Noise Distribution Decomposition Based Multi-Agent Distributional Reinforcement Learning

Wei Geng, Baidi Xiao, *Graduate Student Member, IEEE*, Rongpeng Li, *Senior Member, IEEE*, Ning Wei, Dong Wang, and Zhifeng Zhao, *Member, IEEE*

***Abstract*—Generally, Reinforcement Learning (RL) agent updates its policy by repetitively interacting with the environment, contingent on the received rewards to observed states and undertaken actions. However, the environmental disturbance, commonly leading to noisy observations (e.g., rewards and states), could significantly shape the performance of agent. Furthermore, the learning performance of Multi-Agent Reinforcement Learning (MARL) is more susceptible to noise due to the interference among intelligent agents. Therefore, it becomes imperative to revolutionize the design of MARL, so as to capably ameliorate the annoying impact of noisy rewards. In this paper, we propose a novel decomposition-based multi-agent distributional RL method by approximating the globally shared noisy reward by a Gaussian Mixture Model (GMM) and decomposing it into the combination of individual distributional local rewards, with which each agent can be updated locally through distributional RL. Moreover, a Diffusion Model (DM) is leveraged for reward generation in order to mitigate the issue of costly interaction expenditure for learning distributions. Furthermore, the monotonicity of the reward distribution decomposition is theoretically validated under nonnegative weights and increasing distortion risk function, while the design of the loss function is carefully calibrated to avoid decomposition ambiguity. We also verify the effectiveness of the proposed method through extensive simulation experiments with noisy rewards. Besides, different risk-sensitive policies are evaluated in order to demonstrate the superiority of distributional RL in different MARL tasks.**

***Index Terms*—Diffusion model, distribution decomposition, distributional reinforcement learning, multi-agent reinforcement learning, noisy environment.**

Received 16 June 2024; revised 12 October 2024; accepted 1 November 2024. Date of publication 7 November 2024; date of current version 5 February 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2024YFE0200600, in part by the National Natural Science Foundation of China under Grant 62071425, in part by the Zhejiang Key Research and Development Plan under Grant 2022C01093, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR23F010005, and in part by Zhejiang Lab under Grant 3400-31299. Recommended for acceptance by J. Yu. *(Corresponding authors: Ning Wei; Zhifeng Zhao.)*

Wei Geng, Ning Wei, and Zhifeng Zhao are with the Zhejiang Lab, Zhejiang 311121, China (e-mail: gengwei21@mails.ucas.ac.cn; weining@zhejianglab.com; zhaozf@zhejianglab.com).

Baidi Xiao and Rongpeng Li are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: xiaobaidi@zju.edu.cn; lirongpeng@zju.edu.cn).

Dong Wang is with the Research Institute of China Telecom, Beijing 100033, China (e-mail: wangd5@chinatelecom.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2024.3492272, provided by the authors.

Digital Object Identifier 10.1109/TMC.2024.3492272

I. Introduction

REINFORCEMENT learning (RL) [1] has significant applications in various domains, including game AI training [2], robot control [3], and large language models' optimization [4]. With the rise of Deep Learning (DL) [5] technology and its remarkable achievements in various fields, the integration of Deep Neural Networks (DNNs) and RL [6] has become a hot topic in research. The accompanied tremendous success of deep RL has prompted researchers to shift their focus to the field of Multi-Agent Systems (MAS) [7], and given rise to Multi-Agent Reinforcement Learning (MARL). For most cooperative MARL tasks, it is crucial to develop appropriate means to leverage a global reward to train decentralized action value (i.e., Q-value) or state value (i.e., V -value) networks, and update the policies accordingly, especially for settings with partial observations and constrained communications [8], [9]. In particular, some value decomposition methods [9], [10], [11], [12] are proposed.

Nevertheless, environmental noise widely exists in practice due to internal and external factors (e.g., electronic noise from sensors, the influences of temperature, pressure, and illumination) [13], and most RL algorithms are vulnerable in noisy scenarios with unstable performance [14]. Some off-the-shelf MARL algorithms also face troublesome convergence issues and experience severely deteriorated performance under noisy scenarios, especially for those adversarial tasks where agents might suffer malicious interference from opponents [15]. The distributional RL [16], which models the action value or state value as a distribution, emerges as a promising solution for anti-noise. This distribution can effectively model the randomness of a system or environment, and distributional RL can even surpass human performance in many scenarios where the optimization must go beyond considering the expected return values and evaluate the distribution of certain performance metrics/rewards [17]. For instance, in the field of communication, distributional RL demonstrates its advantages in the robust network optimization that accounts for the outage probabilities [18], the management of slice Service Level Agreements (SLAs) defined by availability and performance thresholds [19], and resource allocation strategies that consider risk in decision-making processes [20]. Meanwhile, in game AI, [16], [21], [22] demonstrate the superior performance of distributional RL in dozens of tasks about Atari. However, incorporating value decomposition into classic distributional RL for noisy MARL tasks is rather counter-effective, considering the multi-folded

detrimental effects. First, it is typically hard to train a quantile network [16], [22], which aims to learn a specific Cumulative Distribution Function (CDF) of action or state value distribution. Therefore, it usually takes distributional RL algorithms more time to converge. Second, making specific assumptions about the exact type of global distribution lacks robustness. As not all types of distribution are additive or support linear operations, changing the types of individual distributions through temporal-difference (TD) update also increases model complexity [23]. Additionally, decomposing the global distribution into individual distributions is erratic and unstable [24], as slight changes in the global distribution usually cause individual distributions to vibrate drastically. Finally, the interaction cost between agents and environment, as well as the learning cost, will significantly increase [25].

In this paper, we propose the Noise Distribution Decomposition (NDD) MARL method, wherein the ideas of distributional RL and value decomposition are jointly leveraged to ameliorate the flexibility of distributed agents to noisy rewards. The utilization of distributional RL enables the agent to remap quantiles using the distortion risk function, thereby better taking account of the distribution over returns and potentially generating distinctive policies with different risk-sensitive functions [22]. Meanwhile, in order to tackle the difficulties of incorporating value decomposition into classic distributional RL for MARL tasks, we successfully leverage a parametric Gaussian Mixture Model (GMM) [26] and establish a viable decomposition means with calibrated Wasserstein-metric-based [21] loss function. Furthermore, inspired by the astonishing representation capabilities of Diffusion Models (DM) [27], [28], [29], [30], [31], we also incorporate DM to model the noise distribution to reduce the interaction cost. Specifically, we use DM to generate additional data based on existing interaction data from agents, and then proportionally combine the generated data with the raw data to train policies in NDD. The main contributions of our work can be summarized as follows:

- We introduce NDD on top of distributional RL and value decomposition in MARL. For the ensemble of agents, NDD approximates the distribution of the noisy global reward by GMM and obtains the local distributional value functions indirectly for yielding more stable individual policies. For a single agent, NDD also contributes to more conveniently learning the action value distribution even with partial observation.
- We extend distributional RL to the multi-agent domain and introduce distortion risk functions, which facilitate the determination to adopt adventurous or conservative strategies by adjusting action value distributions, to provide more flexibility than classical RL with expectation.
- We carefully calibrate a Wasserstein-metric-based loss function to learn the accurate distribution corresponding to each agent and prove the consistency between the globally optimal action and locally optimal actions theoretically.
- To alleviate the high cost of environment interaction required for learning distributions, we introduce DM to augment the data for ameliorating the data scarcity issue and prove the bounded expectations of the generation and approximation errors in the augmented data, which can be a reference for the trade off between interaction cost and errors.

In the rest of this paper, we present the related work of RL and MARL in Section II. Afterward, based on the necessary notations and preliminaries in Section III, we describe our proposed method in Section IV. Subsequently, numerous simulation results are shown to validate the effectiveness and superiority of our method in Section V. Finally, Section VI concludes the paper and lists the future work.

II. Related Work

According to the cooperativeness between their reward functions, MARL can be categorized into three types (i.e., fully cooperative, fully competitive, and mixed tasks) [32]. For fully cooperative tasks, the reward functions of agents are identical, signifying that all agents are striving to achieve a common goal [33], [34]. Algorithms targeted at this type of task can integrate the computational and learning capabilities of MAS, and potentially achieve gains beyond simple summation [32]. For fully competitive tasks, the reward functions are opposite, typically involving two completely antagonistic agents in the environment, while the agents aim to maximize their rewards and minimize the opponent's rewards, with Minimax-Q [35] being a representative algorithm. In mixed tasks, the relationship of agents could be stochastic, leading to complicated reward functions of agents and making this model suitable for self-interested agents [32]. Generally, the resolution of such tasks is often associated with the concept of equilibrium in game theory. In this paper, we primarily focus on the first, fully cooperative setting, since the related algorithms we will mention later are more conducive to implementation, migration, and expansion to more large-scale MAS [32], [36] compared with the other two types. On the other hand, the cooperative algorithm has a close connection with distributed optimization [36]. Therefore, efficient optimization techniques such as decentralized training, asynchronous training [37], and communication learning [38], [39] can be introduced to tackle issues in cooperative learning.

However, there are still many unresolved issues for cooperative algorithms. First, in a multi-agent environment, each agent not only needs to consider its own actions and rewards but also must take account of the influences of the actions undertaken by other agents [32]. Such an intricate process of interaction and interconnection results in constantly undergoing changes in a non-stationary environment, making the actions and strategy choices among agents interdependent. Therefore, the updated strategies of other agents impede each agent from approximating accurate action or state value functions for making a self-optimized strategy. To solve this issue, based on Deep Q-Network (DQN) [40], Castaneda et al. [41] propose to modify the value functions and reward functions to fit the interdependence among agents. Diallo et al. [42] introduce a parallel computing mechanism into DQN to expedite the convergence in non-stationary environments. Foerster et al. [38] focus on improving the experience replay to better adapt to continuously changing non-stationary environments. Another important issue

in cooperative MARL lies in the limitation of partial observation. That is, agents can only make near-optimal decisions based on their own partially observable information. In the collaborative task with local observations and globally shared rewards, value decomposition methods are often adopted to effectively learn the different contributions of agents from global rewards. For example, VDN [9] decomposes the global value function into the sum of individual value functions. Based on VDN, QMIX [10] further generalizes the way of value decomposition through sophisticated transformation, and puts forward an important assumption on the monotonicity of the global action value function with respect to local ones. Focusing on the further exploration of the joint action, Weighted QMIX [11] and QTRAN [12] discuss the relationship between individual optimal actions and the global optimal joint action. Nevertheless, all these value decomposition algorithms [9], [10], [11], [12] suffer from the adoption of over-simplistic expectation operation, which is not sufficient to describe the noisy environment that agents may encounter. In a nutshell, it necessitates a more comprehensive characterization of the environment, and developing novel decomposition means to explore adventurous or conservative strategies. Coincidentally, it falls into the scope of distributional RL [16], [21], [22] and distortion risk functions [22].

Primarily adopted in single-agent settings, distributional RL [16], [21], [22] comprehensively characterizes the fluctuation by replacing the expectation of state or action values with the corresponding distributions and yields appealing effects. For example, C51 [16] achieves significant advantages over DQN, DDQN, Duel DQN, and human baseline across 57 Atari games. QR-DQN [21] applies regression theory [43] to distributional RL and achieves performance beyond C51. Specific distortion risk functions based on human decision-making habits are proven effective by IQN [22] in most Atari games. Furthermore, DFAC [24] extends the value decomposition method to the field of distributional RL in MAS, and develops the joint quantile function decomposition to extend VDN and QMIX to DDN and DMIX on condition of stable and noise-free rewards. However, a noticeable drawback of distributional RL is the increased computation and interaction cost to train quantile networks alongside the TD update. For combatting the computation cost, DRE-MARL [44] designs the multi-action-branch reward estimation and policy-weighted reward aggregation for stabilized training. Additionally, distortion risk functions, which primarily adjust the risk preference of policy in the action value distribution and are mainly employed in tasks where the correlation between risk and reward is evident, are proposed by [22]. Despite the progress in single-agent setting, the adoption of distortion risk functions is worthy of further investigation, as it may disrupt the collaboration among MAS in MARL, and convergence issues may arise if theoretical guarantees lack [22], [24]. Meanwhile, the interaction cost can be compensated by generative models [45], [46]. Among them, DM [27] is renowned for its astonishing capability in the fields of image synthesis [28], image inpainting, color restoration, image decompression [29] and text-to-image [30], [31]. RL has reaped the benefits of DMs with significant improvement in cross-task learning and experience augmentation. For example, [47] and [48] use DM for policy generation in single-agent tasks. J. Geng [49] leverages DM to encapsulate policies within the MAS context, thereby fostering efficient and expressive inter-agent coordination. DOM2 [50] incorporates DM into the policy network and proposes a trajectory-based data-augmentation scheme in training. However, mostly in single-agent settings, these studies lack validation in non-stationary and noisy environments. Moreover, the approximation errors of DM in generating samples also lack theoretical analyses.

TABLE I
MAIN NOTATIONS USED IN THIS PAPER

Notation	Definition
N	Number of agents
$\mathcal{I}$	Set of agents
$\mathcal{S}$, $\boldsymbol{\mathcal{A}}$, and $\boldsymbol{\mathcal{O}}$	The state, joint action and joint observation space
s	Global state
$\boldsymbol{a} = [a^{[1]}, \cdots, a^{[N]}]$	Joint action of all agents
$\boldsymbol{o} = [o^{[1]}, \cdots, o^{[N]}]$	Joint observation of all agents
Q and V	Action value and state value function
$\boldsymbol{w} = [w^{[1]}, \cdots, w^{[N]}]^\top$	Joint weight vector of all agents
μ_i, σ_i	Mean and standard deviation related to agent i
$\boldsymbol{\pi} = [\pi^{[1]}, \ldots, \pi^{[N]}]$	Joint policy of all agents
$\boldsymbol{\theta} = [\theta^{[1]}, \cdots, \theta^{[N]}]$	Neural network parameters of all agents
Z	Random variable fitting action value distribution
F_Z	CDF of random variable Z
τ	Quantile value of a target distribution
$\rho(\tau)$	Distortion risk function about quantile τ
γ	Discount factor
R	Random variable fitting reward distribution
r and $\mathcal{R}$	Reward sample produced by the function $\mathcal{R}$
$r^{<k>}$	Generated reward in k-th step of DM
K	Total diffusion steps of DM
$\upsilon_1, \cdots, \upsilon_K$ and $\omega_1, \cdots, \omega_K$	Predefined hyper-parameters in K steps of DM
Θ	Parameters of DM
$\mathcal{D}_q(\cdot)$	Distribution of random variable or its sample
d_p	Wasserstein distance computed by L_p-norm
$\mathcal{D}$	Global reward distribution
$\hat{\mathcal{D}}_l$	Decomposed distribution of local reward

III. BACKGROUND AND PROBLEM FORMULATION

This section mainly discusses the base model in our work and the formulations of relevant methods. The main notations used in this paper are listed in Table I.

A. System Model

A cooperative multi-agent task can be described as the decentralized partially observable Markov decision process (Dec-POMDP), consisting of a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega, \mathcal{I}, \gamma \rangle$ [10], [51]. Specifically, $s \in \mathcal{S}$ denotes the global state and $\boldsymbol{a} = [a^{[1]}, \ldots, a^{[N]}] \in \mathcal{A}$ is the joint action of N agents. The joint action causes a transition from current state s to next state s' according to the transition function $\mathcal{P}(s'|s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. Specially, all agents share a global reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which means its output $R(s, \boldsymbol{a})$ follows a

global reward distribution $\mathcal{D}$ when given s and $\boldsymbol{a}$, and $r \in \mathbb{R}$ is sampled from $R(s, \boldsymbol{a})$. Each agent $i \in \mathcal{I} \equiv \{1, \ldots, N\}$ draws individual observation $o^{[i]} \in \mathcal{O}^{[i]}$ where $\mathcal{O}^{[i]} \in \mathcal{O}$ according to the observation function $\Omega(s, i) : \mathcal{S} \times \mathcal{I} \to \mathcal{O}^{[i]}$. Generally, the joint observation $\boldsymbol{o} = [o^{[1]}, \ldots, o^{[N]}]$ and we assume that s and $\boldsymbol{o}$ are equivalent. Besides, $\gamma \in [0, 1)$ is a discount factor. Within the framework of Dec-POMDP in t-th step, a joint policy $\boldsymbol{\pi} = [\pi^{[1]}, \ldots, \pi^{[N]}]$ corresponds to a joint action value function as

$$Q^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t) = \mathbb{E}_{s_{t:\infty}, \boldsymbol{a}_{t:\infty}}[Z^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t)], \tag{1}$$

where the random variable for the discounted reward summation $Z^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t) = \sum_{m=0}^{\infty} \gamma^m R(s_{t+m}, \boldsymbol{a}_{t+m})$ has a corresponding CDF F_Z. Notably, when the quantile τ of the CDF is further distorted by distortion risk function $\rho(\tau) : [0, 1] \to [0, 1]$, the distorted action value function $Q_\rho(o, a)$ and distorted random variable $Z_\rho(o, a)$ can be expressed in terms of $\rho(\tau)$ as

$$Q_\rho^\pi(o, a) = \mathbb{E}\left[Z_\rho^\pi(o, a)\right] = \int z \mathrm{d}\rho\left[F_Z(z)\right]. \tag{2}$$

Additionally, DM gradually adds Gaussian noises to sample r from $\mathcal{D}$ by K times and then reconstructs it in reverse. In the diffusion process, $r^{<k>} \sim \mathcal{D}_q(r^{<k>})$ is a generated sample in k-th step, $k \in [0, \ldots, K]$, where $\mathcal{D}_q(\cdot)$ represents the distribution of the corresponding sample. More information regarding distortion risk functions and DM shall be given in Appendix A and B, respectively, available online.

B. Value Decomposition

Value decomposition methods [9], [10], [11], [12] belong to one of the typical methods to learn decentralized action value networks from globally shared rewards. As one of the pioneering works of value decomposition, QMIX [10] holds that under a monotonicity constraint, the joint of individually optimal actions is equivalent to the optimal joint action regarding the action value function, which is also a crucial criterion for MARL with distributed deployment of agents. That is,

$$\underset{\boldsymbol{a}}{\operatorname{argmax}}\, Q_{\text{global}}(\boldsymbol{o}, \boldsymbol{a}) = \begin{pmatrix} \underset{a^{[1]}}{\operatorname{argmax}}\, Q_1(o^{[1]}, a^{[1]}) \\ \vdots \\ \underset{a^{[N]}}{\operatorname{argmax}}\, Q_N(o^{[N]}, a^{[N]}) \end{pmatrix}, \tag{3}$$

where $Q_{\text{global}}(\boldsymbol{o}, \boldsymbol{a})$ indicates the joint action value function of all agents and $Q_i(o^{[i]}, a^{[i]})$ is the action value function corresponding to agent i. Besides, the monotonicity constraint can be formally written as

$$\frac{\partial Q_{\text{global}}(\boldsymbol{o}, \boldsymbol{a})}{\partial Q_i(o^{[i]}, a^{[i]})} \geq 0, \ \forall i = 1, 2, \ldots, N. \tag{4}$$

Furthermore, in order to satisfy (4), a mixing network is proposed to combine $Q_i(o^{[i]}, a^{[i]})$ into $Q_{\text{global}}(\boldsymbol{o}, \boldsymbol{a})$ with non-linear transformation.

C. Distributional RL

Distributional RL [16] aims to learn the distribution of action or state values, so as to capture more comprehensive information about random rewards (i.e., noisy rewards that follow a specific distribution). Besides, distributional RL benefits agents to hold different risk tendencies (e.g., risk-averse and risk-seeking) to choose preferred policies [22]. Specially for multi-agent distributional RL, given the state s and joint action $\boldsymbol{a}$, the transition operator P^π is defined as

$$P^\pi Z^\pi(s, \boldsymbol{a}) \overset{D}{:=} Z^\pi(S', \boldsymbol{A}'), \tag{5}$$

where the operator $U \overset{D}{:=} V$ indicates that the random variable U is distributed according to the same law as V. Consistent with (1), $Z^\pi(s, \boldsymbol{a})$ denotes a random variable with the expectation $Q^\pi(s, \boldsymbol{a})$, and capital letters S' and $\boldsymbol{A}'$ are used to emphasize the random nature of next state-action pair (i.e., $S' \sim \mathcal{P}(\cdot|s, \boldsymbol{a})$ and $\boldsymbol{A}' \sim \boldsymbol{\pi}(\cdot|S')$). On this basis, the distributional Bellman operator $\mathcal{T}^\pi$ [16] of MAS is defined as:

$$\mathcal{T}^\pi Z^\pi(s, \boldsymbol{a}) \overset{D}{:=} R(s, \boldsymbol{a}) + \gamma P^\pi Z^\pi(s, \boldsymbol{a}). \tag{6}$$

That is to say, $Z^\pi(s, \boldsymbol{a})$ is characterized by the interaction of three random variables (i.e., the reward $R(s, \boldsymbol{a})$, the next pair of state-action $(S', \boldsymbol{A}')$ and the corresponding $Z^\pi(S', \boldsymbol{A}')$ according to (5)). In distributional RL, (6) is used to update $Z^\pi(s, \boldsymbol{a})$ with the next state and action by temporal difference.

Accordingly, on account of the definition of $Z^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t)$ in (1) and the equivalence between s_t and $\boldsymbol{o}_t$, the random variable for the global discounted reward summation can be rewritten as

$$Z^{\boldsymbol{\pi}}_{\text{global},t}(\boldsymbol{o}_t, \boldsymbol{a}_t) = \sum_{m=0}^{\infty} \gamma^m R(\boldsymbol{o}_{t+m}, \boldsymbol{a}_{t+m}), \tag{7}$$

$R(\boldsymbol{o}_{t+m}, \boldsymbol{a}_{t+m})$ is a random variable of the noisy reward following the distribution $\mathcal{D}$, which can be approximately regarded as a GMM and decomposed into N Gaussian components.

On the other hand, it is critical to measure the distributional difference between the true distribution of noisy reward and the approximated one. In this regard, Wasserstein metric [21] can be leveraged and mathematically defined as

$$d_p(F, G) = \left(\int_0^1 |F^{-1}(u) - G^{-1}(u)|^p \mathrm{d}u\right)^{\frac{1}{p}}, \tag{8}$$

where F and G are the CDFs of random variables U and V, respectively. $p < \infty$ implies to take an L_p norm for this metric. Furthermore, in single-agent settings, some robust approaches have been proposed by using quantiles to evaluate and then fit the action value distribution [21], [22].

D. Problem Formulation

In this paper, we primarily consider N RL agents (i.e., a group of RL-empowered distributed robots equipped with intelligent sensing devices and basic computing power) to perform certain cooperative tasks as shown in Fig. 1. Consistent with the general settings in Dec-POMDP, at each time-step t, agent i capably

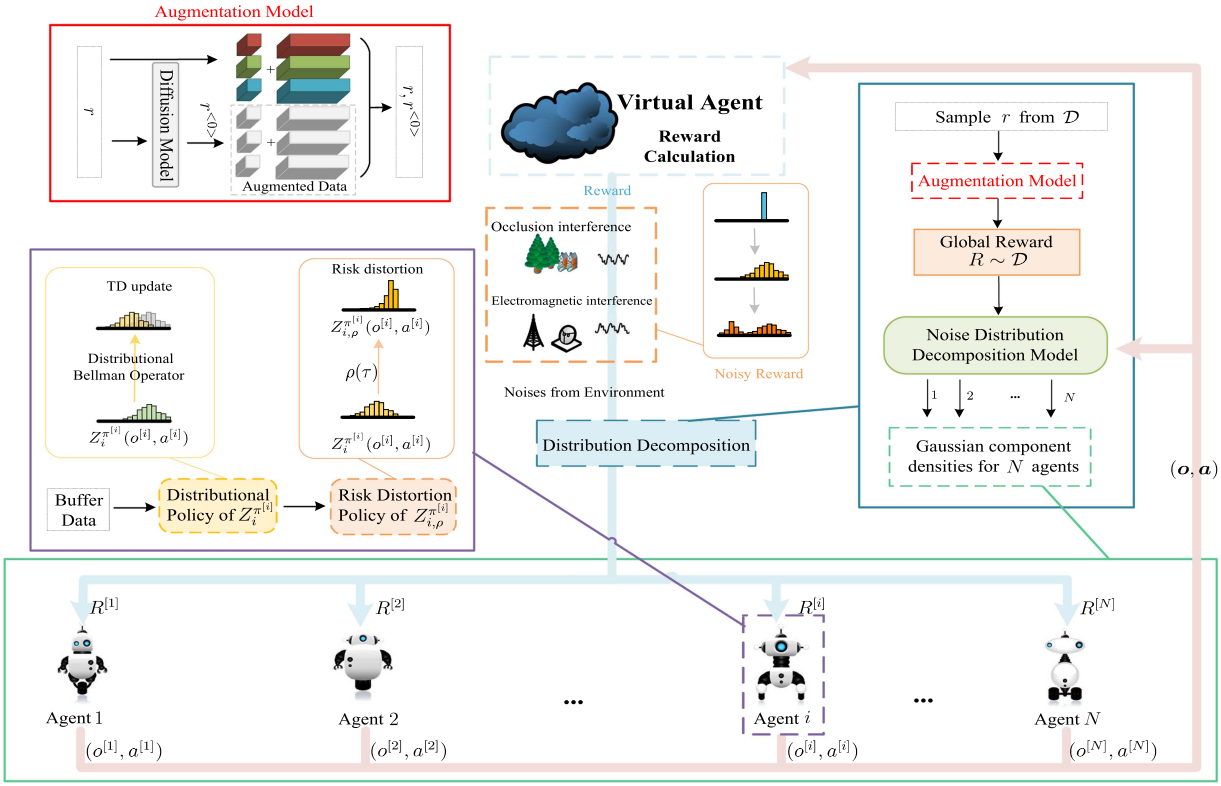


Fig. 1. The scenario and framework of NDD-based cooperative multi-agent distributional RL with noisy rewards.

utilizes on-board sensors to access a partial environmental observation $o_t^{[i]} \in \mathcal{O}^{[i]}$. Observations may include partial states of some teammates, received messages, or the relative positions to certain target points. This often depends on the specific task executed by the whole team. With a DM for data augmentation, every agent attempts to learn action value distributions and remaps them with distortion risk functions for yielding policy corresponding to different tasks. Based on the observation $o_t^{[i]}$, agent i responds with action $a_t^{[i]}$ (i.e., movement direction and message exchange) according to its policy $\pi^{[i]}(a_t^{[i]}|o_t^{[i]})$. Afterwards, the environmental state transits from s_t to next state s_{t+1} according to the transition function $\mathcal{P}(s_{t+1}|s_t, \boldsymbol{a}_t)$. Meanwhile, other observations and global information can be used for the calculation of reward from $\mathcal{R}$ by a suitable "virtual" agent (e.g., a centralized server or a computing-capable global-oriented robot). Notably, since the observation provided by sensors, or exactly the reward, could be polluted by internal or external noise, we formulate the reward by the random variable $R \sim \mathcal{D}$, consistent with Section III-A. More details are in Fig. 1. In the subsequent sections, targeted at the aforementioned MARL settings, we mainly discuss how to apply our method NDD to tackle the noisy rewards, so as to maximize the average reward of episodes for the entire team.

IV. Noise Distribution Decomposition for MARL

In this section, particularly focusing on general MARL settings, we provide details of NDD toward tackling the globally shared noisy rewards with improved learning accuracy and stability. Specifically, following the idea of distributional RL, we choose the action value distribution (i.e., the distribution that $Z^{\boldsymbol{\pi}}_{\text{global}}(\boldsymbol{o}, \boldsymbol{a})$ in (7) follows) rather than the action value function to update agents' joint policy $\boldsymbol{\pi}$. As mentioned earlier, it will also be intractable to directly decompose the distribution of the global $Z^{\boldsymbol{\pi}}_{\text{global}}(\boldsymbol{o}, \boldsymbol{a})$. Thus, we further approximate the distribution of the global reward as a GMM [26] and propose to decompose the noisy reward distribution into individual parts, so as to facilitate the update of local distributions. Meanwhile, to ensure the consistency between local and global action values, we also provide theoretical details of the distribution decomposition and present the alternative functions for distortion risk measures. Finally, DM is introduced to enhance buffer data, and the corresponding generation and approximation errors are analyzed.

A. GMM-Based Distributional MARL

As a parametric probability density function (PDF), a GMM represents a weighted sum of Gaussian component densities [26]. Mathematically, taking the example of any scalar x from random variable $X \sim \texttt{GMM}(\mu_1, \ldots, \mu_N, \sigma_1^2, \ldots, \sigma_N^2, w^{[1]}, \ldots, w^{[N]})$, the corresponding PDF can be expressed as

$$f(x; \mu_1, \ldots, \mu_N, \sigma_1^2, \ldots, \sigma_N^2, w^{[1]}, \ldots, w^{[N]}) = \sum_{i=1}^{N} w^{[i]}\, g(x; \mu_i, \sigma_i^2), \tag{9}$$

where $w^{[1]}, \ldots, w^{[N]}$ are mixture weights with $\sum_{i=1}^{N} w^{[i]} = 1$. $g(x; \mu_i, \sigma_i^2)$ denotes the PDF of the Gaussian distribution with mean μ_i and variance σ_i^2. According to Wiener's Tauberian theorem [52], [53], GMM, which has density completeness, can essentially fit the shape of arbitrary distribution with absolutely integrable PDF to describe data from single or multiple noise sources [54], so $\mathcal{D}$ can also be approximated as $\hat{\mathcal{D}}$ which is a GMM. Thus, $\hat{\mathcal{D}}$ can be further decomposed into Gaussian components $\hat{\mathcal{D}}_l^{[1]}(\theta^{[1]}), \ldots, \hat{\mathcal{D}}_l^{[N]}(\theta^{[N]})$ corresponding to different individual agents.

Without loss of generality, let $R_l^{[1]}, \ldots, R_l^{[N]}$ be independent decomposed local rewards where $R_l^{[i]} \sim \hat{\mathcal{D}}_l^{[i]}(\theta^{[i]})$. We further define a function $\Psi(R_l^{[1]}, \ldots, R_l^{[N]}; w^{[1]}, \ldots, w^{[N]})$ to statistically combine $R_l^{[i]}$ with an assigned probability of $w^{[i]}$, and thus $\Psi(R_l^{[1]}, \ldots, R_l^{[N]}; w^{[1]}, \ldots, w^{[N]})$ conforms to a GMM with its expectation equaling to $\sum_{i=1}^{N} w^{[i]} \mathbb{E}(R_l^{[i]})$. In other words, it satisfies the additive property. Hence, the global reward $R(\boldsymbol{o}_{t+m}, \boldsymbol{a}_{t+m})$ in (7) approximately equals $\Psi(R_l^{[1]}, \ldots, R_l^{[N]}; w_m^{[1]}, \ldots, w_m^{[N]})$, and (7) can be rewritten as (10) shown at the bottom of this page, with $\mathbb{E}[Z^{\boldsymbol{\pi}}_{\text{global},t}(\boldsymbol{o}_t, \boldsymbol{a}_t)] = Q^{\boldsymbol{\pi}}_{\text{global},t}(\boldsymbol{o}_t, \boldsymbol{a}_t)$, which is action value of the whole MAS in t-th step given joint observation, action and policy $\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{\pi}$. Furthermore, the local $Z_{i,t}^{\pi^{[i]}}(o_t^{[i]}, a_t^{[i]})$ of agent i can be represented as

$$Z_{i,t}^{\pi^{[i]}}(o_t^{[i]}, a_t^{[i]}) = \sum_{m=0}^{\infty} \gamma^m R_l^{[i]}(o_{t+m}^{[i]}, a_{t+m}^{[i]}). \tag{11}$$

For simplicity of representation, we might omit $\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{\pi}, t$ in $Z^{\boldsymbol{\pi}}_{\text{global},t}(\boldsymbol{o}_t, \boldsymbol{a}_t)$ and $o_t^{[i]}, a_t^{[i]}, \pi^{[i]}, t$ in $Z_{i,t}^{\pi^{[i]}}(o_t^{[i]}, a_t^{[i]})$, and only use the notations Z_{global} and Z_i.

B. Consistency of Global and Local Optimal Actions

First, we show that the monotonicity constraint mentioned in Section III-B holds for the GMM-based distributional MARL on some conditions.

Theorem 1: For any $i = 1, \ldots, N$ and Z_i defined by (11) in any m-th step, if $w_m^{[i]} \geq 0$ we have

$$\frac{\partial \mathbb{E}(Z_{\text{global}})}{\partial \mathbb{E}(Z_i)} \geq 0. \tag{12}$$

Proof: As the definition in (10), the expectation of Z_{global} computed from M time-steps can be expressed as

$$\mathbb{E}\left[Z_{\text{global}}\right] = \text{tr}(\boldsymbol{W}^\top \boldsymbol{R}^{\#}), \tag{13}$$

where

$$\boldsymbol{W} = \begin{bmatrix} w_0^{[1]} & \ldots & w_0^{[N]} \\ \vdots & \vdots & \vdots \\ w_M^{[1]} & \ldots & w_M^{[N]} \end{bmatrix}, \tag{14}$$

$$\boldsymbol{R}^{\#} = \begin{bmatrix} \gamma^0 \mathbb{E}(R_{l,0}^{[1]}) & \ldots & \gamma^0 \mathbb{E}(R_{l,0}^{[N]}) \\ \vdots & \vdots & \vdots \\ \gamma^M \mathbb{E}(R_{l,M}^{[1]}) & \ldots & \gamma^M \mathbb{E}(R_{l,M}^{[N]}) \end{bmatrix}, \tag{15}$$

and $R_{l,t}^{[i]}$ is the abbreviation of $R_l^{[i]}(o_t^{[i]}, a_t^{[i]})$. Similarly, the local Z_i can be formulated as

$$\left[\mathbb{E}(Z_1) \ldots \mathbb{E}(Z_N)\right] = \mathbf{1}^\top_{(M+1)\times 1} \boldsymbol{R}^{\#}. \tag{16}$$

Taking $\kappa = \min(w_m^{[i]})$, $\sum_i w_m^{[i]} = 1$ $(\forall t)$ implies $0 \leq \kappa \leq \frac{1}{N}$. Therefore, (13) can be rewritten as

$$\begin{aligned} &\mathbb{E}\left[Z_{\text{global}}\right] = \text{tr}(\boldsymbol{W}^\top \boldsymbol{R}^{\#}) \\ &\geq \kappa \cdot \text{tr}(\mathbf{1}^\top_{(M+1)\times N} \boldsymbol{R}^{\#}) = \kappa \cdot \mathbf{1}^\top_{(M+1)\times 1} \boldsymbol{R}^{\#} \mathbf{1}_{N\times 1} \\ &= \kappa \cdot \left[\mathbb{E}(Z_1) \quad \ldots \quad \mathbb{E}(Z_N)\right] \mathbf{1}_{N\times 1}. \end{aligned} \tag{17}$$

Consequently, (12) is satisfied as $\frac{\partial \mathbb{E}(Z_{\text{global}})}{\partial \mathbb{E}(Z_i)} \geq \kappa \geq 0$. ■

Recalling that $\mathbb{E}[Z_{\text{global}}] = Q_{\text{global}}(\boldsymbol{o}, \boldsymbol{a})$ and $\mathbb{E}[Z_i] = Q_i(o^{[i]}, a^{[i]})$, together with (3), we have following corollary.

Corollary 1:

$$\text{argmax}_{\boldsymbol{a}} \mathbb{E}\left[Z_{\text{global}}\right] = \begin{pmatrix} \text{argmax}_{a^{[1]}} \mathbb{E}\left[Z_1\right] \\ \vdots \\ \text{argmax}_{a^{[N]}} \mathbb{E}\left[Z_N\right] \end{pmatrix}. \tag{18}$$

Remark: The proof of Theorem 1 implies that some excessive small weights mean trivial contributions to the group. Meanwhile, from the perspective of minimizing the reward gap between an ideal policy with full observability and partially observable local policies [55], a more uniform distribution of weights can decrease this gap between $\mathbb{E}(Z_{\text{global}})$ and its lower bound $\kappa \cdot \left[\mathbb{E}(Z_1) \quad \ldots \quad \mathbb{E}(Z_N)\right] \mathbf{1}_{N\times 1}$. Especially, for cases where weights are evenly divided among agents (i.e., $w_m^{[i]} = \kappa = \frac{1}{N}, \forall i = 1, \ldots, N$), $\mathbb{E}(Z_{\text{global}}) = \frac{1}{N} \sum_i \mathbb{E}[Z_i]$, which also satisfies (12). However, only enforcing such a strong constraint on weights will incur large errors in decomposition results. Thus, it is meaningful to carefully calibrate the design of the loss function by appending additional constraint terms.

C. Design of Distortion Risk Function

As mentioned in [22], the distortion risk function $\rho(\tau)$ enhances the applicability and flexibility of distributional RL. Based on $\rho(\tau)$, Dabney et al. [22] introduce an parameter η to make $\rho_\eta(\tau)$ more flexible and adaptable. However, for MARL, $\rho_\eta(\tau)$ needs to be subjected to certain constraints in order to ensure that the distorted action value distribution also satisfies the monotonicity constraint in Theorem 1.

$$Z^{\boldsymbol{\pi}}_{\text{global},t}(\boldsymbol{o}_t, \boldsymbol{a}_t) = \sum_{m=0}^{\infty} \gamma^m \Psi\left[R_l^{[1]}(o_{t+m}^{[1]}, a_{t+m}^{[1]}), \ldots, R_l^{[N]}(o_{t+m}^{[N]}, a_{t+m}^{[N]}); w_m^{[1]}, \ldots, w_m^{[N]}\right]. \tag{10}$$

Theorem 2: Given $\frac{\partial \mathbb{E}(Z_{\text{global}})}{\partial \mathbb{E}(Z_i)} \geq 0$, if $\rho'_\eta(\tau) \in (0, +\infty)(\forall \tau)$ we have

$$\frac{\partial \mathbb{E}(Z_{\text{global},\rho})}{\partial \mathbb{E}(Z_{i,\rho})} \geq 0. \tag{19}$$

Here $Z_{\text{global},\rho}$ and $Z_{i,\rho}$ represent that Z_{global}, Z_i have undergone the remap distortion $\rho_\eta(\tau)$.

Proof: According to (2),

$$\begin{aligned}\frac{\partial \mathbb{E}(Z_{\text{global},\rho})}{\partial \mathbb{E}(Z_{i,\rho})} &= \frac{\partial\{\int z \mathrm{d}\rho_\eta\left[F_{\text{global}}(z)\right]\}}{\partial\{\int z \mathrm{d}\rho_\eta\left[F_i(z)\right]\}} \\ &= \frac{\partial\left[\int z\rho'_\eta(\tau_{\text{global}})\mathrm{d}F_{\text{global}}(z)\right]}{\partial\left[\int z\rho'_\eta(\tau_i)\mathrm{d}F_i(z)\right]},\end{aligned} \tag{20}$$

where F_{global}, F_i denote the CDFs of Z_{global}, Z_i; and $\tau_{\text{global}}, \tau_i$ are their quantiles, respectively. Based on the positive-bound assumption of $\rho'_\eta(\tau)(\forall \tau)$, let $\rho'_\eta(\tau) \in [\rho'_{\min}, \rho'_{\max}]$, (20) can be rewritten as

$$\begin{aligned}\frac{\partial \mathbb{E}(Z_{\text{global},\rho})}{\partial \mathbb{E}(Z_{i,\rho})} &\geq \frac{\rho'_{\min}}{\rho'_{\max}} \frac{\partial\left[\int z\mathrm{d}F_{\text{global}}(z)\right]}{\partial\left[\int z\mathrm{d}F_i(z)\right]} \\ &= \frac{\rho'_{\min}}{\rho'_{\max}} \frac{\partial \mathbb{E}(Z_{\text{global}})}{\partial \mathbb{E}(Z_i)}.\end{aligned} \tag{21}$$

Equation (19) holds obviously when $\rho'_\eta(\tau) \in (0, +\infty)$. ■

Remark: Intuitively, the distortion risk functions including CPW [56], [57], WANG [58], POW [22], and CVaR [59] given in Appendix A, available online, satisfy the condition (i.e., $\rho'_\eta(\tau) \in (0, +\infty)$), when they take the values in Table VIII as [22]. Furthermore,

$$\text{argmax}_{\boldsymbol{a}}\, \mathbb{E}\left(Z_{\text{global},\rho}\right) = \begin{pmatrix} \text{argmax}_{a^{[1]}}\, \mathbb{E}\left(Z_{1,\rho}\right) \\ \vdots \\ \text{argmax}_{a^{[N]}}\, \mathbb{E}\left(Z_{N,\rho}\right) \end{pmatrix} \tag{22}$$

holds. Thus, it lays the very foundation to adopt these distortion risk measures in our case.

In summary, based on Theorems 1 and 2, the monotonicity of the reward distribution decomposition is theoretically validated under nonnegative $w^{[i]}(i = 1, \ldots, N)$ and increasing $\rho_\eta(\tau)$. From this monotonicity, the collective performance of MAS will not deteriorate when every agent improves its performance, with consistent global and local convergences.

D. DM-Based Data Augment

The action value distribution reflects more risk information than the action value function, but the interaction cost increases between agents and environment [16]. Especially in MAS, additional interaction demands several times greater than that of a single agent. Hence, using a generative model for data augmentation becomes a feasible compromise [45], [46]. Besides, in order to maintain good compatibility with GMM, DM [27] is used in the NDD, as Theorems 3 and 4 demonstrate bounded generation error and approximation error, which measure the distance between the generated sample and the original sample or the GMM, respectively.

Theorem 3 (Bounded Generation Error): Under the assumption that $K = 25$ and $[\omega_1, \ldots, \omega_K] = 0.499 \times 10^{-2} \times \frac{1}{1+e^{-\boldsymbol{h}}} + 10^{-5}$, where $\boldsymbol{h}$ is an array with a length of K evenly spaced between -6 and 6, the expectation of generation error $\mathbb{E}[(r^{<0>} - r)^2]$ satisfies

$$\mathbb{E}\left[(r^{<0>} - r)^2\right] \lesssim \left(\mathbb{E}(r)\right)^2 + 5.1359\mathbb{D}(r). \tag{23}$$

Theorem 4 (Bounded Approximation Error): Under the assumption consistent with Theorem 3, the upper bound of approximation error expectation for $r^{<0>}$ is less than that for r, where the approximation error ξ for r is defined as the gap between r and the GMM (i.e., $r = r_{\texttt{GMM}} + \xi$) with $r_{\texttt{GMM}} \sim \texttt{GMM}(\cdot)$.

We leave the proofs in Appendix C and D, available online. Moreover, Theorems 3 and 4 show that the gradual inclusion of Gaussian components aligns well with GMM used in the NDD, making the integration of DM with NDD a prudent approach, and further imply how to select appropriate hyperparameters (e.g. diffusion steps K).

Next, we elaborate on the combination between DM and NDD. First, as shown in Appendix B, available online, DM is employed to learn the distribution $\mathcal{D}$ of reward R via its sample r and generate $r^{<0>}$. Afterwards, NDD approximates augmented data to GMM and decomposes it into several Gaussian components for N agents. Each agent updates its policy with distributional RL and certain distortion risk functions. The complete process is visualized in Fig. 1.

E. Loss Function Design

As mentioned before, the NDD networks decompose the globally shared reward into weighted individual rewards for cooperative agents by estimating the distributions of decomposed rewards. Therefore, the estimation error, which can be regarded as a gap between the approximated PDF and true PDF of the globally shared reward, shall be minimized.

Without loss of generality, let P and $\hat{P}$ be the PDFs of the true reward distribution $\mathcal{D}$ and the estimated one $\hat{\mathcal{D}}$ through the NDD networks respectively. Based on the Wasserstein metric in (8), the loss function for NDD as

$$\mathcal{L}_{\text{PDF}}(\boldsymbol{\theta}, \boldsymbol{w}) = \int_{r_{\min}}^{r_{\max}} \left[P(u) - \hat{P}(u; \boldsymbol{\theta})\right]^2 \mathrm{d}u, \tag{24}$$

where $[r_{\min}, r_{\max}]$ denotes the range of the reward distribution. Except the parameters Θ for DM, $\boldsymbol{\theta} = [\theta^{[1]}, \ldots, \theta^{[N]}]$ indicates all parameters of NDD networks corresponding to each agent and $\boldsymbol{w} = [w^{[1]}, \ldots, w^{[N]}]^\top$. Here both $\boldsymbol{\theta}$ and $\boldsymbol{w}$ are the trainable parameters in the NDD networks.

On the other hand, there may exist many possible candidates for decomposed Gaussian distributions $\hat{\mathcal{D}}_l^{[i]}(\theta^{[i]})$, $\forall i = 1, \ldots, N$. To alleviate the potential issue of multiple solutions in the decomposition operation, we further impose another penalty term as

$$\mathcal{L}_{\text{mean}}(\boldsymbol{\theta}) = (\boldsymbol{X} - \mu \cdot \mathbf{1}_{N\times 1})^\top (\boldsymbol{X} - \mu \cdot \mathbf{1}_{N\times 1}), \tag{25}$$

where $\boldsymbol{X} = [\mu_1(\theta^{[1]}), \ldots, \mu_N(\theta^{[N]})]^\top$ and $\mu = \sum_{i=1}^N \mu_i(\theta^{[i]})$, where $\mu_i(\theta^{[i]})$ denotes the mean of $\hat{\mathcal{D}}_l^{[i]}(\theta^{[i]})$.

Algorithm 1: The Centralized Training Procedure for NDD.

Input: Agents from 1 to N, the maximum iteration number T, trajectory buffer $\mathcal{B}$, the error threshold $e_{\min}$ and hyperparameters λ, α.
Output: $\boldsymbol{\theta} = [\theta^{[1]}, \cdots, \theta^{[N]}]$ for agents $i = 1, \cdots, N$.

Initialize $\boldsymbol{\theta}$ and $\boldsymbol{w}$ randomly;
Initialize $\mathcal{B}$; initialize joint policy $\boldsymbol{\pi} = [\pi^{[1]}, \cdots, \pi^{[N]}]$;
for *iteration* $= 1$ *to* T **do**
  while $\mathcal{B}$ *is NOT full* **do**
    for $i = 1$ *to* N **do**
      $a^{[i]} \sim \pi^{[i]}(\cdot|o^{[i]})$;
    end
    Execute $\boldsymbol{a} = [a^{[1]}, \cdots, a^{[N]}]$ and collect set of trajectories into $\mathcal{B}$;
  end
  Generate $r^{<0>}$ by DM in Algorithm 2 when given r;
  $r \leftarrow (r, r^{<0>})$, $e \leftarrow +\infty$;
  while $e > e_{\min}$ **do**
    for $i = 1$ *to* N **do**
      Generate $\mu_i(\theta^{[i]})$ and $\sigma_i(\theta^{[i]})$ of $\hat{\mathcal{D}}_l^{[i]}(\theta^{[i]})$ based on $\langle o^{[i]}, a^{[i]} \rangle$ by the local distribution network;
      Calculate the Gaussian PDF $\hat{P}^{[i]}$ of $R_l^{[i]} \sim \hat{\mathcal{D}}_l^{[i]}(\theta^{[i]})$;
    end
    $\boldsymbol{w} \leftarrow \text{Softmax}(\boldsymbol{w})$;
    Concatenate the generated means as $\boldsymbol{X} = [\mu_1(\theta^{[1]}), \ldots, \mu_N(\theta^{[N]})]^\top$;
    Calculate the PDF of the estimated distribution $\hat{\mathcal{D}}$ in the form of a GMM as $\hat{P} = \sum_{i=1}^{N} w^{[i]} \hat{P}^{[i]}$;
    Update $\boldsymbol{\theta}$ and $\boldsymbol{w}$ by minimizing the loss function as in (27);
    Calculate $e = \mathcal{L}_{\text{PDF}}(\boldsymbol{\theta}, \boldsymbol{w})$ according to (24);
  end
  Clear $\mathcal{B}$;
end

For an individual agent, the weight represents its degree of involvement in the decomposition task. Certain excessively small weight indicates that $\mathcal{L}_{\text{PDF}}$ has minimal supervisory impact on that component and may result in insufficient stability of the Gaussian component after decomposition. This becomes particularly evident in some simple noise scenarios, such as when the noise can be represented by the superposition of a few Gaussian components. In such cases, the remaining agents may not receive effective components. With excessively low weights, these agents might exploit extremely deviant reward distributions in training policies, ultimately affecting their performances. In addition, in order to narrow the disparity between $\mathbb{E}(Z_{\text{global}})$ and its lower bound, as mentioned in Section IV-B, weights can be aligned more evenly. Therefore, we define the

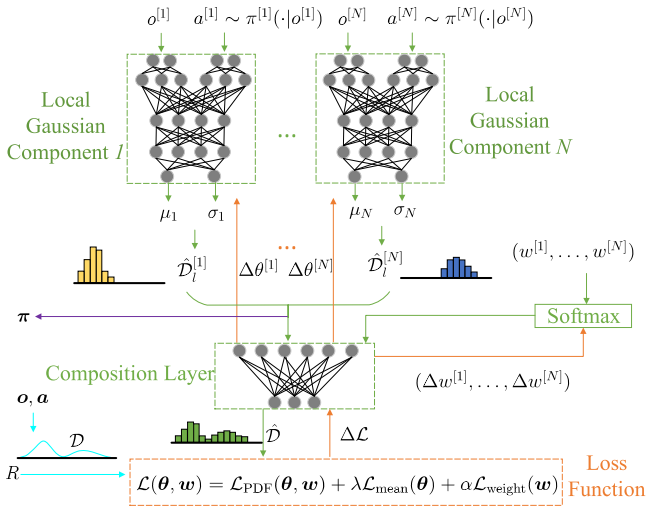


Fig. 2. Overview of the NDD algorithm.

corresponding penalty term as

$$\mathcal{L}_{\text{weight}}(\boldsymbol{w}) = \|\boldsymbol{w} - \frac{1}{N} \cdot \mathbf{1}_{N\times 1}\|_2. \tag{26}$$

To sum up, given hyperparameters λ and α, we obtain the Wasserstein metric-based loss function as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{w}) = \mathcal{L}_{\text{PDF}}(\boldsymbol{\theta}, \boldsymbol{w}) + \lambda \mathcal{L}_{\text{mean}}(\boldsymbol{\theta}) + \alpha \mathcal{L}_{\text{weight}}(\boldsymbol{w}). \tag{27}$$

In a nutshell, for the NDD algorithm depicted in Fig. 2, we first predict local Gaussian components that will be utilized for local training by a multi-layer perceptron (MLP), and aggregate these decomposed ones through another MLP-based composition layer, so as to approximate the GMM along original and augmented rewards. Next we update $\boldsymbol{\theta}$ and $\boldsymbol{w}$ according to (27) and learn $\pi^{[i]}$ with $\hat{\mathcal{D}}_l^{[i]}(\theta^{[i]})$ through distributional RL independently. Details are summarized in Algorithm 1.

V. Experimental Settings and Simulation Results

A. Experimental Settings

We evaluate NDD in Multi-agent Particle world Environments (MPE) and StarCraft Multi-Agent Challenge (SMAC) [60], [61], [62], which is implemented on top of the current SOTA algorithm MultiAgent Proximal Policy Optimization (MAPPO) [63], [64], [65]. MPE is a classical MARL environment with each particle emulating the potential behaviors of one agent. Typically, MPE consists of several task-oriented collaboration scenarios (e.g., Adversary, Crypto, Speaker-Listener, Spread and Reference). Compared to MPE, SMAC is a more complicated environment based on Blizzard's StarCraft II RTS game. Notably, as shown in Table VI, we add 5 types of noise (i.e., Noise 0 to Noise 5) to the globally shared rewards, which are different in MPE and SMAC scenarios so as to be in line with the distinct reward values (i.e., one to two orders of magnitude smaller in SMAC than MPE). Additionally, we choose some typical actor cirtic algorithms MADDPG [61], MATD3 [66] and value decomposition algorithms VDN [9], QMIX [10] as well as MAPPO for

TABLE II
PERFORMANCE OF NDD AND OTHERS WITH FIVE TYPES OF NOISES IN MPE TASKS

Scenario	Type	VDN	QMIX	MATD3	MADDPG	MAPPO	NDD	Baseline
Adversary	Noise 0	−25.63±0.78	−37.39±3.40	−9.67 ± 3.50	−14.81 ± 1.80	−8.23±0.66	**−1.47±0.49**	−2.39±0.22
	Noise 1	−24.24±1.13	−46.17±5.61	−15.88 ± 2.19	−13.32 ± 2.43	−6.16±0.78	**−1.51±0.19**	−1.91±0.35
	Noise 2	−23.72±1.57	−40.39±17.30	−16.21 ± 3.12	−19.33 ± 0.42	−7.76±0.78	**−0.89±0.24**	−1.82±0.10
	Noise 3	−18.12 ± 0.99	−43.18 ± 6.51	−6.97 ± 1.27	−18.49 ± 3.67	−4.90 ± 0.22	**−1.42 ± 0.75**	−1.74 ± 0.69
	Noise 4	−25.72 ± 0.36	−25.79 ± 3.89	−11.04 ± 10.21	−53.30 ± 32.57	−9.72 ± 0.62	**−4.74 ± 1.19**	−0.93 ± 0.19
Crypto	Noise 0	−66.97±3.03	−69.56±0.94	−13.40 ± 7.11	**−1.20 ± 28.79**	−28.90±1.00	−11.77±2.00	−3.68±0.97
	Noise 1	−69.11±3.29	−55.12±13.06	**1.20 ± 16.29**	−4.21 ± 4.08	−22.51±0.47	−5.28±0.68	−8.67±0.63
	Noise 2	−68.29±0.38	−69.35±2.17	−20.45 ± 21.56	−17.84 ± 15.95	−24.84±1.01	**−10.69±0.08**	−2.43±0.13
	Noise 3	−67.46 ± 2.04	−49.68 ± 12.31	**−0.05 ± 2.63**	−12.19 ± 3.29	−19.60 ± 1.42	−19.47 ± 1.25	−1.31 ± 0.08
	Noise 4	−71.94 ± 3.17	−65.22 ± 3.35	−0.60 ± 21.54	**0.60 ± 4.39**	−28.56 ± 3.32	−27.36 ± 1.10	−5.43 ± 2.35
Speaker Listener	Noise 0	−34.98±1.32	−33.66±0.86	−53.08 ± 13.12	−46.36 ± 1.45	**−31.81±3.67**	−33.40±1.39	−31.80±2.79
	Noise 1	−34.52±0.99	**−28.75±2.42**	−48.16 ± 15.37	−45.93 ± 1.05	−34.03±0.33	−29.76±0.93	−32.66±2.46
	Noise 2	−34.53±1.45	−34.06±2.19	−51.03 ± 11.95	−41.52 ± 0.87	−34.88±2.74	**−32.42±0.96**	−28.95±0.47
	Noise 3	−35.51 ± 0.42	**−28.61 ± 0.77**	−62.00 ± 4.94	−44.26 ± 9.32	−32.63 ± 0.78	−33.22 ± 0.45	−31.66 ± 0.41
	Noise 4	−34.90 ± 1.99	−38.01 ± 0.64	−67.28 ± 1.18	−47.03 ± 1.56	−37.33 ± 2.17	**−33.70 ± 0.28**	−30.26 ± 2.05
Spread	Noise 0	−104.84±0.36	−171.02±7.04	−114.46 ± 3.06	−106.85 ± 0.64	**−93.58±0.68**	−94.94±0.52	−91.64±0.69
	Noise 1	−98.90±0.87	−156.00±19.36	−104.02 ± 3.68	−102.49 ± 0.37	−97.86±5.23	**−93.66±2.27**	−94.34±0.83
	Noise 2	−97.59±1.32	−171.36±16.49	−103.27 ± 1.68	−104.44 ± 2.58	−97.91±1.05	**−91.58±0.75**	−92.62±0.87
	Noise 3	−95.05 ± 0.49	−139.44 ± 11.54	−105.26 ± 0.46	−101.05 ± 3.19	**−91.97 ± 1.07**	−94.54 ± 0.19	−93.19 ± 0.73
	Noise 4	−98.37 ± 1.83	−190.89 ± 7.23	−99.37 ± 1.05	−107.87 ± 1.31	−96.00 ± 1.66	**−93.29 ± 1.35**	−91.33 ± 0.28
Reference	Noise 0	−43.28±0.67	−76.25±4.97	−48.16 ± 1.58	−52.35 ± 0.61	−40.65±0.52	**−36.33±0.35**	−36.14±0.11
	Noise 1	−40.49±0.62	−58.67±9.02	−68.77 ± 14.50	−51.22 ± 1.32	−39.07±0.35	**−37.44±0.92**	−36.09±0.73
	Noise 2	−42.41±0.73	−62.82±10.49	−49.53 ± 1.35	−46.13 ± 2.48	−37.42±0.66	**−37.29±1.54**	−36.25±0.20
	Noise 3	−40.05 ± 0.13	−66.59 ± 8.74	−47.79 ± 0.33	−48.05 ± 2.68	−38.88 ± 0.44	**−35.76 ± 0.55**	−36.49 ± 1.02
	Noise 4	−42.56 ± 0.19	−72.21 ± 4.53	−49.18 ± 4.54	−50.92 ± 1.72	−36.57 ± 0.73	**−36.26 ± 1.02**	−35.62 ± 0.16

The "baseline" is the result of MAPPO without noise.

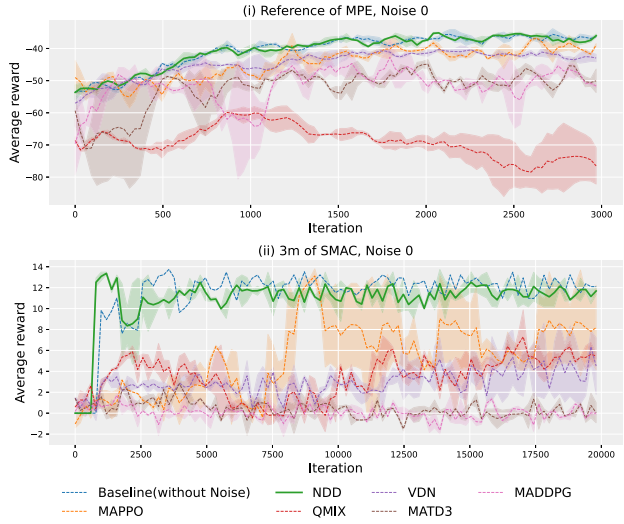


Fig. 3. Performance comparison between algorithms in (i) the MPE Reference task with Noise 0; (ii) the SMAC 3m task with Noise 0. Notably, the "Baseline" indicates the result of MAPPO under noise-free settings; "3m" means 3 Marines in MAS.

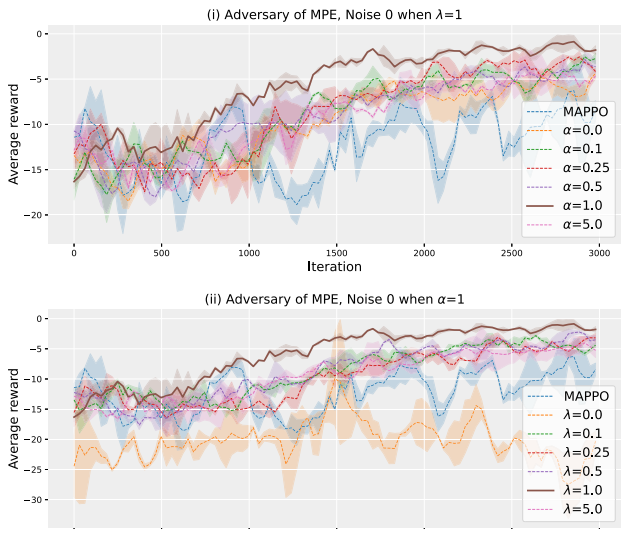


Fig. 4. Performance sensitivity of NDD in Adversary with Noise 0 under the settings of (i) different α with $\lambda = 1$; (ii) different λ with $\alpha = 1$.

comparison. Furthermore, we choose MAPPO under noiseless conditions as the "Baseline" to measure the performance loss of NDD and the others in noisy environments. Considering that the number of iterations is generally based on empirical values and changes over environments, every method is trained with 3,000 iterations for each task in MPE, and 2×10^4 iterations for each task in SMAC. Afterwards, we compare the performance under different types of zero-mean noises to rule out the possible bias impact of noise. More experimental configurations are given in Appendix E, available online.

B. Performance Comparison

Fig. 3 compares NDD with the aforementioned methods and visualizes the corresponding learning process of agents for one MPE task and one SMAC task. Meanwhile, Tables II and III summarize the performance comparisons in all noisy tasks considering five distinctive types of noise as in Table VI. It's worth noting that there are slight differences in the five simulations' results of the "Baseline" over five types of noise for each task in MPE, as the states of agents are randomly initialized at the beginning of each episode, which is in stark

TABLE III
PERFORMANCE OF NDD AND OTHERS WITH FIVE TYPES OF NOISES IN SMAC TASKS

Scenario	Type	VDN	QMIX	MATD3	MADDPG	MAPPO	NDD	Baseline
3m	Noise 0	4.69±2.22	5.74±1.17	0.39 ± 1.16	−0.14 ± 0.32	8.27±3.64	**11.64±0.83**	12.11±0.30
	Noise 1	5.92±1.12	6.52±1.63	−0.16 ± 0.35	0.10 ± 0.45	8.07±2.25	**11.49±1.16**	
	Noise 2	4.98±2.31	2.37±0.74	0.06 ± 0.41	0.01 ± 0.53	**11.84±1.35**	8.47±1.73	
	Noise 3	7.06 ± 1.84	7.75 ± 1.08	0.13 ± 0.41	−0.02 ± 0.16	**11.63 ± 1.10**	11.12 ± 0.73	
	Noise 4	7.70 ± 1.67	6.41 ± 1.61	−3.33 ± 0.36	−3.62 ± 0.25	10.91 ± 0.77	**12.79 ± 0.68**	
8m	Noise 0	**4.55±0.75**	3.39±1.60	1.14 ± 1.47	0.24 ± 1.60	0.42±2.51	0.48±0.49	4.98±0.40
	Noise 1	2.97±0.60	6.42±0.93	0.45 ± 0.50	0.80 ± 0.49	0.86±1.72	**6.84±4.13**	
	Noise 2	2.57±0.61	3.92±0.64	0.72 ± 0.61	0.59 ± 0.70	3.13±1.36	**10.90±0.81**	
	Noise 3	**3.26 ± 0.75**	2.60 ± 0.73	0.56 ± 0.37	0.26 ± 0.40	1.19 ± 1.25	2.95 ± 1.38	
	Noise 4	0.82 ± 0.96	2.80 ± 3.28	−6.73 ± 0.24	−6.71 ± 0.32	3.91 ± 1.76	**8.38 ± 2.25**	
2m vs 1z	Noise 0	2.60±0.77	2.87±0.68	0.92 ± 1.45	0.35 ± 1.04	3.51±0.17	**3.65±0.09**	3.32±0.01
	Noise 1	2.62±1.20	2.70±0.44	0.21 ± 0.77	−0.06 ± 0.46	3.40±0.23	**3.46±0.06**	
	Noise 2	2.67±0.74	2.41±0.79	0.34 ± 0.53	0.20 ± 0.99	3.31±0.47	**3.59±0.05**	
	Noise 3	1.97 ± 1.01	2.28 ± 0.45	0.41 ± 0.51	0.14 ± 0.29	3.18 ± 0.69	**3.68 ± 0.08**	
	Noise 4	−7.74 ± 1.11	−6.56 ± 0.58	−8.96 ± 0.64	−9.01 ± 0.53	2.28 ± 0.26	**3.55 ± 0.04**	
2s3z	Noise 0	2.72±0.98	3.28±2.43	2.12 ± 1.12	1.68 ± 1.24	2.97±1.00	**7.72±0.40**	6.83±0.33
	Noise 1	3.11±0.63	2.29±0.61	0.71 ± 0.47	0.70 ± 0.38	1.52±0.33	**7.21±0.46**	
	Noise 2	3.36±0.69	2.02±0.47	1.16 ± 0.67	0.84 ± 0.71	6.51±1.75	**7.49±0.29**	
	Noise 3	3.33 ± 0.86	2.43 ± 0.60	1.13 ± 0.25	0.42 ± 0.39	3.40 ± 3.01	**5.94 ± 0.78**	
	Noise 4	−1.60 ± 0.48	−1.33 ± 0.76	−5.48 ± 0.80	−6.06 ± 1.08	2.75 ± 1.07	**6.91 ± 0.39**	
2s vs 1sc	Noise 0	0.14±1.22	−0.14±1.16	0.13 ± 1.50	−0.17 ± 0.37	6.42±0.59	**6.46±0.02**	6.44±0.01
	Noise 1	0.37±1.22	−0.67±0.98	−0.13 ± 1.04	−0.07 ± 0.99	6.25±0.28	**6.44±0.02**	
	Noise 2	0.07±1.51	0.49±0.69	0.02 ± 1.10	0.29 ± 0.79	6.29±0.43	**6.44±0.03**	
	Noise 3	−0.05 ± 0.91	0.20 ± 0.71	0.14 ± 0.49	−0.13 ± 0.28	6.39 ± 0.27	**6.43 ± 0.03**	
	Noise 4	−17.70 ± 0.59	−10.33 ± 7.55	−17.62 ± 0.39	−10.28 ± 7.58	3.78 ± 0.19	**6.45 ± 0.02**	
3s vs 3z	Noise 0	3.20±1.46	0.91±1.64	2.47 ± 1.01	1.90 ± 1.30	2.76±1.84	**4.79±0.16**	0.00±0.00
	Noise 1	**3.20±1.57**	1.49±1.54	1.70 ± 0.80	1.84 ± 0.63	0.48±0.68	2.08±2.06	
	Noise 2	1.38±0.95	0.69±0.95	2.07 ± 0.83	1.97 ± 0.62	3.97±1.08	**4.83±0.08**	
	Noise 3	2.71 ± 1.14	0.45 ± 0.71	1.81 ± 0.46	1.27 ± 0.33	1.96 ± 2.43	**4.28 ± 0.67**	
	Noise 4	−4.34 ± 1.04	−6.92 ± 2.31	−6.89 ± 0.55	−7.60 ± 0.38	2.35 ± 0.22	**4.84 ± 0.18**	
3s vs 4z	Noise 0	2.52±1.13	**3.01±0.80**	−0.26 ± 1.46	2.21 ± 1.55	0.38±1.27	2.96±0.03	2.96±0.04
	Noise 1	2.78±0.29	**2.89±0.44**	0.04 ± 1.19	1.70 ± 1.31	0.43±0.47	2.80±0.04	
	Noise 2	2.62±0.50	2.13±0.86	0.59 ± 1.07	0.34 ± 1.01	0.11±0.68	**2.89±0.09**	
	Noise 3	2.46 ± 0.35	**2.87 ± 0.20**	0.24 ± 0.29	0.57 ± 0.61	2.45 ± 0.51	2.52 ± 0.35	
	Noise 4	−4.36 ± 3.41	−0.81 ± 0.96	−12.06 ± 0.55	−11.00 ± 1.07	0.87 ± 0.17	**2.80 ± 0.03**	
3s5z	Noise 0	2.81±1.02	4.13±1.28	1.42 ± 1.32	1.37 ± 1.78	3.24±1.56	**5.71±0.13**	5.67±0.05
	Noise 1	2.77±0.27	3.38±0.46	1.91 ± 0.75	2.20 ± 0.73	3.69±0.75	**5.27±0.37**	
	Noise 2	2.89±0.42	3.91±0.85	2.57 ± 0.61	1.46 ± 0.64	3.24±1.04	**5.75±0.24**	
	Noise 3	3.09 ± 0.50	**3.80 ± 0.30**	1.79 ± 0.68	2.25 ± 0.86	1.85 ± 1.60	3.23 ± 0.38	
	Noise 4	−1.84 ± 0.65	−1.49 ± 0.64	−6.05 ± 0.74	−6.81 ± 1.02	0.74 ± 1.45	**5.82 ± 0.17**	
5m vs 6m	Noise 0	2.12±1.02	2.08±0.91	1.64 ± 1.02	0.31 ± 0.90	2.40±2.10	**4.83±0.23**	4.60±0.68
	Noise 1	2.86±0.40	**3.71±0.25**	0.79 ± 0.66	0.18 ± 0.34	2.59±2.02	2.20±2.19	
	Noise 2	3.04±0.50	**3.12±0.89**	0.97 ± 0.64	0.41 ± 0.27	2.49±0.50	2.89±0.64	
	Noise 3	2.36 ± 0.45	**4.10 ± 0.20**	0.82 ± 0.24	0.05 ± 0.28	0.89 ± 0.89	2.87 ± 2.13	
	Noise 4	2.68 ± 0.52	2.74 ± 0.36	−3.89 ± 0.22	−4.25 ± 0.22	3.34 ± 0.16	**4.83 ± 0.11**	
8m vs 9m	Noise 0	0.74±1.05	**2.91±0.78**	0.78 ± 1.31	0.06 ± 1.55	1.89±1.73	1.54±1.51	4.49±0.16
	Noise 1	2.43±0.24	1.64±0.48	0.58 ± 0.77	0.27 ± 0.88	2.72±0.68	**3.57±0.18**	
	Noise 2	2.12±0.42	2.00±0.64	0.48 ± 0.86	0.56 ± 0.43	−0.15±0.70	**2.78±1.18**	
	Noise 3	1.74 ± 0.35	3.29 ± 2.03	0.48 ± 0.46	0.17 ± 0.32	1.35 ± 1.38	**3.63 ± 0.44**	
	Noise 4	0.44 ± 2.71	1.40 ± 1.87	−6.69 ± 0.30	−6.40 ± 0.43	4.75 ± 0.39	**5.55 ± 0.50**	

The "baseline" is the result of MAPPO without noise. "3m" means 3 marines in MAS, and "z, s, sc" in tasks' names are "zealot, stalker, spine crawler", respectively.

contrast to SMAC. From the results, NDD is significantly superior to the others and exhibits a satisfactory anti-noise effect. For example, in Reference, the interference of noise produces a significant negative impact on experimental results, especially for the value decomposition algorithms. Accordingly, it can be observed from Fig. 3 that VDN, QMIX, MATD3, MADDPG, and MAPPO fluctuate heavily and converge to lower values. Regardless of the noise distribution, NDD stabilizes the performance and generates results similar to noise-free tasks in most scenarios. By the way, the effect of QMIX is much worse than other methods, due to that the mixing network used in QMIX requires more stable rewards until convergence. Furthermore, in noisy scenarios, a complex mixing network yields less robust results than the one with simple summation. Hence, we primarily choose GMM due to its simplicity and robustness. In addition, all algorithms exhibit performance decreases in complex noises. Due to the simultaneous involvement of multiple noise sources and non-Gaussian components, the results for Noise 3 and Noise 4 are worse than others, but the NDD yields more comparable performance to that obtained in noise-free cases.

Fig. 4 provides the sensitivity study about λ and α. Additionally, we choose the high-performance MAPPO for comparison. It can be observed that $\lambda = 1$ is the optimal value and either too large or too small may cause performance degradation. Similar observations can be applied to α. Additionally, NDD is relatively more sensitive to λ compared with α, and the performance when $\lambda = 0$ (i.e., no $\mathcal{L}_{\text{mean}}(\boldsymbol{\theta})$ in the loss function in (27)) is even worse than MAPPO. In other words, $\mathcal{L}_{\text{mean}}(\boldsymbol{\theta})$ is of vital importance to the NDD.

C. Distribution Decomposition Validation

We further evaluate the performance of NDD under simulated noise with arbitrary distribution. In order to further evaluate the accuracy of NDD, we decompose each noise distribution into

TABLE IV
MODELING ACCURACY OF NOISE DISTRIBUTION DECOMPOSITION UNDER DIFFERENT GMM ASSUMPTIONS

Noise Distribution	Decomposed Gaussian Components			Weights	Wasserstein Distance d_p
	Part 1	*Part 2*	*Part 3*		
$\mathcal{N}(0,5)$	$\mathcal{N}(5.4 \pm 0.48, 11 \pm 1.2)$	$\mathcal{N}(-0.47 \pm 0.20, 6.7 \pm 0.82)$	$\mathcal{N}(-4.4 \pm 0.14, 19 \pm 0.97)$	$[0.35 \pm 0.016, 0.30 \pm 0.029, 0.34 \pm 0.0051]$	$4.1 \pm 0.038 \times 10^{-3}$
$\mathcal{N}(0,3)$	$\mathcal{N}(2.8 \pm 0.14, 3.3 \pm 0.25)$	$\mathcal{N}(-0.48 \pm 0.11, 2.7 \pm 0.23)$	$\mathcal{N}(-2.3 \pm 0.28, 8.2 \pm 1.4)$	$[0.30 \pm 0.029, 0.35 \pm 0.025, 0.35 \pm 0.016]$	$3.8 \pm 0.12 \times 10^{-3}$
$0.5\mathcal{N}(1,5)+0.5\mathcal{N}(-1,5)$	$\mathcal{N}(4.4 \pm 2.4, 22 \pm 10)$	$\mathcal{N}(-0.10 \pm 1.4, 21 \pm 13)$	$\mathcal{N}(-2.5 \pm 1.1, 13 \pm 0.071)$	$[0.30 \pm 0.059, 0.34 \pm 0.017, 0.35 \pm 0.042]$	$3.9 \pm 0.072 \times 10^{-3}$
$0.4\mathcal{N}(5,1)+0.6\mathcal{N}(-5,3)$	$\mathcal{N}(4.9 \pm 0.0030, 0.87 \pm 0.046)$	$\mathcal{N}(-3.6 \pm 0.042, 32 \pm 2.0)$	$\mathcal{N}(-5.4 \pm 0.036, 5.2 \pm 0.67)$	$[0.34 \pm 0.017, 0.30 \pm 0.059, 0.35 \pm 0.042]$	$4.1 \pm 0.74 \times 10^{-3}$
$0.25\beta(1,2)+0.75\mathcal{N}(-5,3)$	$\mathcal{N}(0.29 \pm 0.10, 0.26 \pm 0.14)$	$\mathcal{N}(-4.2 \pm 0.057, 3.4 \pm 0.60)$	$\mathcal{N}(-7.0 \pm 0.27, 6.5 \pm 0.24)$	$[0.33 \pm 0.00093, 0.33 \pm 0.00047, 0.33 \pm 0.0014]$	$4.5 \pm 0.32 \times 10^{-3}$
$0.3\mathcal{N}(-1,5)+0.2\mathcal{N}(-2,5)+0.5\mathcal{N}(1.4,5)$	$\mathcal{N}(5.9 \pm 0.32, 13 \pm 1.1)$	$\mathcal{N}(-0.19 \pm 0.31, 10 \pm 0.13)$	$\mathcal{N}(-4.0 \pm 0.20, 15 \pm 0.020)$	$[0.32 \pm 0.026, 0.33 \pm 0.0038, 0.35 \pm 0.030]$	$3.5 \pm 0.033 \times 10^{-3}$
$0.35\mathcal{N}(-6,1)+0.3\beta(1,2)+0.35\chi^2(9)$	$\mathcal{N}(9.3 \pm 0.093, 0.60 \pm 0.082)$	$\mathcal{N}(0.30 \pm 0.061, 0.19 \pm 0.062)$	$\mathcal{N}(-5.9 \pm 0.0018, 0.95 \pm 0.039)$	$[0.35 \pm 0.037, 0.32 \pm 0.026, 0.33 \pm 0.011]$	$8.8 \pm 0.64 \times 10^{-3}$

$\mathcal{N}$ indicates a gaussian distribution, while β and χ^2 denote a beta and a chi-square distribution, respectively.

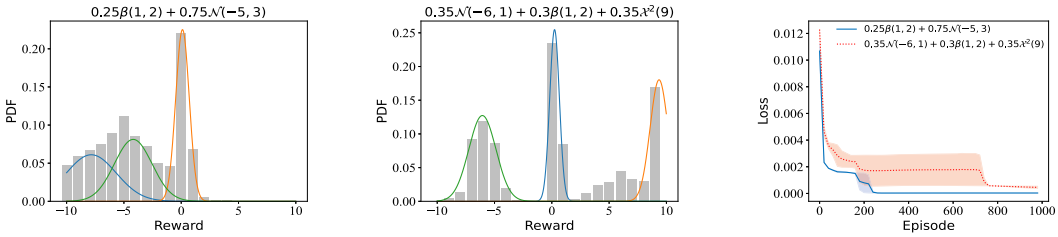


Fig. 5. Results of distribution decomposition over two noise cases (i.e., $0.25\beta(1,2)+0.75\mathcal{N}(-5,3)$, $0.35\mathcal{N}(-6,1)+0.3\beta(1,2)+0.35\chi(9)$). The left two sub-figures show the comparison between practical PDF histogram and weighted PDFs of decomposed distributions (3 colored curves); while the right sub-figure shows the curve of loss over training iterations. $\mathcal{N}, \beta, \mathcal{X}^2$ indicate a Gaussian, a Beta and a Chi-Square distribution, respectively.

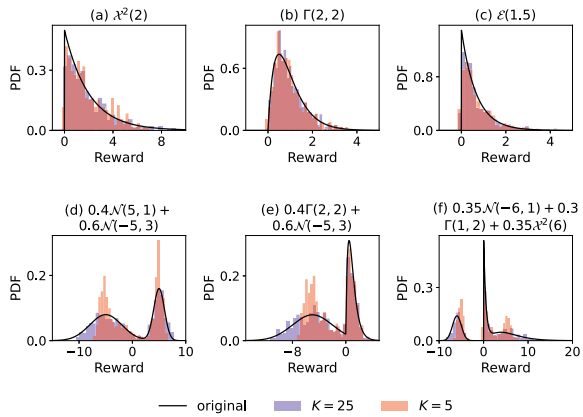


Fig. 6. The performance of DM in generated samples is illustrated. $\mathcal{N}, \mathcal{X}^2, \Gamma, \mathcal{E}$ are a Gaussian, a Chi-Square, a Gamma, and an exponential distribution, respectively.

3 parts and summarize the related intermediate results (i.e., decomposed distributions, corresponding weights, and computed Wasserstein distance d_p between the real and approximated distributions) in Table IV. Though it generally leads to inferior decomposition accuracy for more complex noise distribution, all the losses are rather low in terms of Wasserstein distance (even for noise including non-Gaussian components). To make the decomposition results more intuitive, Fig. 5 compares the PDFs of two noise distributions (i.e., $0.25\beta(1,2)+0.75\mathcal{N}(-5,3)$ and $0.35\mathcal{N}(-6,1)+0.3\beta(1,2)+0.35\chi(9)$) with weighted decomposed distributions, and provides the learning loss curves. It can be found from Fig. 5 that weighted distributions align closely with the noise distribution. Besides, after several hundred episodes of training, it quickly converges. In other words, the adoption of GMM to characterize the noise distribution is promising.

D. Results for Different Distortion Risk Functions

In order to manifest the effectiveness and stability of MARL with distortion risk functions $\rho_\eta(\tau)$, we test some formats of $\rho_\eta(\tau)$ (i.e., CPW [56], [57], WANG [58], POW [22], and CVaR [59]) with parameters configured as in Table VIII for NDD. Table V shows the performance of adopting different $\rho_\eta(\tau)$ in five MPE tasks, in terms of the means and standard deviations of the results. Note that means and standard deviations are compared independently. The "Expectation" in this Table means NDD, which uses *expectation* of action value distribution to update policies directly without any $\rho_\eta(\tau)$.

It can be observed from Table V that apart from Spread, wherein the Expectation method only achieves optimal performance, most of the tasks impose superior performance after imposing the risk preference on strategies. The optimal values are distributed relatively evenly among different types

TABLE V
AVERAGE SCORES IN MPE'S TASKS UNDER DIFFERENT DISTORTION RISK FUNCTIONS

Scenario	Type	CPW 0.71	WANG 0.75	WANG -0.75	POW -2.0	CVaR 0.25	CVaR 0.1	Expectation
Adversary	Noise 0	**−0.67** ± 0.77	−11.21±7.08	−2.51±0.63	−4.05±0.80	−1.26±**0.42**	−4.89±1.52	−1.72±0.45
	Noise 1	−1.47±0.63	−1.70±0.61	−1.89±1.06	−1.30±1.89	**−0.15**±**0.35**	−2.85±1.16	−1.93±0.43
	Noise 2	−0.86±0.67	−2.23±0.95	−1.31±0.61	−2.61±0.63	**−0.78**±0.62	−2.98±0.64	−1.35±**0.40**
Crypto	Noise 0	**−8.77**±6.29	−14.74±0.56	−13.85±1.85	−17.15±**0.50**	−9.32±1.20	−14.54±1.68	−12.09±1.69
	Noise 1	−8.86±2.06	−14.75±**0.35**	−9.88±3.79	−10.98±1.19	−10.25±0.91	−11.46±2.73	**−5.98**±0.81
	Noise 2	−11.19±1.83	**−8.96**±1.58	−13.70±1.90	−14.74±**0.49**	−10.93±0.64	−9.89±2.54	−10.74±0.52
Speaker Listener	Noise 0	**−29.70**±1.62	−32.55±2.80	−35.37±**1.50**	−33.02±1.96	−31.88±3.57	−34.65±2.25	−31.37±1.80
	Noise 1	−31.35±1.42	−34.00±3.25	−32.60±1.81	−34.84±2.35	−32.00±1.92	−36.23±1.97	**−30.12**±**1.34**
	Noise 2	−30.72±1.78	**−30.26**±1.31	−33.29±1.84	−31.68±1.44	−30.58±**0.96**	−33.67±1.04	−32.00±1.35
Spread	Noise 0	−94.25±1.37	−106.15±2.80	−97.14±2.15	−101.67±3.07	−94.66±1.54	−96.51±1.62	**−93.02**±**0.71**
	Noise 1	−94.55±1.29	−102.21±4.33	−99.08±1.27	−99.94±1.78	−95.46±**0.83**	−94.09±2.19	**−93.36**±1.88
	Noise 2	−93.78±1.98	−98.58±1.81	−99.70±2.08	−99.99±**0.75**	−92.61±1.64	−96.53±1.70	**−91.24**±1.58
Reference	Noise 0	−36.46±**0.49**	−37.35±1.40	**−35.31**±1.19	−35.94±1.15	−37.11±0.96	−39.96±1.00	−37.22±0.89
	Noise 1	**−35.48**±1.03	−36.70±1.11	−37.36±1.00	−36.71±**0.76**	−36.45±2.04	−37.64±1.23	−36.87±0.79
	Noise 2	**−36.64**±0.93	−37.05±1.16	−37.25±0.90	−37.02±0.84	−36.86±**0.79**	−39.10±1.17	−37.31±1.33

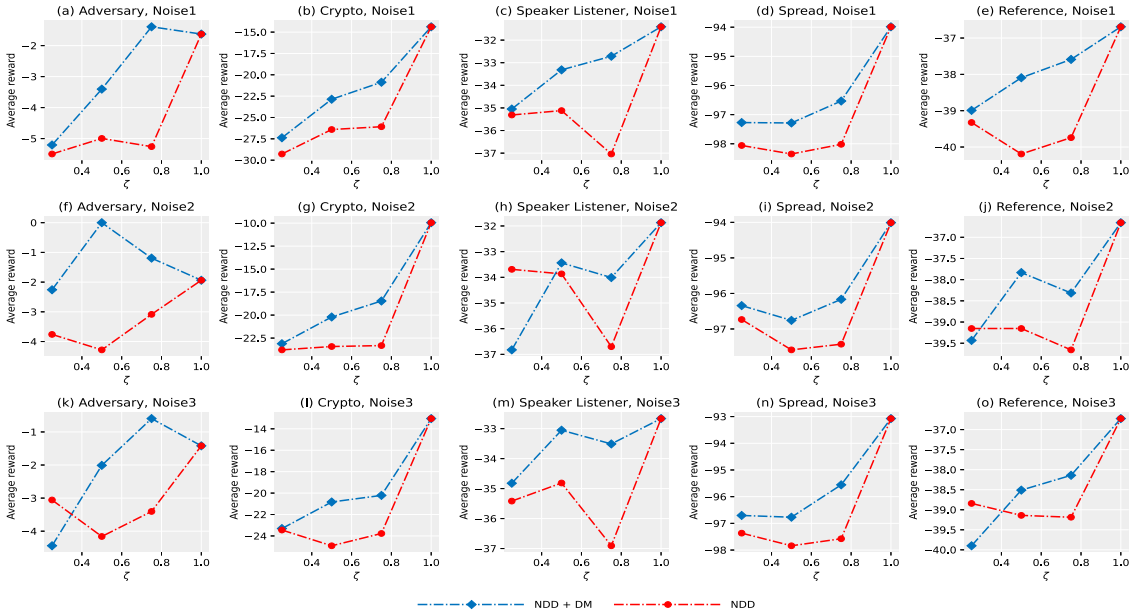


Fig. 7. The performance comparison of the NDD algorithm before and after using DM.

and parameters of $\rho_\eta(\tau)$, which reflects the significance of distributional MARL. As a function that better aligns with human decision-making habits, CPW(0.71) achieves the best results in one-third of the cases. The stability of strategy with CVaR(0.25) is the highest, at the cost of poorer performance on the effectiveness. Interestingly, some distortion risk functions are ineffective. For instance, CVaR(0.1) does not work in any of the 15 cases. In a word, CPW(0.71) can contribute to maximizing returns. Meanwhile, if stability and fault tolerance are the primary prerequisites, POW(-2) and CVaR(0.25) are worthy of the recommendation.

E. DM-Based Data Augmentation

In this section, our experiments mainly focus on assessing the data preciseness provided by DM for NDD. Beforehand, we first evaluate the distribution of the data generated by DM and compare it with the practical distribution in Fig. 6. In the case with $K = 25$, benefiting from the alignment capability of DM, the simulated data can resemble the original data, thus reducing the sampling and communication costs.

Next, Fig. 7 depicts the performance of the NDD approach with and without DM-based data augment. In this part, we intentionally hide some proportion of data for DM to supplement. For simplicity, the ratio of the remaining data quantity to the complete one is denoted as ζ. Generally speaking, along with the increase of ζ, the NDD approach gives superior performance, consistent with our intuition. More importantly, the use of DM significantly improves performance, strongly demonstrating the effectiveness of DM. That's to say, the same training effect can be achieved with significantly fewer training data than that without DM. On the other hand, for an over-small ζ, the incorporation of

DM leads to a decline in the performance of NDD, as too little data induces a significant gap between the distribution learned by DM and the original distribution. Specifically, according to analyses based on (46), the estimated expectations $\mathbb{E}(r)$ and variances $\mathbb{D}(r)$ are distorted in this data-lacking case, with non-negligible generation error. Therefore Fig. 7 also provides a good reference for balancing performance and interaction cost. Generally speaking, supplementing data within 50% from DM sounds reasonable.

VI. Conclusion

In this paper, we propose NDD to tackle the noisy rewards during cooperative and partially observed MARL tasks. In particular, we leverage the GMM-based reward distribution decomposition to characterize and approximate the distributional global noisy rewards by local Gaussian components, so as to mitigate the negative impact of noises. We also introduce distortion risk functions to leverage additional information learned from the action value distribution and discuss their applicability. Additionally, after providing the bounded generation and approximation error, we use diffusion models for data augmentation to address the high training cost issue. Moreover, we theoretically establish the consistency between the globally optimal action and locally optimal ones and carefully calibrate the loss function designed to update NDD networks. Extensive experimental results in MPE and SMAC also prove the effectiveness and superiority of NDD.

Our future work is dedicated to analyzing the time and space complexity of NDD, as well as exploring the mathematical relationship between hyperparameters in DM and two errors of generation and approximation.

References

[1] R. S. Sutton et al., *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998, vol. 135.

[2] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, San Francisco, California USA, 2017, pp. 2140–2146.

[3] B. Singh et al., "Reinforcement learning in robotic applications: A comprehensive survey," *Artif. Intell. Rev.*, vol. 55, pp. 945–990, Apr. 2021.

[4] Y. Du et al., "Guiding pretraining in reinforcement learning with large language models," 2023, *arXiv:2302.06692*.

[5] Y. LeCun et al., "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] P. Henderson et al., "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artif. Intell.*, New Orleans, Lousiana, USA, 2018, pp. 3207–3214.

[7] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[8] M. Hüttenrauch et al., "Guided deep reinforcement learning for swarm systems," 2017, *arXiv: 1709.06011*.

[9] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning," 2017, *arXiv: 1706.05296*.

[10] T. Rashid et al., "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Proc. Conf. Mach. Learn.*, vol. 21, no. 1, pp. 7234–7284, 2020.

[11] T. Rashid et al., "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 10199–10210.

[12] K. Son et al., "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Long Beach, California, USA, 2019, pp. 5887–5896.

[13] Y. Jia et al., "An UAV wireless communication noise suppression method based on OFDM modulation and demodulation," *Radio Sci.*, vol. 55, no. 2, pp. 1–14, 2020.

[14] J. Wang et al., "Reinforcement learning with perturbed rewards," in *Proc. AAAI Conf. Artif. Intell.*, New York, USA, 2020, pp. 6202–6209.

[15] O. Kilinc and G. Montana, "Multi-agent deep reinforcement learning with extremely noisy observations," 2018, *arXiv: 1812.00922*.

[16] M. G. Bellemare et al., "A distributional perspective on reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Sydney, Australia, 2017.

[17] J. Chen et al., "Distributional-utility actor-critic for network slice performance guarantee," in *Proc. 24th Int. Symp. Theory, Algorithmic Found., Protocol Des. Mobile Netw. Mobile Comput.*, 2023, pp. 161–170.

[18] A. Ben-Tal et al., *Robust Optimization*, vol. 28. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[19] Y. Kim and H. Lim, "Multi-agent reinforcement learning-based resource management for end-to-end network slicing," *IEEE Access*, vol. 9, pp. 56178–56190, 2021.

[20] R. Fantacci and B. Picano, "When network slicing meets prospect theory: A service provider revenue maximization framework," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3179–3189, Mar. 2020.

[21] W. Dabney et al., "Distributional reinforcement learning with quantile regression," in *Proc. AAAI Conf. Artif. Intell.*, New Orleans, Louisiana, USA, 2018, pp. 2892–2901.

[22] W. Dabney et al., "Implicit quantile networks for distributional reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Stockholmsmässan, Stockholm Sweden, 2018, pp. 1104–1113.

[23] I. Menache et al., "Basis function adaptation in temporal difference reinforcement learning," *Ann. Oper. Res.*, vol. 134, pp. 215–238, 2005.

[24] W.-F. Sun et al., "DFAC framework: Factorizing the value function via quantile mixture for multi-agent distributional Q-learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 9945–9954.

[25] Y. Birman et al., "Cost-effective ensemble models selection using deep reinforcement learning," *Inf. Fusion*, vol. 77, pp. 133–148, 2022.

[26] D. Reynolds, "Gaussian mixture models," *Encyclopedia Biometrics*, vol. 741, pp. 659–663, 2009.

[27] J. Ho et al., "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.

[28] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 8780–8794.

[29] C. Saharia et al., "Palette: Image-to-image diffusion models," in *Proc. ACM SIGGRAPH Conf.*, Vancouver BC Canada, 2022, pp. 15:1–15:10.

[30] A. Nichol et al., "GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models," 2021, *arXiv:2112.10741*.

[31] A. Ramesh et al., "Hierarchical text-conditional image generation with clip latents," 2022, *arXiv:2204.06125*.

[32] L. Buşoniu et al., "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications (Studies in Computational Intelligence)*. Berlin, Germany: Springer, 2010, pp. 183–221.

[33] Y. Wang and C. W. De Silva, "Multi-robot box-pushing: Single-agent Q-learning vs. team Q-learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, 2006, pp. 3694–3699.

[34] A. Galindo-Serrano and L. Giupponi, "Distributed Q-learning for aggregated interference control in cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 4, pp. 1823–1834, May 2010.

[35] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. 11th Int. Conf. Mach. Learn.*, San Francisco, CA, Morgan Kaufmann, 1994, pp. 157–163.

[36] H. Wang et al., "Cooperative distributed optimization in multiagent networks with delays," *IEEE Trans. Syst. Man Cybern.: Syst.*, vol. 45, no. 2, pp. 363–369, Feb. 2015.

[37] F. Xiao and L. Wang, "Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays," *IEEE Trans. Autom. Control*, vol. 53, no. 8, pp. 1804–1816, Sep. 2008.

[38] J. Foerster et al., "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona Spain, 2016, pp. 2137–2145.

[39] S. Sukhbaatar et al., "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona Spain, 2016, pp. 2244–2252.

[40] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

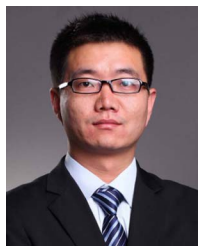
[41] A. O. Castaneda, "Deep reinforcement learning variants of multi-agent learning algorithms," Master of Science, School of Informatics, University of Edinburgh, 2016.

[42] E. A. O. Diallo et al., "Learning to coordinate with deep reinforcement learning in doubles pong game," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl.*, Cancun, Mexico, 2017, pp. 14–19.

[43] R. Koenker, *Quantile Regression*. Cambridge, U.K.: Cambridge Univ. Press, 2005, vol. 38.
[44] J. Hu et al., "Distributional reward estimation for effective multi-agent deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 12619–12632.
[45] B. Hu et al., "Deep generative models with data augmentation to learn robust representations of movement intention for powered leg prostheses," *IEEE Trans. Med. Robot. Bionics*, vol. 1, no. 4, pp. 267–278, Nov. 2019.
[46] S. Wang, Y. Yang, Z. Wu, Y. Qian, and K. Yu, "Data augmentation using deep generative models for embedding based speaker recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 2598–2609, 2020.
[47] Z. Wang et al., "Diffusion policies as an expressive policy class for offline reinforcement learning," 2023, *arXiv:2208.06193*.
[48] H. He et al., "Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning," 2023, *arXiv:2305.18459*.
[49] J. Geng et al., "Diffusion policies as multi-agent reinforcement learning strategies," in *Proc. Int. Conf. Artif. Neural Netw.*, Heraklion, Greece, 2023, pp. 356–364.
[50] Z. Li et al., "Beyond conservatism: Diffusion policies in offline multi-agent reinforcement learning," 2023, *arXiv:2307.01472*.
[51] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Berlin, Germany: Springer, 2016.
[52] N. Wiener, "Tauberian theorems," *Ann. Math.*, vol. 33, pp. 1–100, 1932.
[53] E. Narayanan, "Wiener tauberian theorems for L1 (K\ G/ K)," *Pacific J. Math.*, vol. 241, no. 1, pp. 117–126, 2009.
[54] N. Ito et al., "Noisy cGMM: Complex Gaussian mixture model with non-sparse noise model for joint source separation and denoising," in *Proc. 26th Eur. Signal Process. Conf.*, 2018, pp. 1662–1666.
[55] J. Chen et al., "RGMComm: Return gap minimization via discrete communications in multi-agent reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, Vancouver, Canada, 2024, pp. 17327–17336.
[56] A. Tversky and D. Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty," *J. Risk Uncertainty*, vol. 5, no. 4, pp. 297–323, 1992.
[57] G. Wu and R. Gonzalez, "Curvature of the probability weighting function," *Manage. Sci.*, vol. 42, no. 12, pp. 1676–1690, 1996.
[58] S. S. Wang, "A class of distortion operators for pricing financial and insurance risks," *J. Risk Insurance*, vol. 67, no. 1, pp. 15–36, Mar. 2000.
[59] Y. Chow and M. Ghavamzadeh, "Algorithms for CVaR optimization in MDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach Convention Center, Long Beach, 2014, pp. 3509–3517.
[60] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *Proc. AAAI Conf. Artif. Intell.*, New Orleans, Louisiana, USA, 2018, pp. 1495–1502.
[61] R. Lowe et al., "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, California, USA, 2017, pp. 6379–6390.
[62] M. Samvelyan et al., "The starcraft multi-agent challenge," 2019, *arXiv: 1902.04043*.
[63] C. S. de Witt et al., "Is independent learning all you need in the starcraft multi-agent challenge?," 2020, *arXiv: 2011.09533*.
[64] J. Schulman et al., "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.
[65] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 24611–24624.
[66] S. Fujimoto et al., "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, Stockholmsmässan, Stockholm Sweden, 2018, pp. 1582–1591.
[67] J. Sohl-Dickstein et al., "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Int. Conf. Mach. Learn.*, Lille, France, 2015, pp. 2256–2265.
[68] E. Dubinsky, "Teaching mathematical induction II," *J. Math. Behav.*, vol. 8, no. 3, pp. 285–304, 1989.

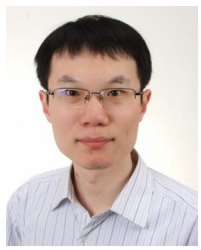
Wei Geng received the MS degree in software engineering from the Hangzhou Institute for Advanced Study, UCAS, in 2024. Now, he is an intern researcher with Zhejiang Lab. His research interests include multi-agent reinforcement learning.

Baidi Xiao (Graduate Student Member, IEEE) received the MS degree from the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. His research interests currently focus on deep learning, multi-agent reinforcement learning, and the Internet of Vehicles.



Rongpeng Li (Senior Member, IEEE) is currently an associate professor with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. He was a research engineer with the Wireless Communication Laboratory, Huawei Technologies Company, Ltd., Shanghai, China, from August 2015 to September 2016. He was a visiting scholar with the Department of Computer Science and Technology, University of Cambridge, Cambridge, U.K., from February 2020 to August 2020. His research interest currently focuses on networked intelligence for communications evolving (NICE). He received the Wu Wenjun Artificial Intelligence Excellent Youth Award, in 2021. He serves as an editor for *China Communications*.

Ning Wei is currently an engineering expert with Zhejiang Lab, Hangzhou, China. He was an AI&Robotic Engineer with KUKA Robotics, Shanghai, China, from May 2014 to November 2017. He was an AI researcher with the Hikvision Research Institute, Hangzhou, China, from December 2017 to July 2022. His research interest currently focuses on Reinforcement Learning and Simulation&Data Mixed-Driven Intelligence. He is also a member of the China Computer Federation.



Dong Wang received the PhD degree in computing from the University of the West of Scotland, Paisley, U.K., in 2020. He is a senior research engineer with 6G Research Center, China Telecom Research Institute, Beijing, China, and leads the network architecture and services team. He also serves as a TSC co-chair of ONAP, Linux Foundation Networking. His research interests include 6G mobile networks, network architecture and operation, intent-driven networks, and complex heterogeneous networks.

Zhifeng Zhao (Member, IEEE) received the BE degree in computer science, the ME degree in communication and information systems, and the PhD degree in communication and information systems from the PLA University of Science and Technology, Nanjing, China, in 1996, 1999, and 2002, respectively. From 2002 to 2004, he acted as a post-deoctoral researcher with Zhejiang University, Hangzhou, China, where his researches were focused on multimedia next-generation networks (NGNs) and soft switch technology for energy efficiency. From 2005 to 2006, he acted as a senior researcher with the PLA University of Science and Technology, where he performed research and development on advanced energy-efficient wireless routers, *ad-hoc* network simulators, and cognitive mesh networking test-bed. From 2006 to 2019, he was an associate professor with the College of Information Science and Electronic Engineering, Zhejiang University. Currently, he is with the Zhejiang Lab, Hangzhou as the chief engineering officer. His research areas include software-defined networks (SDNs), wireless networks in 6G, computing networks, and collective intelligence. He is the symposium co-chair of ChinaCom 2009 and 2010. He is the Technical Program Committee (TPC) co-chair of the tenth IEEE International Symposium on Communication and Information Technology (ISCIT 2010).