Technology, Guangzhou, China, in 2021, and the M.E. degree from the School of Software Engineering, South China University of Technology, in 2024.

His research interests include embodied AI and reinforcement learning.

Yufeng Wang received the B.E. degree in electronic and information engineering from Wuhan University of Technology, Wuhan, China, in 2020, and the master's degree in information and communication engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2023. He is currently pursuing the Ph.D. degree in software engineering with South China University of Technology, Guangzhou, China.

His research interests include large language models and reinforcement learning.

Peihao Chen received the B.E. degree in automation science and engineering and the Ph.D. degree in software engineering from South China University of Technology, Guangzhou, China, in 2018 and 2024, respectively.

His research interests include embodied AI and multimodal understanding.

Runhao Zeng received the Ph.D. degree in software engineering from South China University of Technology, Guangzhou, China, in 2021.

He is currently an Associate Professor with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen, China. He has authored or co-authored several peer-reviewed articles on computer vision and machine learning at top-tier conferences and journals, including the Proceedings of NeurIPS, CVPR, ICCV, and TPAMI. His research interests include machine learning and computer vision, with a particular focus on video analysis.

Yinjie Lei (Senior Member, IEEE) received the M.S. degree in image processing from Sichuan University (SCU), Chengdu, China, in 2009, and the Ph.D. degree in computer vision from The University of Western Australia (UWA), Perth, WA, Australia, in 2013.

Since 2017, he has been the Vice Dean of the College of Electronics and Information Engineering, SCU, where he is currently an Associate Professor. His main research interests include 3-D biometrics, object recognition, and semantic segmentation.

Siyuan Zhou received the B.E. degree in electrical engineering and computer science from Peking University, Beijing, China, in 2022. He is currently pursuing the Ph.D. degree with The Hong Kong University of Science and Technology (HKUST), Hong Kong.

His research focuses on embodied AI and diffusion models.

Qing Du received the bachelor's, master's, and Ph.D. degrees in computer science and engineering from South China University of Technology, Guangzhou, China, in 2002, 2005, and 2014, respectively.

She is currently an Assistant Professor with the School of Software Engineering, South China University of Technology. Her research interests include machine learning and natural language processing.

Mingkui Tan received the bachelor's degree in environmental science and engineering and the master's degree in control science and engineering from Hunan University, Changsha, China, in 2006 and 2009, respectively, and the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2014.

From 2014 to 2016, he was a Senior Research Associate on computer vision at the School of Computer Science, University of Adelaide, Adelaide, SA, Australia. He is currently a Professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. His research interests include machine learning, sparse analysis, deep learning, and large-scale optimization.

Chuang Gan is currently an Assistant Professor with the UMass Amherst, Amherst, MA, USA. His research interests mainly include multimodality learning for video understanding.