MCMARL: Parameterizing Value Function via Mixture of Categorical Distributions for Multi-Agent Reinforcement Learning

Jian Zhao, Mingyu Yang, Youpeng Zhao, Xunhan Hu, Wengang Zhou, Senior Member, IEEE, and Houqiang Li, Fellow, IEEE

Abstract—In cooperative multi-agent tasks, a team of agents jointly interact with an environment by taking actions, receiving a team reward, and observing the next state. During the interactions, the uncertainty of environment and reward will inevitably induce stochasticity in the long-term returns, and the randomness can be exacerbated with the increasing number of agents. However, such randomness is ignored by most of the existing value-based multi-agent reinforcement learning (MARL) methods, which only model the expectation of Q-value for both the individual agents and the team. Compared to using the expectations of the long-term returns, it is preferable to directly model the stochasticity by estimating the returns through distributions. With this motivation, this article proposes a novel value-based MARL framework from a distributional perspective, i.e., parameterizing value function via Mixture of Categorical distributions for MARL (MCMARL). Specifically, we model both the individual and global Q-values with categorical distribution. To integrate categorical distributions, we define five basic operations on the distribution, which allow the generalization of expected value function factorization methods (e.g., value decomposition networks (VDN) and QMIX) to their MCMARL variants. We further prove that our MCMARL framework satisfies the Distributional-Individual-Global-Max principle with respect to the expectation of distribution, which guarantees the consistency between joint and individual greedy action selections in the global and individual Q-values. Empirically, we evaluate MCMARL on both the stochastic matrix game and the challenging set of StarCraft II micromanagement tasks, showing the efficacy of our framework.

Index Terms—Distribution mixing, distribution modeling, multi-agent reinforcement learning (MARL).

I. INTRODUCTION

REINFORCEMENT learning (RL) aims to learn a mapping from the observation state to the action of an agent so as to maximize a long-term return received from an environment. Recently, RL has been successfully investigated from single-agent problems to multi-agent tasks in a variety of fields, which requires decision making. Multiplayer games [1], [2], [3], [4], [5], sensor networks [6], [7], Internet safety [8], robotic control [9], [10], and traffic light control [11] are some examples of multi-agent systems that cooperate to achieve common goals [12]. In this article, we focus on cooperative multi-agent reinforcement learning (MARL) with partial observability and communication constraints. In such a setting, the agents are required to take actions to interact with the environment in a decentralized manner. Generally, in MARL, the observed long-term return is characterized by stochasticity due to partial observations, changing policies of all the agents and environment model dynamics. Moreover, the stochasticity caused by actions will be intensified with the increasing number of agents. Owing to the randomness in the long-term returns, it is preferable to model the value functions via distributions rather than the expectations. Bellemare et al. [13] have proven that the distributional Bellman operator for policy evaluation is a contraction in the p-Wasserstein distance, and their proposed algorithm has managed to outperform traditional value-based RL algorithms, indicating the potential of solving RL problems from the perspective of distributions.

To model the value functions via distributions, the current mainstream solution is distributional RL, which predicts the distribution over returns instead of a scalar mean value by leveraging either a categorical distribution [13] or a quantile function [14]. Most of the studies in distributional RL focus on single-agent domains [13], [14], [15], [16], which cannot be directly applied to the value-based MARL. The reasons arise from two aspects: 1) in value-based MARL, the individual...
distributional $Q$-values should be integrated into global distributional $Q$-value and 2) the integration should guarantee the consistency between joint and individual greedy action selections in the global and individual $Q$-values, called Distributional-Individual-Global-Max (DIGM) principle.

To the best of our knowledge, there exist few works focusing on distributional MARL [17], [18]. Only one recent work [19], named DFAC, models both the individual and global $Q$-values from a distributional perspective, which decomposes the return distribution into the deterministic part (i.e., expected value) and stochastic part with mean zero. To satisfy the DIGM principle, DFAC relies on a strong assumption that the expectation of global value can be fitted by the expectation of individual value, which does not necessarily hold in practice. Taking a two-agent system as a toy example, the relationship between individual and global values is $Z_{\text{tot}} = \text{Relu}(Z_1 + Z_2)

\text{Case1 : } Z_1 = \begin{cases} 1, & \text{prob} = 0.5 \\ -1, & \text{prob} = 0.5 \end{cases}, \quad Z_2 = 1
\text{E}[Z_{\text{tot1}}] = 1
\text{Case2 : } Z_1 = \begin{cases} 2, & \text{prob} = 0.5 \\ -2, & \text{prob} = 0.5 \end{cases}, \quad Z_2 = 1
\text{E}[Z_{\text{tot2}}] = 1.5.

In the above two cases, the individual values follow different distributions with the same expectation, while the expectations of the global value are different. Thereby, it is difficult to fit the global value expectation only by the individual value expectations.

To this end, we propose a novel distributional MARL framework, i.e., parameterizing value function via Mixture of Categorical distributions for MARL (MCMARL). Our method models both the individual and global $Q$-value distributions by categorical distributions. In this way, the distributions of individual $Q$-values capture the uncertainty of the environment from each agent’s perspective, while the distribution of global $Q$-value directly approximates the randomness of the total return. To integrate the individual distributions into the global distribution, we define five basic operations, namely, Weighting, Bias, Convolution, Projection, and Function, which can realize the transformation of the distribution and the combination of multiple distributions. These basic operations allow the generalization of expected value function factorization methods (e.g., value decomposition networks (VDN) and QMIX) to their MCMARL variants without violating DIGM.

To evaluate the capability of MCMARL in distribution factorization, we first conduct a simple stochastic matrix game, where the true return distributions are known. The results reveal that the distributions estimated by our method are very close to the true return distributions. Beyond that, we perform experiments on a range of unit micromanagement benchmark tasks in StarCraft II [20]. The results on StarCraft II micromanagement benchmark tasks show that 1) our MCMARL framework is a more beneficial distributional MARL method than DFAC and 2) distributional QMIX (DQMIX) (MCMARL variant of QMIX) always achieves the leading performance compared to the baselines. Furthermore, we analyze the impact of the hyperparameter—the size of the support set of categorical distribution—and figure out that a size of 51 is sufficient to obtain considerable performance.

II. Background

In this section, we introduce some background knowledge for convenience of understanding our method. First, we discuss the problem formulation of a fully cooperative MARL task. Next, we introduce the concept of deep multi-agent Q-learning. Then, we present the centralized training with decentralized execution (CTDE) paradigm and recent representative value function factorization methods in this field. Finally, we describe the concept of distributional RL and summarize the related studies.

A. Decentralized Partially Observable Markov Decision Process

We model a fully cooperative multi-agent task as a decentralized partially observable Markov decision process (Dec-POMDP) [21], following the most recent works in the cooperative MARL domain. The Dec-POMDP can be described as a tuple $M = (S, Z, A, r, P, O, N, \gamma)$, where $S$ is a finite set of global states, $Z$ is the set of individual observations, and $A$ is the set of individual actions. At each time step, each agent $i \in N := \{g_1, \ldots, g_N\}$ selects an action $a_i \in A$, forming a joint action $a := [a_1, \ldots, a_N] \in A^N$. This leads to a transition on the environment according to the state transition function $P(s' | s, a) : S \times A^N \times S \to [0, 1]$, and the environment returns a joint reward (i.e., team reward) $r(s, a) : S \times A^N \to \mathbb{R}$ shared among all the agents. $\gamma \in [0, 1]$ is the discount factor. Each agent $i$ can only receive the individual and partial observations $o_i \in Z$, according to the observation function $O(s, i) := (Z \times A)^i$, on which it constructs its individual policy $\pi_i(a_i | \tau_i) : T \times A \to [0, 1]$. The objective of a fully cooperative multi-agent task is to learn a joint policy $\pi := [\tau_i]_{i=1}^N$ so as to maximize the expected cumulative team reward. To be mentioned, the operator $:=$ is used as a definition or assignment operator, typically used in computer programming and mathematical notation to define a symbol or variable.

B. Deep Multi-Agent Q-Learning

An early multi-agent $Q$-learning algorithm may be independent $Q$-learning (IQL) [22], which learns decentralized policy for each agent independently. IQL is simple to implement but suffers from nonstationarity of the environment and may lead to nonconvergence of the policy. To this end, many multi-agent $Q$-learning algorithms [23], [24], [25], [26], [27] are dedicated to learning a global $Q$-value function

$$Q_{\text{tot}}(s, \tau, a) = \mathbb{E}_{s_{1: \infty}, a_{1: \infty}} \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a \right]$$

where $\tau$ is the joint action-observation history and $r_t$ is the team reward at time $t$. Similar to deep q-network (DQN) [28], deep multi-agent $Q$-learning algorithms represent the global $Q$-value function with a deep neural network parameterized by
\( \theta \) and then use a replay memory to store the transition tuple \((\tau, a, r, \tau')\). To ensure the stability of the learning process, there usually exists a target network that can be parameterized by \(\theta^-\). Parameters \(\theta\) are learnt by sampling a batch of transitions \(\{(\tau^{(i)}, a^{(i)}, r^{(i)}, \tau'^{(i)})\}_{i=1}^B\) to minimize the following temporal difference (TD) error:

\[
\mathcal{L}(\theta) = \sum_{i=1}^B \left[ (y^{(i)} - Q_{\text{tot}}(\tau^{(i)}, a^{(i)}; \theta)) \right]^2
\]

(2)

where \(y^{(i)} = r^{(i)} + \gamma \max_{a'} Q_{\text{tot}}(\tau'^{(i)}, a'; \theta^-)\). It is a common practice to periodically update the parameters of the target network to match those of the primary network, which helps to reduce the potential for overestimation or oscillation in the learning process, leading to more stable and effective training. This can be done by copying the parameters from the primary network to the target network every \(C\) steps, where \(C\) is a hyperparameter depending on the specific algorithm and the problem domain. The joint policy can be derived as \(\pi(\tau) = \arg \max_a Q_{\text{tot}}(\tau, a; \theta)\).

C. CTDE and Value Function Factorization

In cooperative MARL, fully decentralized methods [22], [29] are scalable but suffer from nonstationarity issue. On the contrary, fully centralized methods [30], [31] mitigate the nonstationarity issue but encounter the challenge of scalability, as the joint state-action space grows exponentially with the number of agents. To combine the best of both the worlds, a popular paradigm called CTDE has drawn substantial attention recently. In CTDE, agents take actions based on their own local observations and are trained to coordinate their actions in a centralized way. During execution, the policy of each agent only relies on its local action-observation history, which guarantees the decentralization.

Recent value-based MARL methods realize CTDE mainly by factorizing the global Q-value function into individual Q-value functions [23], [24], [25]. To ensure that the collection of individual optimal actions of each agent during execution is equivalent to the optimal actions selected from global Q-value, value function factorization methods have to satisfy the following individual-global-max (IGM) [25] condition:

\[
\arg \max_a Q_{\text{tot}}(\tau, a) = \left( \begin{array}{c}
\arg \max_{a_i} Q_1(\tau_1, a_1) \\
\vdots \\
\arg \max_{a_N} Q_N(\tau_N, a_N)
\end{array} \right).
\]

(3)

As the first attempt of this stream, VDN [23] represents the global Q-value function as a sum of individual Q-value functions. Considering that VDN ignores the global information during training, QMIX [24] assigns the nonnegative weights to individual Q-values with a nonlinear function of the global state. These two factorization methods are sufficient to satisfy (3) but inevitably limit the global Q-value function class they can represent due to their structural constraint. Let us consider a multi-agent task where certain agents must make sacrifices in their own rewards to maximize the overall team reward. In such a scenario, the derivative of global Q-value with respect to these individual Q-values may turn out to be negative, which signifies the limitations of these algorithms in capturing such dynamics. To address the representation limitation, QTRAN proposes to learn a state-value function and transform the original global Q-value function \(Q_{\text{tot}}\) into an easily decomposable one \(Q'_{\text{tot}}\) that shares the same optimal actions with \(Q_{\text{tot}}\) [25]. However, the computationally intractable constraint imposed by QTRAN may lead to poor performance in complex multi-agent tasks.

D. Distributional RL

Distributional RL aims to approximate the distribution of returns (i.e., the discounted cumulative rewards) denoted by a random variable \(Z(s, a)\), whose expectation is the scalar value function \(Q(s, a)\). Similar to the Bellman equation of Q-value function, the distributional Bellman equation can be defined by

\[
Z(s, a) \overset{D}{=} R(s, a) + \gamma Z(s', a')
\]

(4)

where \(s' \sim P(\cdot | s, a)\) and \(a' \sim \pi(\cdot | s')\). Different from traditional RL, distributional RL considers the concept of random variables for various elements, such as the reward function denoted as \(R(s, a)\). These elements are associated with specific probability distributions, thereby exhibiting randomness in their characteristics. Here, the operator \(\overset{D}{=}\) is used to define a term or symbol in terms of its distributional properties. In fact, it is often adopted in probability theory and statistics to define random variables or parameters based on their probability distributions. For instance, \(X \overset{D}{=} N(0, 1)\) means that the random variable \(X\) is defined in a standard normal distribution with mean 0 and variance 1. As revealed in (4), \(Z(s, a)\) involves three sources of randomness: the reward \(R(s, a)\), the transition \(P(\cdot | s, a)\), and the next-state value distribution \(Z(s', a')\) [13]. Then, we have the distributional Bellman optimality operator \(T^*\) as follows:

\[
T^* Z(s, a) \overset{D}{=} \sum_a R(s, a) + \gamma Z(s', a', \arg \max_{a'} \mathbb{E}[Z(s', a')]).
\]

(5)

Based on the distributional Bellman optimality operator, the objective of distributional RL is to reduce the distance between the distribution \(Z(s, a)\) and the target distribution \(T^* Z(s, a)\). Therefore, a distributional RL algorithm must address two issues: 1) how to parameterize the return distribution and 2) how to choose an appropriate metric to measure the distance between two distributions. To model the return distribution, many RL methods in the single agent reinforcement learning (SARL) domain are proposed with promising results [13], [14], [15], [16]. In this article, we employ the categorical distribution [13], which represents the distribution with probability masses placed on a discrete set of possible returns, and then minimize the Kullback–Leibler (KL) divergence between the Bellman target and the current estimated return distribution.

III. Method

In this section, we first define five basic operations on the distribution of random variables that satisfy the DIMG principle. Based on the operations, we give an introduction to
our MCMARL framework and illustrate the variants of VDN and QMIX under the MCMARL framework. Furthermore, we briefly present the training and execution strategy.

A. Basic Operations on Distribution

Let \( X \) be a discrete random variable, following a categorical distribution, denoted by \( X \sim (M, V_{\text{min}}, V_{\text{max}}, P) \), where \( M \in \mathbb{N} \) and \( V_{\text{min}}, V_{\text{max}} \in \mathbb{R} \). The support of \( X \) is a set of atoms \( \{x_j = V_{\text{min}} + j\Delta x : 0 \leq j < M\} \), where \( \Delta x := \frac{V_{\text{max}} - V_{\text{min}}}{M-1} \) and \( P \) is the atom probability, i.e.,

\[
X = x_j \quad \text{w.p.} \quad p_j. \tag{6}
\]

To apply transformation and combination to random variables with categorical distribution, we define five basic operations that are also demonstrated in the corresponding figures (Fig. 1–5).

Operation 1 (Weighting): Analogous to the scaling operation to a scalar variable, the weighting operation \( W \) to scale up a discrete random variable by \( w \in \mathbb{R} \) is defined as follows:

\[
W_wX := wx_j \quad \text{w.p.} \quad p_j. \tag{7}
\]

The weighting operation over a distribution \( X \) can be abbreviated as \( w \cdot X \).

Operation 2 (Bias): Analogous to the panning operation to a scalar variable, the bias operation \( B_b \) to pan a discrete random variable by \( b \in \mathbb{R} \) is defined as follows:

\[
B_bX := x_j + b \quad \text{w.p.} \quad p_j. \tag{8}
\]

which is abbreviated as \( X + b \).

Operation 3 (Convolution): To combine the two random variables \( X_1 \sim (M_1, V_{1,\text{min}}, V_{1,\text{max}}, P_1) \) and \( X_2 \sim (M_2, V_{2,\text{min}}, V_{2,\text{max}}, P_2) \) with the same atom interval \( \Delta x \), we define the convolution operation \( \text{Conv}(\cdot, \cdot) \) as follows:

\[
\text{Conv}(X_1, X_2) := x_j^* \quad \text{w.p.} \quad p_j^* \quad \text{where} \quad x_j^* = V_{1,\text{min}} + V_{2,\text{min}} + \Delta x \cdot j \tag{9}
\]

where \( 0 \leq j < M_1 + M_2 - 1 \). Let \( \hat{M} := M_1 + M_2 - 1 \). If \( M_1 \geq M_2 \), then

\[
p_j^* = \begin{cases} 
\sum_{k=0}^{j} p_{1,k}p_{2,j-k}, & 0 \leq j < M_2 \\
\sum_{k=j-M_2+1}^{M_1-1} p_{1,k}p_{2,j-k}, & M_2 \leq j < M_1 \\
\sum_{k=j-M_2+1}^{M_1-1} p_{1,k}p_{2,j-k}, & M_1 \leq j < \hat{M} 
\end{cases} \tag{10}
\]

If \( M_1 < M_2 \), then

\[
p_j^* = \begin{cases} 
\sum_{k=0}^{j} p_{1,j-k}p_{2,j}, & 0 \leq j < M_1 \\
\sum_{k=j-M_2+1}^{M_1-1} p_{1,j-k}p_{2,j}, & M_1 \leq j < M_2 \\
\sum_{k=j-M_2+1}^{M_1-1} p_{1,j-k}p_{2,j}, & M_2 \leq j < \hat{M}
\end{cases} \tag{11}
\]

Conv\((X_1, X_2)\) is abbreviated as \( X_1 \ast X_2 \).

Operation 4 (Projection): To map the random variable distribution of \( [x_j] \) to atoms \( \hat{x}_k \) where \( \hat{x}_k = V_{\text{min}} + k\Delta \hat{x} : 0 \leq k < K \) and \( \Delta \hat{x} := \frac{V_{\text{max}} - V_{\text{min}}}{K-1} \), we define the projection operation \( \Phi(\hat{x}_k) \) as follows:

\[
\Phi(\hat{x}_k)X := \hat{x}_k \quad \text{w.p.} \quad \sum_j \left[ 1 - \frac{|[x_j] - \hat{x}_k|}{\Delta \hat{x}} \right] p_j \tag{12}
\]

where \([j]\) bounds its argument in the range \([a, b]\).

Operation 5 (Function): To apply nonlinear operation over a random variable, we define the function operation \( F_j, f : \mathbb{R} \to \mathbb{R} \) as follows: which is abbreviated as \( f(X) \).

Furthermore, we give theoretical proof that the five basic operations approximately satisfy the DIGM principle. Correspondingly, the network structure composed of these five basic operations also satisfies the DIGM principle.
B. DIGM Proof

To ensure the consistency between joint and individual greedy action selections, the distribution mixing network must satisfy the DIGM [19] condition, which is formulated as follows:

\[
\arg \max_a \mathbb{E} [Z_{tot}(\tau, a)] = \left( \begin{array}{c}
\arg \max_{a_1} \mathbb{E} [Z_1(\tau_1, a_1)] \\
\vdots \\
\arg \max_{a_N} \mathbb{E} [Z_N(\tau_N, a_N)] 
\end{array} \right).
\]

One sufficient condition for the distribution mixing network that meets the DIGM condition is that all the operations meet the DIGM condition. This is easy to understand as if all the five basic operations adhere to the DIGM principle, the distribution mixing network also meets the DIGM principle due to the transitive nature of this property considering that the mixing network is composed of these five operations. This observation can be likened to the scenario where \( f(x) \) and \( g(x) \) represent two increasing functions, whereby it can be deduced that \( g(f(x)) \) is likewise an increasing function. We provide some propositions to demonstrate that, under certain conditions, the five basic distribution operations satisfy the DIGM condition, and detailed proof is presented in the Appendix in the supplementary material.

C. Framework of MCMARL

For each agent, we parameterize its individual \( Q \)-value distribution with categorical distribution, which has high flexibility to approximate any shape of the distribution. Assume that the support set of the agent’s distribution, denoted as \([z] = \{z_1, z_2, \ldots, z_M | z_1 \leq \cdots \leq z_M, M \in \mathbb{N}\}\), is uniformly distributed over the predefined range \([V_{\min}, V_{\max}]\), where \(V_{\min}, V_{\max} \in \mathbb{R}\) are the minimum and maximum returns, respectively. Note that all the individual \( Q \)-value distributions share the same support set.

Based on these assumptions, learning individual distributions is equivalent to learning the atom probabilities. For each agent, there is one agent network \( \theta : \mathcal{Z} \times \mathcal{A} \rightarrow \mathbb{R}^M \), which estimates the probabilities of atoms in the support set. Taking agent \( i \) as an example, at each time step, its agent network receives the current individual observation \( o_i^t \) and the last action \( a_i^{t-1} \) as input and generates the atom probabilities as follows:

\[
Z_i(\tau_i, a_i) = z_j \quad \text{w.p.} \quad p_i,j(\tau_i, a_i) := \frac{e^{\theta_j(\tau_i, a_i)}}{\sum_k e^{\theta_k(\tau_i, a_i)}} \quad (13)
\]

where \( j \in \{1, \ldots, M\} \) and \( \tau_i \) denotes the agent \( i \)'s action-observation history. For scalability, parameters \( \theta \) are shared among all the agents.

Given the five basic operations, the variants of the existing value-based models under our MCMARL framework can be designed based on the basic operations over the individual distributions, as shown in Fig. 7(b). Here, we introduce distributional value decomposition networks (DVDN) and DQMIX, which are the corresponding MCMARL variants of VDN and QMIX, respectively.

VDN sums up the individual values as the global value, and its MCMARL variant, DVDN, is to apply the Convolution operation over the individual value distributions followed by the Projection operation:

1. \( A_{tot} := Z_1 \times Z_2 \times \cdots \times Z_N \) (Convolution)
2. \( Z_{tot} := \Phi[z](A_{tot}) \) (Projection).

Corresponding to the design of VDN, DVDN simply takes a sum of the individual randomness as the global randomness, which relies on the fact that the individual \( Q \)-values are independent. However, the independence of individual \( Q \)-value distributions does not necessarily hold, which might limit the performance of DVDN (see A.2 for details).

DQMIX, the MCMARL variant of QMIX, mixes the individual \( Q \)-values distributions into global \( Q \)-value distribution \( Z_{tot} \) by leveraging a multilayer neural network. The sequence of operations of \( k \)th layer is formulated as follows:

1. \( A_{i,j}^k := w_{i,j}^k \cdot Z_i^k \) (Weighting)
2. \( B_{i,j}^k := \Phi[z](A_{i,j}^k) \) (Projection)
3. \( C_{i}^k := B_{1,i}^k \ast B_{2,i}^k \ast \cdots \ast B_{N_k,i}^k \) (Convolution)
4. \( D_i^k := C_i^k + b_i^k \) (Bias)
5. \( E_i^k := f(D_i^k) \) (Function)
6. \( Z_{tot}^{k+1} := \Phi[z](E_i^k) \) (Projection)

where \( i \in \{1, \ldots, N_k\}, j \in \{1, \ldots, N_{k+1}\} \) and \( N_k \) is the number of input distributions of \( k \)th layer. The parameters \( w_{i,j}^k \in \mathbb{R} \) and \( b_i^k \in \mathbb{R} \) are generated by the \( k \)th hypernetwork conditioned on the global state. Note that each hypernetwork that generates \( w_{i,j}^k \) is followed by an absolute activation function, which guarantees that the parameters of the Weighting operations are nonnegative. We also present an example of the MCMARL variant of value-based factorization model, namely, DQMIX, in Fig. 6 to intuitively demonstrate our design.

D. Training and Inference

In the training phase, each agent \( i \) interacts with the environment using the \( \epsilon \)-greedy policy over the expectation of individual \( Q \)-value distribution, i.e., \( \mathbb{E}[Z_i] = \sum_j p_{i,j} z_j \). The transition tuple \((\tau, a, r, \tau')\) is stored into a replay memory. Then, the learner randomly fetches a batch of samples from the replay memory. The network is optimized by minimizing the sample loss, i.e., the cross-entropy term of KL divergence

\[
D_{KL}(\Phi[T^*Z_{tot}(\tau, a; \theta^-) || Z_{tot}(\tau, a; \theta)]) \quad (14)
\]

where \( T^*Z_{tot}(\tau, a; \theta^-) \) is the Bellman target according to (5), \( \theta^- \) are the parameters of a target network that are periodically copied from \( \theta \), and \( \Phi \) is the projection of Bellman target onto the support of \( Z_{tot}(\tau, a; \theta) \).

Since our method satisfies the DIGM condition, the policy learned during centralized training can be directly applied to execution. During the execution phase, each agent chooses a greedy action \( a_i \) at each time step with respect to \( \mathbb{E}[Z_i] \).
IV. EXPERIMENTS

In this section, we first present our method on a simple stochastic matrix game to show MCMARL’s ability to approximate the true return distribution and the benefits of modeling the value distribution. Then, we further evaluate the efficacy of MCMARL on StarCraft Multi-Agent Challenge (SMAC) benchmark environment [20]. Finally, we study the impact of atom number on our approach. All of our experiments are conducted on GeForce RTX 2080 Ti GPU. The implementation code is available at.

A. Evaluation on Stochastic Matrix Game

Matrix game is widely adopted to test the effectiveness of the methods [19], [24], [25]. To demonstrate the capacity of MCMARL to approximate the true return distribution, we design a two-agent stochastic matrix game. Specifically, two agents jointly take actions and will receive a joint reward, which follows a distribution rather than a deterministic value. Here, we set the joint reward to follow a normal distribution or a mixture of normal distributions, as illustrated in Fig. 7(a).

Take the MCMARL variant of QMIX as an example, we train DQMIX on the matrix game for 2 million steps with full exploitation (i.e., $\epsilon$-greedy exploration with $\epsilon = 1$). Full exploitation ensures that DQMIX can explore all the available game states, such that the representational capacity of the state-action value distribution approximation remains the only limitation [24]. As shown in Fig. 7(b), the learned global $Q$-value distributions are close to the true return distributions in terms of the mean and variance. Moreover, we visualize the true return distribution and the learned distribution of joint action in Fig. 7(c)-(f). It can be observed that the estimated distributions are extremely close to the true ones, which cannot be achievable by expected value function factorization methods.

B. Evaluation on the SMAC Benchmark

We further conduct experiments on the SMAC [20] benchmark to evaluate: 1) the performance of MCMARL compared to DFAC, in terms of the distributional framework and 2) the performance of DQMIX (MCMARL variant of a recent model QMIX) compared to MARL baselines.

Before the discussion of the results, we briefly introduce experimental settings. The experimental environment is the SMAC [20] benchmark, which is based on the popular real-time strategy game StarCraft II. The common hyperparameters of all methods are set to be the same as that in the default implementation of PyMARL. For our MCMARL framework, we set $M = 51$ (refer to C51 [13]) and choose $V_{\min} = -10$ and $V_{\max} = 20$ from preliminary experiments on SMAC. To speed up the data collection, we use parallel runners to generate a total of 20 million time steps data for each scenario and train the network with a batch of 32 episodes after collecting every eight episodes. Performance is evaluated every 10,000 time steps with 32 test episodes.

To compare the performance of MCMARL and DFAC, we test the variants of VDN and QMIX under the two distributional frameworks, as shown in Table I. DDN and DMIX are the DFAC variants of VDN and QMIX, respectively, while DVDN and DQMIX are our MCMARL variants of VDN and QMIX. It can be observed that MCMARL variants consistently achieve better performance than DFAC variants, which demonstrates the superiority of our MCMARL framework in distributional MARL. Besides, the learning curves of these methods in Fig. 8 show that DQMIX outperforms the baselines with faster convergence.

To illustrate the efficacy of DQMIX, we implement representative value-based MARL baselines. The final performance of these algorithms is presented in Table II. We can see that

---

1[Online]. Available: https://github.com/wudiymy/DQMIX

---
Fig. 7. (a) Stochastic matrix game. Each agent $g_i \in \{g_1, g_2\}$ takes an action $a_i \in \{A^{(1)}, A^{(2)}\}$ and then receives a joint reward that is sampled from the corresponding distribution in the matrix. $N(\mu, \sigma^2)$ represents the Gaussian distribution with mean $\mu$ and variance $\sigma^2$. The joint action $(A^{(2)}, A^{(2)})$ results in a bimodal distribution $\frac{1}{2}N(1, 2) + \frac{1}{2}N(8, 2)$, which means that the joint reward is sampled from $N(1, 2)$ or $N(8, 2)$ with equal probability. (b) Learned global $Q$-value distribution of DQMIX. $\mu$ and $\sigma^2$ indicate the sampled mean and the sampled variance, respectively. (c)-(f) Total return distributions of joint action $(A^{(1)}, A^{(1)})$, $(A^{(2)}, A^{(1)})$, $(A^{(1)}, A^{(2)})$, and $(A^{(2)}, A^{(2)})$, respectively. The orange line represents the true return distribution (i.e., the sampled joint reward distribution). The green histogram shows the global $Q$-value distribution learned by DQMIX.

TABLE I
AVERAGE VALUES AND STANDARD DEVIATIONS OF TEST WIN RATE (%) OF FIVE INDEPENDENT RUNS AFTER 20 MILLION TRAINING TIME STEPS OF ALGORITHMS UNDER DIFFERENT DISTRIBUTIONAL FRAMEWORKS

| Method   | Scenario                   | 10m_vs_11m | 5m_vs_6m | 2s_vs_1sc | 2c_vs_64zg | 2s3z |
|----------|----------------------------|------------|----------|-----------|------------|------|
| DFAC [19]| DDN                       | 3.68±5.67  | 77.02±4.19| 34.95±4.69| 98.75±2.50 | 88.63±4.41| 87.06±2.40|
|          | DMIX                      | 35.49±29.38| 93.14±7.69| 68.48±10.25| 98.39±1.61 | 96.37±3.91| 96.71±3.13|
| MCMARL   | DVDN                      | 12.79±9.40 | 84.79±3.91| 52.38±10.43| 99.38±1.25 | 89.57±5.13| 88.86±4.04|
|          | DQMIX                     | 64.95±8.35 | 98.35±2.13| 75.59±3.45 | 99.88±0.23 | 98.67±1.22| 99.07±1.20|

TABLE II
AVERAGE VALUES AND STANDARD DEVIATIONS OF TEST WIN RATE (%) OF FIVE INDEPENDENT RUNS AFTER 20 MILLION TRAINING TIME STEPS OF VALUE-BASED MARL ALGORITHMS AND VARIANTS FOR QMIX UNDER DISTRIBUTIONAL FRAMEWORKS

| Method   | Scenario                   | 10m_vs_11m | 5m_vs_6m | 2s_vs_1sc | 2c_vs_64zg | 2s3z |
|----------|----------------------------|------------|----------|-----------|------------|------|
| IQL [22] |                            | 12.57±12.14| 72.19±6.25| 50.96±7.97| 99.42±0.79 | 84.81±3.72| 81.69±3.69|
| VDN [23] |                            | 15.56±19.02| 84.10±8.42| 65.05±2.15| 99.38±1.25 | 85.88±6.02| 96.67±3.57|
| QMIX [24]|                            | 0.32±0.64  | 84.28±1.87| 72.33±10.03| 99.07±1.25 | 97.92±1.77| 98.09±1.34|
| QTRAN [25]|                           | 7.46±14.65 | 83.67±6.10| 29.96±7.73 | 98.08±2.57 | 90.08±4.54| 97.53±1.51|
| QPLEX [27] |                           | 86.25±2.25 | 89.22±1.79| 74.37±4.88 | 99.53±0.32 | 98.28±1.47| 97.97±0.60|
| DMIX [19]|                            | 35.49±29.38| 93.14±7.69| 68.48±10.25| 98.39±1.61 | 96.37±3.91| 96.71±3.13|
| DQMIX    |                            | 64.95±8.35 | 98.35±2.13| 75.59±3.45 | 99.88±0.23 | 98.67±1.22| 99.07±1.20|

The bold values mean that these are the best results among the algorithms in the corresponding scenarios.
Fig. 8. Test win rate curves on the SMAC benchmark. The solid line shows the median win rate, and the shadow area represents the min to max win rate on five random seeds. (a) Corridor. (b) 10m_vs_11m. (c) 5m_vs_6m. (d) 2s_vs_1sc. (e) 2c_vs_64zg. (f) 2s3z.

C. Impact of Atom Number

Furthermore, we conduct an ablation experiment to study MCMARL’s performance in relation to the number of atoms of categorical distribution, which is the core hyperparameter in our framework. Fig. 9 reports the test win rate curves of DQMIX on 5m_vs_6m by varying the number of atoms with the value of \{5, 11, 25, 51, 75\}. It can be observed that the performance is extremely poor when the number of atoms is 5, and the test win rate reaches an acceptable performance when the number is greater than 11. The results indicate that more atoms contribute to better performance, and the performance approaches saturation as the number of atoms increases. This is consistent with the fact that, given the fixed value range, support with more atoms has better expressive power and the learned distribution is more likely to be close to the true one. Considering that the performance is approaching saturation when the support size is greater than 51, we set the number of atoms to be 51 throughout this work, to balance the effectiveness and efficiency.

V. CONCLUSION

In this article, we propose MCMARL, a novel distributional value-based MARL framework, which explicitly models the stochastic in long-term returns by categorical distribution and enables to extend the existing value-based factorization MARL models to their distributional variants. To integrate the individual Q-value distributions into the global one, we design five basic distributional operations and theoretically prove that they satisfy the DIGM principle. In this way, the MCMARL variants composed of these five basic operations meet the DIGM principle, which ensures the feasibility of decentralized execution. Beyond that, empirical experiments on the stochastic matrix game and SMAC benchmark demonstrate the efficacy of MCMARL. The limitation of this work is that we do not support the multiplication of the distributions; thereby, our MCMARL framework cannot be applied to the attention-based MARL models. In the future, we will further investigate other basic operations on the distribution, such as multiplication.
REFERENCES

[1] C. Berner et al., “Dota 2 with large scale deep reinforcement learning,” 2019, arXiv:1912.06680.
[2] Y. Ye et al., “Towards playing full MOBA games with deep reinforcement learning,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2020, pp. 621–632.
[3] T. Zhang, Z. Liu, Z. Pu, and J. Yi, “Peer incentive reinforcement learning for cooperative multi-agent games,” IEEE Trans. Games, early access, doi: 10.1109/TG.2022.3196925.
[4] J. Barambones, J. Cano-Beníto, I. Sánchez-Rivero, R. Imbert, and F. Richoux, “Multi-agent systems on virtual games: A systematic mapping study,” IEEE Trans. Games, vol. 15, no. 2, pp. 134–147, Jun. 2023.
[5] J. Zhao, W. Shu, Y. Zhao, W. Zhou, and H. Li, “Improving deep reinforcement learning with mirror loss,” IEEE Trans. Games, early access, doi: 10.1109/TG.2022.3164470.
[6] C. Zhang and V. Lesser, “Coordinated multi-agent reinforcement learning in networked distributed POMDPs,” in Proc. AAAI Conf. Artif. Intell., 2011, pp. 764–770.
[7] H. Tannous et al., “GAMEREHAB, HOME: A new engineering system using serious game and multisensor fusion for functional rehabilitation at home,” IEEE Trans. Games, vol. 13, no. 1, pp. 89–98, Mar. 2021.
[8] Y. Xu and Z.-G. Wu, “Data-based collaborative learning for multi-agent system under distributed denial-of-service attacks,” IEEE Trans. Cogn. Dev. Syst., early access, doi: 10.1109/TCDS.2022.3172937.
[9] T. Zhao, M. Deng, Z. Li, and Y. Hu, “Cooperative manipulation for a mobile dual-arm robot using sequences of dynamic movement primitives,” IEEE Trans. Cogn. Dev. Syst., vol. 12, no. 1, pp. 18–29, Mar. 2018.
[10] E. Antonioni, V. Suriani, F. Riccio, and D. Nardi, “Game strategies for physical robot soccer players: A survey,” IEEE Trans. Games, vol. 13, no. 4, pp. 342–357, Dec. 2021.
[11] P. Zhou, X. Chen, Z. Liu, T. Braud, P. Hui, and J. Kangasharju, “DRLE: Decentralized reinforcement learning at the edge for traffic light control in the IoV,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 4, pp. 2262–2273, Apr. 2021.
[12] Y. Rizk, M. Awad, and E. W. Tunstel, “Decision making in multiagent systems: A survey,” IEEE Trans. Cogn. Dev. Syst., vol. 10, no. 3, pp. 514–529, Sep. 2018.
[13] M. G. Bellemare, W. Dabney, and R. Munos, “A distributional perspective on reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 449–458.
[14] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, “Implicit quantile networks for distributional reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1096–1105.
[15] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, “Distributional reinforcement learning with quantile regression,” in Proc. AAAI Conf. Artif. Intell., 2017, pp. 2892–2901.
[16] D. Yang, L. Zhao, Z. Lin, T. Qin, J. Bian, and T.-Y. Liu, “Fully parameterized quantile function for distributional reinforcement learning,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 6190–6199.
[17] J. Hu, S. A. Harding, H. Wu, S. Hu, and S.-w. Liao, “QR-mix: Distributional value function factorisation for cooperative multi-agent reinforcement learning,” 2020, arXiv:2009.04197.
[18] H. Qin et al., “RMIX: Learning risk-sensitive policies for cooperative reinforcement learning agents,” in Proc. Adv. Neural Inf. Process. Syst., 2021, vol. 34, pp. 23049–23062.
[19] W.-F. Sun, C.-K. Lee, and C.-Y. Lee, “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.
[20] S. Samvelyan et al., “The StarCraft multi-agent challenge,” in Proc. Int. Conf. Joint Auton. Agents MultiAgent Syst., 2019, pp. 2186–2188.
[21] F. A. Oliehoek and C. Amato, A Concise Introduction to Decentralized POMDPs. New York, NY, USA: Springer, 2016.
[22] M. Tan, “Multi-agent reinforcement learning: Independent versus cooperative agents,” in Proc. Int. Conf. Mach. Learn., 1993, pp. 487–494.
[23] P. Sunehag et al., “Value-decomposition networks for cooperative multi-agent learning based on team reward,” in Proc. Int. Conf. Joint Auton. Agents MultiAgent Syst., 2018, pp. 2085–2087.
[24] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, “QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 4292–4301.
[25] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, “QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 5887–5896.

Jian Zhao received the Ph.D. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2023. His research interests include reinforcement learning and multi-agent systems.

Mingyu Yang received the B.E. degree in electronic engineering and information science in 2021 from the University of Science and Technology of China, Hefei, China, where he is currently working toward the M.E. degree in information and communication engineering with the Department of Electronic Engineering and Information Science. His research interests include reinforcement learning and multi-agent systems.

Youpeng Zhao received the B.E. degree in electronic engineering and information science in 2021 from the University of Science and Technology of China, Hefei, China, where he is currently working toward the M.E. degree in information and communication engineering with the Department of Electronic Engineering and Information Science. His research interests include reinforcement learning and multi-agent systems.

Xunhan Hu received the M.E. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2023. Her research interests include reinforcement learning and multi-agent systems.
Wengang Zhou (Senior Member, IEEE) received the B.E. degree in electronic information engineering from Wuhan University, Wuhan, China, in 2006, and the Ph.D. degree in electronic engineering and information science from the University of Science and Technology of China (USTC), Hefei, China, in 2011.

From 2011 to 2013, he was a Postdoctoral Researcher with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA. He is currently a Professor with the Department of Electronic Engineering and Information Science, USTC. He has authored or coauthored more than 100 papers in IEEE/ACM transactions and CCF Tier-A international conferences. His research interests include multimedia information retrieval, computer vision, and computer game.

Dr. Zhou was the recipient of the Best Paper Award at 2012 International Conference on Internet Multimedia Computing and Service. He received the Award for the Excellent Ph.D. Supervisor of the Chinese Society of Image and Graphics in 2021 and the Award for the Excellent Ph.D. Supervisor of Chinese Academy of Sciences in 2022. He was also the recipient of the First Class Wu-Wenjun Award for Progress in Artificial Intelligence Technology in 2021 and the Outstanding Service Award at 2021 IEEE International Conference on Multimedia and Expo (ICME). He was the Publication Chair of 2021 IEEE ICME 2021. He is the winner of National Science Funds of China (National Natural Science Foundation of China) for Excellent Young Scientists. He is an Associate Editor and a Lead Guest Editor for IEEE TRANSACTIONS ON MULTIMEDIA.

Houqiang Li (Fellow, IEEE) received the B.S., M.Eng., and Ph.D. degrees in electronic engineering from the University of Science and Technology of China, Hefei, China, in 1992, 1997, and 2000, respectively.

He is currently a Professor with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. He has authored and coauthored more than 200 papers in journals and conferences. His research interests include reinforcement learning, multimedia search, image/video analysis, and video coding and communication.

Dr. Li was the recipient of the National Technological Invention Award of China (second class) in 2019, the National Natural Science Award of China (second class) in 2015, the Best Paper Award at 2012 Visual Communications and Image Processing Conference (VCIP), the Best Paper Award at 2012 International Conference on Internet Multimedia Computing and Service, and the Best Paper Award at 2011 International Conference on Mobile and Ubiquitous Multimedia. He is the winner of National Science Funds of China (National Natural Science Foundation of China) for Distinguished Young Scientists, the Distinguished Professor of Changjiang Scholars Program of China, and the Leading Scientist of Ten Thousand Talent Program of China. He is an Associate Editor for IEEE TRANSACTIONS ON MULTIMEDIA. He was an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He was the General Co-Chair of 2021 IEEE International Conference on Multimedia and Expo and the Technical Program Committee Co-Chair of VCIP 2010.