Mingling Foresight with Imagination: Model-Based Cooperative Multi-Agent Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently, model-based agents have achieved better performance than model-free ones using the same computational budget and training time in single-agent environments. However, due to the complexity of multi-agent systems, it is tough to learn the model of the environment. The significant compounding error may hinder the learning process when model-based methods are applied to multi-agent tasks. This paper proposes an implicit model-based multi-agent reinforcement learning method based on value decomposition methods. Under this method, agents can interact with the learned virtual environment and evaluate the current state value according to imagined future states in the latent space, making agents have the foresight. Our approach can be applied to any multi-agent value decomposition method. The experimental results show that our method improves the sample efficiency in different partially observable Markov decision process domains.

1 Introduction

In recent years, reinforcement learning has made remarkable achievements in game AI, robots, and autonomous driving. It is primarily due to the progress of model-free reinforcement learning (MFRL) research. Unlike model-based reinforcement learning (MBRL), MFRL does not require dynamics of the environment and learns from the data generated by directly interacting with the environment. Therefore, although most environment models are unavailable in the real world, MFRL can master complex tasks through large-capacity value function approximators. However, MFRL requires large amounts of training data that can be costly and risky in reality, which means that the sample efficiency is the main bottleneck of the current MFRL algorithms. So in the problem that the dynamics model is known, we should prioritize using MBRL, which can even obtain an analytical solution to optimal control problems.

Multi-agent systems are more complex than single-agent tasks. The strategy of each agent changes during the learning process in multi-agent systems, which makes the environment unstable from the perspective of each agent. One method [39, 10, 33, 17, 20] is for agents to use channels to transmit information to each other, in this way to reach a consensus between agents. In addition, many current studies on multi-agent reinforcement learning have focused on the paradigm of centralized training with decentralized execution (CTDE) [25]. The agent only obtains global information during training can significantly alleviate the instability problem. The first branch [25, 11, 15] that emerged is using the Actor-Critic structure's advantages to build the centralized critic and decentralized actor framework. The value decomposition method [40, 34, 38, 26, 43, 42] is also the most popular CTDE approach lately, and it has obtained excellent performance in the decentralized partial observable Markov decision process (Dec-POMDP) domains [28]. Nevertheless, most of the above methods are model-free and require access to an impractically large number of trajectories.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Some work combines model-based and model-free methods. They usually learn environment models and solve control problems simultaneously, and have achieved good performance in some single-agent scenarios, such as MuJoCo [12] and Atari [18]. However, in multi-agent problems, the training of the world model faces great challenges because of the complexity of the environments. All current work on model-based multi-agent reinforcement learning can only be applicable for simple scenarios such as matrix games. In this paper, we propose **M**odel-**B**ased **V**alue **D**ecomposition (MBVD), a method that introduces the idea of model learning into value decomposition. When humans make decisions, they not only rely on the current state but also consider the future state obtained after several interactions with the environment following their current strategies. The phenomenon is called "long-term vision" or foresight. Enlightened by this ability of humans, we make agents have the foresight to cooperate by obtaining the aggregated latent states in the future. Finally, we evaluated the performance of MBVD in several different domains, including StarCraft II [36], Google Research Football [23] and Multi-Agent MuJoCo [32]. We clarified that MBVD has high sample efficiency, which exceeds the performance of other baselines in most scenarios. To the best of our knowledge, our study is the first attempt to apply the model-based ideas to Dec-POMDP problems.

2 Related Work

2.1 Single-Agent Model-based RL

The model-based approaches are divided into three different categories. The first branch is represented by Dyna-Q [41, 24, 18, 6, 45], the method of alternating the two processes, including learning environment models and improving policies. The focus of this idea is that the world model generates trajectories used by the training of reinforcement learning. Analytic-gradient algorithms can calculate the analytic gradient related to the reinforcement learning optimization goal through the parameterized world model, thereby directly improving the policy. This method is also called Policy Search with Backpropagation through Time [13, 8, 14, 7, 12]. The last method rolls out the learned models over multiple time steps to predict the value of states. VPN [27] plans and calculates the path with the largest accumulated reward in rollouts and goes back to the current moment to improve targets. MVE [9] uses an update method similar to n-step q-learning to calculate the target state value. However, a fundamental limitation is that the above algorithms rely heavily on the world model. So if the learned environment model is inaccurate, it will lead to the collapse of the entire learning process. Especially the last method suffers from the limitation because, in addition to learning the dynamics model of the environment, model-augmented value expansion algorithms [37, 44, 16, 1, 48, 3] also need to learn the reward function of the environment to calculate the accurate value function. It is impractical in sparse reward environments.

2.2 Multi-Agent Model-based RL

So far, there is relatively little work on multi-agent model-based RL. [46] obtained the sample complexity of multi-agent model-based RL through theoretical proof when the dynamics model was available. Nevertheless, it only applies to simple infinite-horizon zero-sum discounted Markov games and has not been implemented. M^3–UCRL [31] combines model-based RL with mean-field game theory, which can be used in cooperation problems like stylized swarm motion. However, due to the limitation of the mean-field theory, M^3-UCRL can only be suitable for the scenarios of a large population of agents. Both [30] and [47] construct the environment model that includes a transition function and a prediction model for the opponents' actions, and then train their policies with the opponent-wise rollouts. These two methods have been evaluated in the multi-particle environment, and the results demonstrate that they can reduce sample complexity. However, it is not feasible to build the opponent models without access to opponents' information. CPS [2] approximates a factored sparse Q-function, which is similar to the value decomposition method. Besides, CPS learns the environment model and maintains a replay buffer with priority, but it cannot solve the partially observable problems. Similarly, in the multi-agent model-based RL field, we also face compounding errors caused by imperfect learned models.

MBVD we proposed focuses on learning a standard forward dynamics model and aggregating the information contained in the environment model rollouts. Then the current state value, which is with respect to the imagined rollouts, can be obtained. We believe the imagined information can enable all agents to understand the current situation better.

3 Preliminaries

3.1 Dec-POMDP

Partially observable stochastic games (POSGs) are one of the most general games, and the Dec-POMDP is an important subclass of POSGs. Formally, the Dec-POMDP is defined as a tuple $G = \langle S, U, A, P, r, Z, O, n, \gamma \rangle$. Each agent $a \in A := \{1, \ldots, n\}$ selects the appropriate action $u^a \in U$ at each time step, with only access to the local observation $z^a \in Z$ provided by the observation function $O(s, a) : S \times A \to Z$, where $s \in S$ is the global state of the environment. $\boldsymbol{u} \in \boldsymbol{U} \equiv U^n$ denotes the combined action of all agents. The state transition function, commonly known as the environmental dynamics, is written as $P(s' \mid s, \boldsymbol{u}) : S \times \boldsymbol{U} \times S \to [0, 1]$. Noted that all agents in Dec-POMDPs have the same reward function: $r(s, \boldsymbol{u}) : S \times \boldsymbol{U} \to \mathbb{R}$. And γ is the discount factor.

3.2 Value Decomposition

When there are multiple agents in the system, it is impossible to train each agent separately or treat all agents as one entity for joint training in cooperative multi-agent reinforcement learning. The appearance of value decomposition methods promotes the collaboration between agents and solves the credit assignment problem. In the Dec-POMDP, we make the following assumption:

$$\arg\max_{\boldsymbol{u}} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}) = \begin{pmatrix} \arg\max_{u_1} Q_1(\tau_1, u_1) \\ \vdots \\ \arg\max_{u_n} Q_n(\tau_n, u_n) \end{pmatrix},$$

where $\boldsymbol{\tau} \in T^n$ represents the joint action-observation histories of all agents, Q_{tot} is the global action-value function, and Q_n is the individual ones. The assumption is known as the Individual-Global-Max (IGM) [38] principle, which states that a task may only be decentralized if there is consistency between the local greedy actions and global ones. Many multi-agent reinforcement learning algorithms have performed well by observing the IGM principle, like VDN [40] and QMIX [34].

3.3 Environment Models

The most notable feature of MBRL is that while training the policy, it also learns the world model, a parametric model used to simulate the environment. Two different models can be learned unsupervised from local observations and actions, namely auto-regressive and state-space models.

Auto-regressive models [19, 4] are intuitive and straightforward. However, there are two reasons for the high computational complexity of the auto-regressive models: calculated items of the generation process cannot be reused, and auto-regressive models need to render high-dimensional observations explicitly. On the contrary, state-space models [29, 22] first abstract the environment to find a compact latent state space $\mathcal{S}$ containing all vital information. In the multi-agent system, each $\hat{s}_t \in \mathcal{S}$ is an abstract representation of the local observations $\boldsymbol{z}_t$ of all agents. So the observation function can be expressed by $p(\boldsymbol{z}_t \mid \hat{s}_{0:t}, \boldsymbol{u}_{0:t-1}) = p(\boldsymbol{z}_t \mid \hat{s}_t)$. Using imagined rollouts obtained by the transition function on the low-dimensional latent state space, state-space models can significantly reduce the amount of calculation. The factorization of the predictive distribution is as follows:

$$p(\boldsymbol{z}_{1:T}, \boldsymbol{u}_{0:T}) = p_{\text{init}}(\hat{s}_0) \int \prod_{t=1}^{T} \left(p\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t\right) p(\boldsymbol{u}_t \mid \boldsymbol{z}_t) p(\boldsymbol{z}_t \mid \hat{s}_t) \right) d\hat{s}_{1:T},$$

where $p_{\text{init}}(\hat{s}_0)$ denotes the prior distribution of the initial state $\hat{s}_0$, which is usually a constant. $p(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t)$ is the true posterior of the latent state, and $p(\boldsymbol{u}_t \mid \boldsymbol{z}_t)$ means the joint policy $\boldsymbol{\pi}$ of all agents.

4 Model-Based Value Decomposition

This section will elaborate on MBVD, a novel model-based multi-agent reinforcement learning algorithm based on value decomposition. We will introduce the framework and flowchart of MBVD first and then describe the detailed implementation of MBVD.

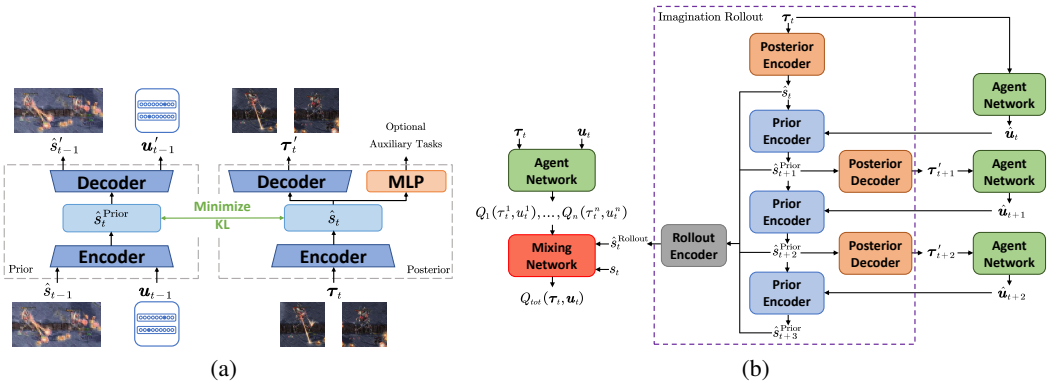


Figure 1: Illustration of MBVD implementation. (a) The imagination module in MBVD. (b) The workflow of MBVD. The rollout horizon in the figure is 3.

4.1 The MBVD Framework

To maintain the applicability of algorithms, the framework of the reinforcement learning part in MBVD is consistent with other value decomposition methods, including the agent network and the mixing network. We focus on the model learning part of MBVD, called the imagination module.

Inspired by amortized variational inference, we employ neural networks to approximate the intractable posterior $p(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t)$. The approximate posterior is defined by $q_\theta(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t)$, where θ is the parameters. So we can derive the evidence lower bound (ELBO) as follows:

$$
\begin{aligned}
&\log p\left(\boldsymbol{z}_{1:T}, \boldsymbol{u}_{0:T}\right) \\
=&\log \mathbb{E}_{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}\left[\frac{p\left(\hat{s}_{1:T}, \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}\right] \\
\geq&\mathbb{E}_{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)} \log\left[\frac{p\left(\hat{s}_{1:T}, \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}\right],
\end{aligned}
$$

and the ELBO can be broken down as:

$$
\begin{aligned}
&\mathcal{L}\left(\boldsymbol{z}_{1:T}, \boldsymbol{u}_{0:T}\right) \\
=&\sum_{t=1}^{T}\left\{\log\left[p(\boldsymbol{u}_t \mid \boldsymbol{z}_t)\right]+\log\left[p(\boldsymbol{z}_t \mid \hat{s}_t)\right]-\mathcal{D}_{\mathrm{KL}}\left[q_\theta\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t\right) \| p\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}\right)\right]\right\}.
\end{aligned} \tag{1}
$$

The first term $\log\left[p(\boldsymbol{u}_t \mid \boldsymbol{z}_t)\right]$ is the joint policy and we can ignore it. Furthermore, the term $\log\left[p(\boldsymbol{z}_t \mid \hat{s}_t)\right]$ can be viewed as the observation model. In addition, we propose a new parameterized function $p_\phi^{\text{Prior}}(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1})$ to approximate the dynamics model $p(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1})$, which is unavailable.

MBVD follows the idea of planning in latent spaces, so we use the Variational Autoencoder (VAE) [21] to maximize the ELBO. The reason is that generative models can find a low-dimensional space by reconstructing the original data, which is consistent with abstracting the decision-related state from the real observations. For the last term in Equation 1, we regard the two items in Kullback-Leibler (KL) divergence as the posterior and the prior. Intuitively, according to Equation 1, we need to minimize the KL divergence between the posterior $q_\theta(\cdot)$, which incorporates information about the current observations $\boldsymbol{z}$, with the prior $p_\phi^{\text{Prior}}(\cdot)$ that tries to predict the posterior without access to the current observations. Then we use two VAEs as the prior and the posterior, respectively, as shown in Figure 1(a). In this way, we obtain the prior latent state and the posterior latent state:

$$
\begin{aligned}
\hat{s}_t^{\text{Prior}} &\sim p_\phi^{\text{Prior}}\left(\cdot \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}\right), \\
\hat{s}_t &\sim q_\theta\left(\cdot \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t\right).
\end{aligned}
$$

However, we use a modified version of the posterior model in the implementation process. Since the hidden output h_t^a of the recurrent neural network in the agent network can be regarded as the

integration of all past information of the individual agent, the posterior $q_\theta(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t)$ can be rewritten as $q_\theta(\hat{s}_t \mid \boldsymbol{h}_t)$. The advantages of minimizing the KL loss are that we can not only guide the prior to predicting the latent posterior state distribution without the information of the current step but also control how much information the posterior abstracts from the original observation. It is worth noting that the learned prior model can predict forward in the latent space based on the actions of all agents, even if the actions are different from reality. So the imagination module in MBVD we proposed is an action-conditional environment model. From another perspective, we can view the prior model as the transition function, and the decoder $q_\theta(\boldsymbol{z}_t \mid \hat{s}_t)$ of the posterior can be regarded as the observation function. Through the above framework, we have built the imagination module that is general and model-agnostic for the multi-agent value decomposition methods.

In addition to learning the transition function and observation function, in some complex environments such as StarCraft II, the imagination module also needs to predict the feasible action set $\mathcal{A}$ to choose imagined actions more reasonably. Since the posterior can access the actual information of the current step, we can perform multiple auxiliary tasks by using the latent state $\hat{s}_t \sim q_\theta\left(\hat{s}_t \mid \boldsymbol{h}_t\right)$ as the intermediate variable to predict the additional environmental signals. The reward function $r(\cdot)$ is crucial in reinforcement learning because it steers the agents' behavior. However, predicting the reward function in noisy, sparse-reward, or complex environments is often difficult. So the imagination module in MBVD does not utilize the reward signal in the environment. MBVD adopts the implicit model-based idea without reward prediction.

The imagination model we offered has all the essential components of the environment, except the action-value model, which can evaluate the expected returns. We leave the implementation of the action-value model to the reinforcement learning process. With the help of the imagined states generated from the simulated model, the agents seem to have the foresight, which means that the agents estimate the action value based on long-term consequences rather than direct results. Thus, we can discern the role of the imagination module in MBVD.

4.2 The MBVD Flow Diagram

The generation process of imagined rollouts can be found in Figure 1(b). At time t, $\boldsymbol{\tau}_t$ refers to the trajectory history of all agents. Since the posterior encoder can be regarded as the inverse function of the observation function, we can infer the latent state $\hat{s}_t$ from $\boldsymbol{\tau}_t$. Furthermore, according to the current joint policy $\boldsymbol{\pi}$ of all agents (implemented by the agent network), we can get the actions $\hat{\boldsymbol{u}}_t \sim \boldsymbol{\pi}(\cdot \mid \boldsymbol{\tau}_t)$ that all agents should perform to maximize the estimate of the state-action value. Then the learned prior model is used to derive the next latent state $\hat{s}_{t+1}^{\text{Prior}}$ based on $\hat{s}_t$ and $\hat{\boldsymbol{u}}_t$. To get the joint action $\hat{\boldsymbol{u}}_{t+1}$ at the next step in the imagined rollouts, we need to obtain the trajectory history $\boldsymbol{\tau}'_{t+1} \sim q_\theta\left(\cdot \mid \hat{s}_{t+1}^{\text{Prior}}\right)$ corresponding to the latent state $\hat{s}_{t+1}$ through the observation function. We perform one step forward in the imagined rollout in the above way. Then we can roll out the environment model over multiple time steps by feeding the following imagined variables into the model. As mentioned above, we use all individual hidden outputs $\boldsymbol{h}$ instead of the trajectories $\boldsymbol{\tau}$ to accelerate the training speed in the implementation.

We use the latest policies of all agents as the rollout policies. Here is a reasonable and intuitive explanation: when people make a strategic decision, they always use their current policies as the imagined policies for their decision-making, rather than previous strategies that have led to poor performance. Besides, for the stability of reinforcement learning, we turn the random process involved in the imagination module during reinforcement learning into a deterministic form. For example, the imagined actions $\hat{\boldsymbol{u}}$ are selected from greedy policies. Moreover, we directly take the mean of the approximating distribution as the inferred latent state $\hat{s}$ and trajectory history $\hat{\boldsymbol{\tau}}$. However, we still use the reparameterization trick when learning the imagination module.

One k-step rollout starting with $\hat{s}_t$ can be written as $\left\{\left(\hat{s}_t, \boldsymbol{\tau}_t, \hat{\boldsymbol{u}}_t\right), \left(\hat{s}_{t+1}^{\text{Prior}}, \boldsymbol{\tau}'_{t+1}, \hat{\boldsymbol{u}}_{t+1}\right), \ldots, \left(\hat{s}_{t+k}^{\text{Prior}}, \boldsymbol{\tau}'_{t+k}, \hat{\boldsymbol{u}}_{t+k}\right)\right\}$. Since the latent state $\hat{s}$ is inferred from all agents' historical trajectories and actions, $\hat{s}$ contains the information of $\boldsymbol{\tau}$ and $\boldsymbol{u}$. Furthermore, to ensure that the parameters of the imagination module do not increase with the rollout horizon k, the set of imagined latent states $\left\{\hat{s}_t, \hat{s}_{t+1}^{\text{Prior}}, \ldots, \hat{s}_{t+k}^{\text{Prior}}\right\}$ is fed into the GRU [5] to get the aggregated rollout state $\hat{s}_t^{\text{Rollout}}$. Finally, we concatenate the aggregated rollout state $\hat{s}_t^{\text{Rollout}}$ with the real global state s_t and input them into the mixing network of the value decomposition framework. We believe that the imagined states contain information about the possible states of the future, which can help all agents evaluate the current state more accurately.

MBVD does not require a pre-trained environment model, which means the simulated model and the action-value model are all trained from scratch simultaneously. As an end-to-end efficient MBRL algorithm, MBVD can be extended to any value decomposition method with the mixing network.

4.3 Overall Learning Objective

Next, we will elaborate on the training objectives of MBVD. MBVD involves two processes: the reinforcement learning process, whose objective is to minimize the td-error; the other is learning the imagination module, which includes the optimization of the prior and the posterior. These two processes are carried out simultaneously. For the convenience of description, we define the parameters of the value decomposition framework as ψ.

MBVD can be seen as the value decomposition method with an imagination module, so the loss function for reinforcement learning is consistent with the original value decomposition method. The most obvious difference is that the global action-value function Q_{tot} is calculated with respect to the actual state s and the latent rollout state $\hat{s}^{\text{Rollout}}$. It is important to note that MBVD generates the aggregated rollout state $\hat{s}^{\text{Rollout}}$ conditioned on the past state s in the replay buffer, so we do not change the off-policy update paradigm. The loss function for reinforcement learning can be obtained:

$$\mathcal{L}_{\text{RL}} = \left(y^{tot} - Q_{tot}(\boldsymbol{\tau}_t, \boldsymbol{u}_t, s_t, \hat{s}_t^{\text{Rollout}}; \psi)\right)^2,$$

where $y^{tot} = r_t + \gamma \max_{\boldsymbol{u}_{t+1}} Q_{tot}\left(\boldsymbol{\tau}_{t+1}, \boldsymbol{u}_{t+1}, s_{t+1}, \hat{s}_{t+1}^{\text{Rollout}};\ \psi^-\right)$, and ψ^- represents the parameters of the target network.

The loss function of the posterior can be divided into reconstruction loss and KL divergence loss. The posterior model infers the current latent state after the observations of all agents are given, which requires that the posterior model extract helpful information from the original input. It can be achieved by narrowing the difference between the model output $\boldsymbol{\tau}'$ and the model input $\boldsymbol{\tau}$. The reconstruction loss function of the prior model is similar, and both of them can be computed by:

$$\mathcal{L}_{\text{RC}} = \text{MSE}\left(\boldsymbol{\tau}_t, \boldsymbol{\tau}'_t; \theta\right), \qquad \mathcal{L}_{\text{RC}}^{\text{Prior}} = \text{MSE}\left(\left(\hat{s}_{t-1}, \boldsymbol{u}_{t-1}\right), \left(\hat{s}'_{t-1}, \boldsymbol{u}'_{t-1}\right); \phi\right),$$

where MSE means the mean square error. Furthermore, in addition to the KL term in Equation 1, we throw in the KL divergence between the prior distribution and the standard Gaussian distribution $\mathcal{N}(0,1)$ as a regular term to avoid the overly complex prior distribution of the latent state $\hat{s}^{\text{Prior}}$. So the KL loss is as follows:

$$\begin{aligned}\mathcal{L}_{\text{KL}} = &\mathcal{D}_{\text{KL}}\left[p_\phi^{\text{Prior}}\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}\right) \| \mathcal{N}(0,1)\right] \\ &+ \mathcal{D}_{\text{KL}}\left[q_\theta\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t\right) \| p_\phi^{\text{Prior}}\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}\right)\right].\end{aligned}$$

We use the KL balancing mechanism to optimize the KL term in the actual implementation. KL balancing enables the prior to minimize KL loss at a faster learning rate than the posterior, which can be expressed as:

$$\begin{aligned}&\mathcal{D}_{\text{KLbalancing}}\left[q_\theta(\cdot) \| p_\phi^{\text{Prior}}(\cdot)\right] = \\ &\qquad \alpha \mathcal{D}_{\text{KL}}\left[q_\theta(\cdot) \| \operatorname{Detach}\left(p_\phi^{\text{Prior}}(\cdot)\right)\right] + (1-\alpha)\mathcal{D}_{\text{KL}}\left[\operatorname{Detach}\left(q_\theta(\cdot)\right) \| p_\phi^{\text{Prior}}(\cdot)\right],\end{aligned}$$

where $\alpha \in [0,1]$ is a configurable parameter, $\operatorname{Detach}(\cdot)$ can stop the backpropagation from gradients of certain variables or functions.

In addition, we have other optional training objectives for auxiliary tasks in complex scenarios. In this paper, we make predictions of the feasible action set only in StarCraft II:

$$\mathcal{L}_{\text{FA}} = \text{BCE}\left(\mathcal{A}_t, \mathcal{A}'_t; \phi\right),$$

where BCE denotes the binary cross-entropy error. Thus, the total loss function can be written as:

$$\mathcal{L} = \mathcal{L}_{\text{RL}} + \mathcal{L}_{\text{RC}} + \mathcal{L}_{\text{RC}}^{\text{Prior}} + \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{FA}}. \tag{2}$$

By minimizing the total loss function $\mathcal{L}$, we can guide MBVD to accelerate reinforcement learning with the help of the imagination module.

5 Experiments

This section will consider whether the value decomposition method can benefit from the implicit model-based method. We compare the performance of MBVD with other popular baselines, including VDN, QMIX, MAVEN [26], Weighted QMIX [35], ROMA [42], and RODE [43]. Then by conducting the ablation experiments, we can clarify that every component in the learned model is indispensable, and different horizons of the rollout have a significant impact on performance. Finally, we will visualize the imagined rollouts, which will more intuitively and powerfully show the foresight of MBVD. Note that the implementation of agents in MBVD in all experiments is based on QMIX. The details of all experiments can be found in Appendix B.

(a) StarCraft II (b) Google Research Football (c) Multi-Agent Discrete MuJoCo

Figure 2: Examples of the three experimental platforms.

5.1 Performance on StarCraft II

SMAC is a multi-agent micro-management experimental platform based on the real-time strategy game StarCraft II, which contains a wealth of scenarios corresponding to different challenges. To verify the performance improvement brought by the imagination module, we pay more attention to the performance comparison between MBVD and QMIX. We select some representative scenarios. The median performance and the 25-75% percentiles are illustrated in Figure 3.

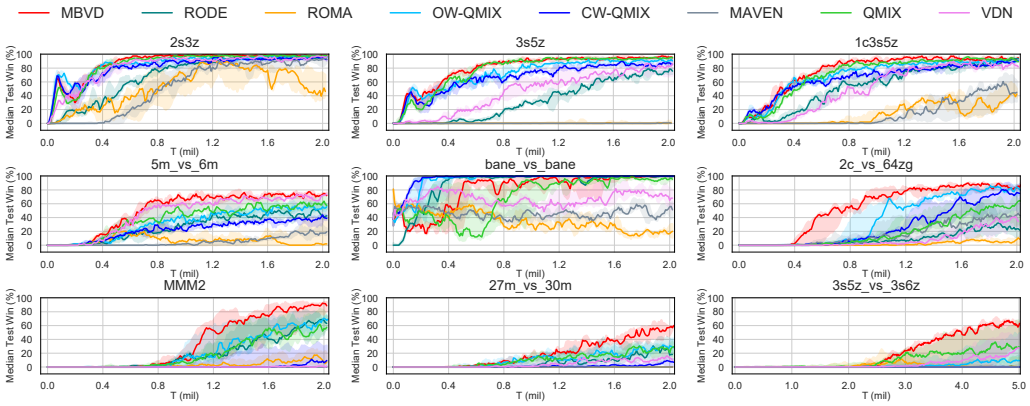


Figure 3: Performance comparison with baselines in different SMAC scenarios.

MBVD reaches state-of-the-art performance in most scenarios and exceeds the basic algorithm QMIX. Especially in hard or super hard scenarios such as *2c_vs_64zg*, *MMM2*, and *3s5z_vs_3s6z*, the superiority of MBVD is more significant. In *5m_vs_6m*, due to the uncertainty of the environment, other baselines will encounter a performance bottleneck. However, MBVD can predict the future state to achieve the best performance in *5m_vs_6m*. Even in some easy scenarios, MBVD still has high sampling efficiency. It is impossible for some complex QMIX-style variants. MBVD can robustly reduce the sampling complexity in both easy and hard scenarios.

5.2 Performance on Google Research Football

The Google Research Football environment provides a novel reinforcement learning environment where multiple agents can be trained to play football. To verify the effectiveness of our proposed method, we carry out MBVD and other baselines on the Football Academy, which is a diverse set of mini scenarios of varying difficulty. We select three representative official scenarios, and the experimental results are delivered in Figure 4.

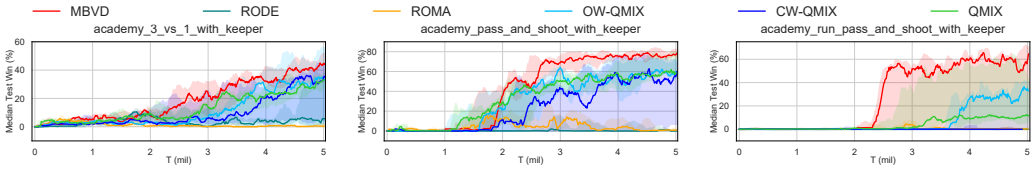


Figure 4: Performance comparison with baselines in different Google Research Football scenarios.

Compared with QMIX, which has a relatively simple structure, algorithms such as RODE and ROMA do not perform well. A possible reason is that the number of agents in these scenarios is not large, and the role assignments may hinder the learning. Conversely, MBVD based on QMIX achieves the highest sample efficiency in three scenarios with different difficulty levels. The experimental results obtained in Google Research Football also fully demonstrate the generalization capability of MBVD.

5.3 Performance on Multi-Agent Discrete MuJoCo

Many model-based reinforcement learning algorithms, including the work mentioned above, tend to be compared on MuJoCo. [32] recently proposed a novel benchmark for continuous cooperative multi-agent robotic control in the multi-agent field, called Multi-Agent MuJoCo. A given single robotic agent is viewed as a body graph containing many disjoint sub-graphs, and each sub-graph contains one or more joints that can be controlled. In this paper, each joint is regarded as an agent that makes local decisions conditioned on partial observations. Since this paper focuses on the impacts of the world model, we propose a discrete variant of Multi-Agent MuJoCo. Detailed environment settings can be found in Appendix B.3. We selected four representative scenarios, and Figure 5 describes the episode return of each algorithm. Since most of the relatively complex algorithms cannot converge in this environment, we only choose MBVD, QMIX, and VDN for comparison.

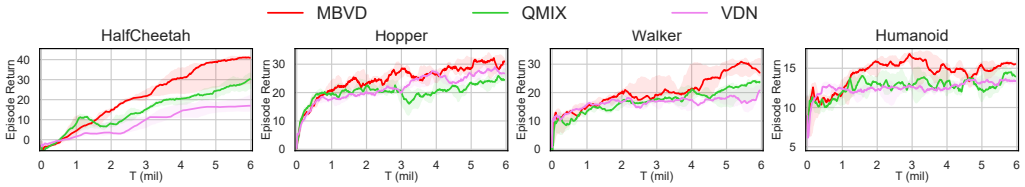
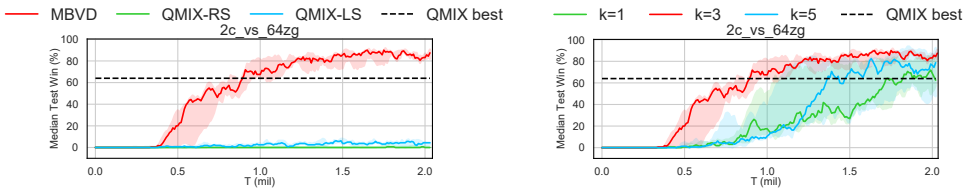


Figure 5: Median episode return on different Multi-Agent Discrete MuJoCo tasks.

From the results, we can intuitively see the performance improvement brought by the imagination module because we see MBVD as a variant of QMIX that has an additional implicit foresight module. MBVD outperformed QMIX in all given scenarios, which also means MBVD we proposed can be applied to different environments.

5.4 Ablation Studies

We carry out ablation studies to test the contribution of the components of the imagination module and investigate how the rollout horizon k affects the performance of MBVD. For the first study, we proposed two variants of QMIX that input the additional information to the mixing network, QMIX-RS and QMIX-LS. Both of them aggregate the information of the next k steps from the actual trajectories rather than imagined rollouts. However, the difference is that QMIX-RS uses the real states and QMIX-LS the latent states. To answer the second problem, we run MBVD with different rollout horizons on the *2c_vs_64zg* scenario of SMAC and compare their performance.



(a) Win rates for MBVD and the ablation.

(b) Influence of the k for MBVD.

Figure 6: Results for ablation studies on *2c_vs_64zg* map.

As shown in Figure 6(a), both QMIX-RS and QMIX-LS failed to solve the task, which means that the transition function that can generate the imagined rollouts is critical. Besides, we can also infer the contribution of the observation function from that the better performance of QMIX-LS than QMIX-RS. For the horizon of the rollouts, we can conclude from Figure 6(b). When using short-horizon rollouts, the agents cannot fully interact with the imagined model, resulting in the same performance as vanilla QMIX. However, as k increases significantly, MBVD loses the monotonic improvements because of the compounding error. The effect of k holds the same for the other scenarios, but the optimal choice of k in each task is different. If not explicitly stated, we will set k to 3 in this paper for convenience.

5.5 Visualization

To intuitively explain agents' foresight ability in MBVD, we visualized the latent state sequence generated by the interaction between agents and the imagined model. For the trajectories of an episode in the *2c_vs_64zg* scenario, we visualized the t-sne embeddings of the latent states predicted at each step in the rollout and compared them with the real latent state embedding. From Figure 7, we find that the difference between the embedding of the imagined latent states and real ones gradually increases as the horizon length grows, but there are similarities between them. It also reveals that our proposed imagination module can capture and learn the dynamics of the multi-agent environment.

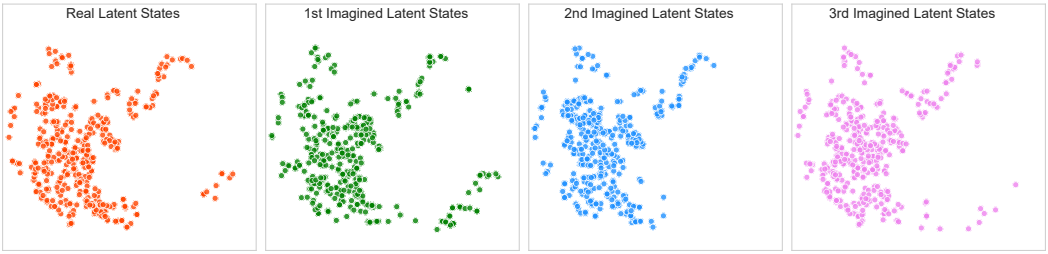


Figure 7: The 2D t-SNE embedding of real latent states and imagined latent states.

6 Conclusion

Due to the high complexity of multi-agent systems, MBRL algorithms are difficult to be applied to them. This paper proposes MBVD, a novel and implicit model-based cooperative multi-agent reinforcement learning method. By learning the world model and imagining the future latent states after making several decisions under the current policy to estimate the current state value, agents in MBVD obtain the foresight ability. Through experiments and visualization, we have proved the efficiency and generalization of MBVD. This is the first study on Dec-POMDPs from the perspective of model-based reinforcement learning.

The defect in this study is that the rollout horizon is manually chosen. In our future research, we intend to concentrate on how to choose the appropriate rollout horizon. The issue is an intriguing one which could be usefully explored in further research.

References

[1] Zaheer Abbas, Samuel Sokota, Erin Talvitie, and Martha White. Selective dyna-style planning under limited model capacity. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1–10. PMLR, 2020.

[2] Eugenio Bargiacchi, Timothy Verstraeten, Diederik M. Roijers, and Ann Now'e. Model-based multi-agent reinforcement learning with cooperative prioritized sweeping. *ArXiv*, abs/2001.07527, 2020.

[3] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8234–8244, 2018.

[4] Lars Buesing, Théophane Weber, Sébastien Racanière, S. M. Ali Eslami, Danilo Jimenez Rezende, David P. Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, and Daan Wierstra. Learning and querying fast generative models for reinforcement learning. *ArXiv*, abs/1802.03006, 2018.

[5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179.

[6] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization. *ArXiv*, abs/1809.05214, 2018.

[7] Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 465–472. Omnipress, 2011.

[8] Michael Fairbank. Reinforcement learning by value gradients. *ArXiv*, abs/0803.3539, 2008.

[9] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *ArXiv*, abs/1803.00101, 2018.

[10] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2137–2145, 2016.

[11] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2974–2982. AAAI Press, 2018.

[12] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[13] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[14] Nicolas Heess, Gregory Wayne, David Silver, Timothy P. Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2944–2952, 2015.

[15] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2961–2970. PMLR, 2019.

[16] Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[17] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7265–7275, 2018.

[18] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[19] Mehdi Khashei, Mehdi Bijari, and Gholam Ali Raissi Ardali. Improvement of auto-regressive integrated moving average models using fuzzy logic and artificial neural networks (anns). *Neurocomputing*, 72: 956–967, 2009.

[20] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[21] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[22] Jayant Kulkarni and Liam Paninski. State-space decoding of goal-directed movements. *IEEE Signal Processing Magazine*, 25:78–86, 2008.

[23] Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google research football: A novel reinforcement learning environment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4501–4510. AAAI Press, 2020.

[24] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[25] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6379–6390, 2017.

[26] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: multi-agent variational exploration. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7611–7622, 2019.

[27] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6118–6128, 2017.

[28] Frans A. Oliehoek and Chris Amato. A concise introduction to decentralized pomdps. In *SpringerBriefs in Intelligent Systems*, 2016.

[29] Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua T. Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of Computational Neuroscience*, 29:107–126, 2009.

[30] Young Joon Park, Yoon Sang Cho, and Seoung Bum Kim. Multi-agent reinforcement learning with approximate model learning for competitive games. *PLoS ONE*, 14, 2019.

[31] Barna Pasztor, Ilija Bogunovic, and Andreas Krause. Efficient model-based multi-agent mean-field reinforcement learning. *ArXiv*, abs/2107.04050, 2021.

[32] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems*, 2021.

[33] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv: Artificial Intelligence*, 2017.

[34] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4292–4301. PMLR, 2018.

[35] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[36] Mikayel Samvelyan, Tabish Rashid, C. S. D. Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *ArXiv*, abs/1902.04043, 2019.

[37] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, L. Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 7839:604–609, 2020.

[38] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5887–5896. PMLR, 2019.

[39] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with back-propagation. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2244–2252, 2016.

[40] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech M. Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning. *ArXiv*, abs/1706.05296, 2018.

[41] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16:285–286, 2005.

[42] Tonghan Wang, Heng Dong, Victor R. Lesser, and Chongjie Zhang. ROMA: multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9876–9886. PMLR, 2020.

[43] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: learning roles to decompose multi-agent tasks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[44] Chenjun Xiao, Yifan Wu, Chen Ma, Dale Schuurmans, and Martin Müller. Learning to combat compounding-error in model-based reinforcement learning. *ArXiv*, abs/1912.11206, 2019.

[45] Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based reinforcement learning with theoretical guarantees. *ArXiv*, abs/1807.03858, 2019.

[46] Kaiqing Zhang, Sham M. Kakade, Tamer Basar, and Lin F. Yang. Model-based multi-agent RL in zero-sum markov games with near-optimal sample complexity. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[47] Weinan Zhang, Xihuai Wang, Jian Shen, and Ming Zhou. Model-based multi-agent policy optimization with adaptive opponent-wise rollouts. In *IJCAI*, 2021.

[48] Bo Zhou, Hongsheng Zeng, Fan Wang, Yunxiang Li, and Hao Tian. Efficient and robust reinforcement learning with uncertainty-based value expansion. *ArXiv*, abs/1912.05328, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Derivation of the ELBO

The complete derivation of the ELBO in Equation 1 is as follows.

$$
\begin{aligned}
&\log p\left(\boldsymbol{z}_{1:T}, \boldsymbol{u}_{0:T}\right) \\
=&\log \mathbb{E}_{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}\left[\frac{p\left(\hat{s}_{1:T}, \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}\right] \\
\geq&\mathbb{E}_{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)} \log\left[\frac{p\left(\hat{s}_{1:T}, \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}\right] \\
=&\int q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right) \log\left[\frac{p\left(\hat{s}_{1:T}, \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}{q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right)}\right] d\hat{s}_{1:T} \\
=&\int \sum_{t=1}^{T} q_\theta\left(\hat{s}_{1:T} \mid \boldsymbol{u}_{0:T}, \boldsymbol{z}_{1:T}\right) \log\left[\frac{p(\boldsymbol{u}_t \mid \boldsymbol{z}_t)\, p(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1})\, p(\boldsymbol{z}_t \mid \hat{s}_t)}{q_\theta(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t)}\right] d\hat{s}_{1:T} \\
=&\sum_{t=1}^{T}\Bigg\{\int q_\theta\left(\hat{s}_{1:t} \mid \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t}\right) \log\left[p(\boldsymbol{u}_t \mid \boldsymbol{z}_t)\, p(\boldsymbol{z}_t \mid \hat{s}_t)\right] d\hat{s}_{1:t} \\
&+\int q_\theta\left(\hat{s}_{1:t} \mid \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t}\right) \log\left[\frac{p(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1})}{q_\theta(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t)}\right] d\hat{s}_{1:t}\Bigg\} \\
=&\sum_{t=1}^{T}\Bigg\{\int q_\theta\left(\hat{s}_{1:t} \mid \boldsymbol{u}_{0:t}, \boldsymbol{z}_{1:t}\right) \log\left[p(\boldsymbol{u}_t \mid \boldsymbol{z}_t)\, p(\boldsymbol{z}_t \mid \hat{s}_t)\right] d\hat{s}_{1:t} \\
&-\int q_\theta\left(\hat{s}_{1:t-1} \mid \boldsymbol{u}_{0:t-1}, \boldsymbol{z}_{1:t-1}\right) \mathcal{D}_{\mathrm{KL}}\left[q_\theta\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t\right) \| p\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}\right)\right] d\hat{s}_{1:t}\Bigg\} \\
=&\sum_{t=1}^{T}\left\{\log\left[p(\boldsymbol{u}_t \mid \boldsymbol{z}_t)\right] + \log\left[p(\boldsymbol{z}_t \mid \hat{s}_t)\right]\right. \\
&\left.-\mathcal{D}_{\mathrm{KL}}\left[q_\theta\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{z}_t\right) \| p\left(\hat{s}_t \mid \hat{s}_{t-1}, \boldsymbol{u}_{t-1}\right)\right]\right\},
\end{aligned}
\tag{3}
$$

where the inequality in Equation 3 is obtained via Jensen's inequality.

B Experimental Setup

In order to clarify the generalization of our proposed method, we used three different Dec-POMDP domains, SMAC, Google Research Football, and Multi-Agent Discrete MuJoCo, as experimental platforms. All experiments in this paper are carried out with five different seeds on Nvidia GeForce RTX 3090 and Intel(R) Xeon(R) Platinum 8280. We use the official codes for other baseline algorithms, and the hyperparameters are consistent with their original work. Next, we will introduce the settings of the three environments, respectively.

B.1 SMAC

In SMAC, each agent can only obtain entity information within the visible range. The goal of training is to guide the allied agents to defeat the enemy units, so the reward function is related to the health value of all enemy agents. Besides, agents can obtain the current set of available actions. We set the dimension of the latent state in SMAC as the product of 16 and the number of allied agents, and other training hyperparameters for MBVD follow that of QMIX. For MBVD, each independent experiment takes 8 to 30 hours, which is the same as the time spent by QMIX.

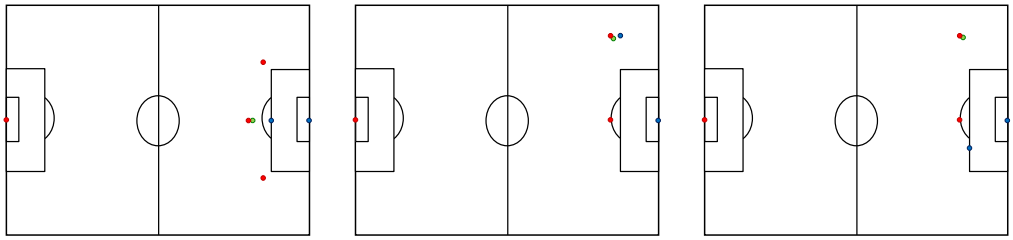
We used StarCraft version SC2.4.6.2.69232 instead of the relatively easy version SC2.4.10. The results for different versions are not directly comparable since the underlying dynamics differ. Table 1 provides an overview of the SMAC scenarios. The recognized difficulties of the scenarios are also determined based on the version SC2.4.6.2.69232 of StarCraft.

Name	Ally Units	Enemy Units	Type	Difficulty
2s3z	2 Stalkers 3 Zealots	2 Stalkers 3 Zealots	Heterogeneous Symmetric	Easy
3s5z	3 Stalkers 5 Zealots	3 Stalkers 5 Zealots	Heterogeneous Symmetric	Easy
1c3s5z	1 Colossus 3 Stalkers 5 Zealots	1 Colossus 3 Stalkers 5 Zealots	Heterogeneous Symmetric	Easy
5m_vs_6m	5 Marines	6 Marines	Homogeneous Asymmetric	hard
bane_vs_bane	4 Banelings 20 Zerglings	4 Banelings 20 Zerglings	Heterogeneous Symmetric	hard
2c_vs_64zg	2 Colossi	64 Zerglings	Homogeneous Asymmetric Large Action Space	hard
MMM2	1 Medivac 2 Marauders 7 Marines	1 Medivac 3 Marauder 8 Marines	Heterogeneous Asymmetric Macro tactics	Super Hard
27m_vs_30m	27 Marines	30 Marines	Homogeneous Asymmetric Massive Agents	Super Hard
3s5z_vs_3s6z	3 Stalkers 5 Zealots	3 Stalkers 6 Zealots	Heterogeneous Asymmetric	Super Hard

Table 1: Maps in different scenarios.

B.2 Google Research Football

In Google Research Football, we need to train our players to kick the ball into the opponent's goal. We chose three official scenarios in the Football Academy: *academy_3_vs_1_with_keeper*, *academy_pass_and_shoot_with_keeper*, and *academy_run_pass_and_shoot_with_keeper*. The initial positions of all players and the ball in the three scenarios are shown in Figure 9. We control all the agents in red (except the goalkeeper on the far left) against the agents in blue, which are controlled by built-in AI. Our players have 19 discrete actions, including moving, passing, shooting ,and so on. Unlike SMAC, the agents in Google Research Football cannot know which actions are feasible. The global state of the environment includes the two-dimensional position coordinates and moving directions of all agents on the field, as well as the three-dimensional position coordinates and moving directions of the football. The physical meaning of the local observation is the same as that of the global state, except that the absolute positions of all entities are replaced with relative ones.



(a) (b) (c)

Figure 8: The initial position of each agent in the Google Research Football scenarios considered in our paper: (a) *academy_3_vs_1_with_keeper*, (b) *academy_pass_and_shoot_with_keeper*, and (c) *academy_run_pass_and_shoot_with_keeper*. The red dots represent our players, and the blue dots denote the opposing players. The football is represented by green dots.

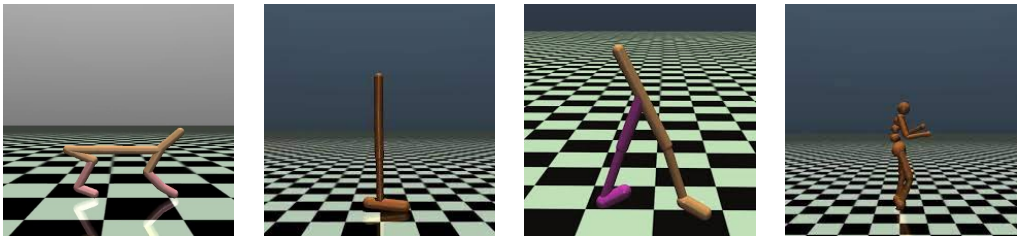
For the reward function, in addition to the reward for scoring a goal, we also added an additional reward contribution for moving the ball close to the opponent's goal, similar to the official CHECKPOINT reward function in the Football Engine. To increase the game's difficulty and speed up the

agent's training, we set the episode to end when the ball returns to the left half. Besides, scoring goals and reaching the maximum time step will also cause the episode to be terminated.

Due to the small number of agents in Google Research Football scenarios, we adjusted the latent state dimension to the product of 8 and the number of our players. In addition, since there is no information about the feasible action set in this environment, we ignore the $\mathcal{L}_{\text{FA}}$ item in Equation 2. All experiments in Google Research Football were completed within two days.

B.3 Multi-Agent Discrete MuJoCo

In the original Multi-Agent MuJoCo, the action space of each joint is $[-1, 1]$. To accommodate algorithms like QMIX, We discretize the action space into K equally spaced atomic actions. The set of atomic actions for any joint is $\mathcal{A} = \left\{ \frac{2j}{K-1} - 1 \right\}_{j=0}^{K-1}$. We treat each joint as an agent, and joints are connected by adjacent edges. A configurable parameter $l \geq 0$ determines the maximum graph distance to the agent at which joints are observable. The agent observation is then given by a fixed order concatenation of the representation vector of each observable joint. Furthermore, all features in the state are normalized. With the above modifications, we get a benchmark for cooperative multi-agent robotic control with discrete action spaces. In this paper, we set $K = 31$ and $l = 1$.



(a) HalfCheetah (6) (b) Hopper (3) (c) Walker (6) (d) Humanoid (17)

Figure 9: Illustration of benchmark tasks in Multi-Agent MuJoCo. The numbers in brackets mean the number of joints (agents) contained in each robot.

B.4 Hyperparameters

In this paper, we use the QMIX-style framework with its default hyperparameters suggested by the original paper for the reinforcement learning process of MBVD. Table 2 presents the hyperparameters of MBVD. We use n to represent the number of our agents in the environment.

Description	**Value**
Type of optimizer	RMSProp
RMSProp param α	0.99
RMSProp param ϵ	0.00001
Learning rate	0.0005
How many episodes to update target networks	200
Reduce global norm of gradients	10
Batch size	32
Capacity of replay buffer (in episodes)	5000
Discount factor γ	0.99
Starting value for exploraton rate annealing	1
Ending value for exploraton rate annealing	0.05
Horizon of the imagined rollout k	3
KL balancing α	0.3
Dimension of the latent state $\hat{s}$ in SMAC	$n\times16$
Dimension of the latent state $\hat{s}$ in Google Research Football	$n\times8$
Dimension of the latent state $\hat{s}$ in Multi-Agent Discrete MuJoCo	$n\times8$
Dimension of the aggregated rollout state $\hat{s}^{\text{Rollout}}$	Same as that of the real state s

Table 2: Hyperparameter settings.