Latent Inference for Effective Multi-Agent Reinforcement Learning under Partial Observability

Salma Kharrat[*] KAUST

Fares Fourati[*] KAUST

Marco Canini KAUST

Mohamed-Slim Alouini KAUST

Vaneet Aggarwal Purdue University

Abstract

Partial observability remains a core challenge in cooperative multi-agent reinforcement learning (MARL), often causing poor coordination and suboptimal policies. We show that state-of-the-art methods fail even in simple settings under partial observability. To address this, we propose LIMARL, a latent-inference framework that augments centralized training with decentralized execution (CTDE) via structured latent representations. LIMARL integrates (i) a state representation module that learns compact global state embeddings, and (ii) a recurrent inference module that enables agents to recover these embeddings from local histories. We provide theoretical analysis on sufficiency and robustness under partial observability. Empirically, LIMARL outperforms strong baselines in diagnostic tasks and challenging SMAC and SMACv2 scenarios, demonstrating better performance and faster convergence. Our results highlight latent inference as an effective and scalable solution for partially observable MARL. An implementation of LIMALR is available at `https://github.com/salmakh1/LIMARL`.

1 Introduction

Reinforcement Learning (RL) provides a framework for sequential decision-making via interaction with an environment to maximize long-term reward [39]. Multi-Agent RL (MARL) extends this framework to settings involving multiple autonomous agents, where coordination and partial observability introduce fundamental challenges. MARL has driven advances across diverse domains, including autonomous driving [36, 29, 6, 49], dynamic ride sharing [1, 14], collaborative robotics [47], distributed resource management [26, 3], and traffic engineering [11]. By learning adaptive policies through interaction, MARL offers a scalable and principled approach to multi-agent systems.

A wide range of MARL methods have emerged to address coordination and scalability. Fully centralized approaches are limited by communication and scalability constraints [2], while independent learning, though scalable, suffers from instability due to non-stationarity. Therefore, Centralized Training with Decentralized Execution (CTDE) has become the dominant paradigm, leveraging global information during training while maintaining decentralized execution.

Within CTDE, key methods include Value Decomposition Networks (VDN) [38] and QMIX [32], which decompose joint value functions to enable tractable learning. Extensions improve expressiveness via advantage-based decomposition [43], adaptive weighting [30], or attention mechanisms [42, 45]. However, partial observability remains a central challenge: during execution, agents generally lack access to the global state and must act based on local, noisy observations.

[*]Equal contribution. Contact: `salma.kharrat@kaust.edu.sa`, `fares.fourati@kaust.edu.sa`

18th European Workshop on Reinforcement Learning (EWRL 2025).

To highlight the challenges current MARL methods face under partial observability, we introduce a simple yet revealing two-agent, three-state matrix game (Figure 1). Despite its simplicity, this setting exposes fundamental limitations in existing approaches, which fail to learn the optimal policy. Motivated by this, we propose a novel method within the CTDE framework that directly tackles partial observability through explicit latent state learning and inference.

Unlike prior work that relies on full-state access during training to learn mixing functions, our method introduces a structured latent inference mechanism. We augment CTDE with two components: (1) a *State Representation Module (SRM)* that encodes global states into compact latent embeddings, and (2) a *Recurrent Observation-to-Latent Inference Module (ROLIM)* that infers these embeddings from local action-observation histories. During training, SRM supervises ROLIM via an alignment loss to ensure inferred latents capture essential state information. During execution, agents use ROLIM to infer latent states, which are then combined with local observations to estimate utilities, enabling more effective decision-making under partial observability. By modeling latent dynamics, our approach reduces uncertainty and improves policy learning under partial observability.

We evaluate our method on diagnostic tasks and standard cooperative multi-agent benchmarks, demonstrating strong performance across diverse scenarios. Results show significant improvements over existing methods. These findings highlight the potential of integrating representation learning and recurrent inference with MARL to address partial observability and enable more effective systems.

The main contributions are as follows:

(1) A latent inference framework for MARL under partial observability: We propose LIMARL, a method that unifies self-supervised state representation learning and recurrent latent inference from local agent histories, within the CTDE paradigm. This enables agents to approximate global state information during decentralized execution.

(2) Theoretical guarantees for decentralized latent-based control: We show that LIMARL preserves optimality under bounded reconstruction error, enables globally optimal coordination via decentralized greedy policies with monotonic mixing, and maintains stable utility estimates under latent inference errors.

(3) Comprehensive empirical validation across cooperative benchmarks: We evaluate LIMARL on both diagnostic tasks and complex SMAC and SMACv2 scenarios, demonstrating improved or competitive performance compared to recent baselines.

2 Background

2.1 Decentralized Partially Observable Markov Decision Process (Dec-POMDP)

In this paper, we consider a fully cooperative multi-agent task modeled as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [27]. A Dec-POMDP is formally defined as $\mathcal{G} = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, \Omega, O, r, \gamma \rangle$, where $\mathcal{N} \triangleq \{1, \ldots, n\}$ represents a finite set of agents, and $\mathcal{S}$ is the set of global states of the environment. Each agent can choose from a finite set of actions $\mathcal{A}$, and the state transition probability $P(s'|s, \boldsymbol{a})$ determines the likelihood of transitioning from state s to state s' given the joint action $\boldsymbol{a} \in \mathcal{A}^n$. The set of observations for each agent is denoted by Ω, with $O(o^i|s, a^i)$ specifying the probability of agent i observing o^i given state s and its action $a^i \in \mathcal{A}$. The reward function $r(s, \boldsymbol{a})$ is shared among all agents, and $\gamma \in [0, 1)$ represents the discount factor.

At each time step, each agent $i \in \mathcal{N}$ observes $o^i \in \Omega$ and selects an action a^i from its action space. The joint action $\boldsymbol{a}$ induces a transition to the next state $s' \sim P(\cdot|s, \boldsymbol{a})$, and a global reward $r(s, \boldsymbol{a})$ is received. Due to partial observability, each agent i maintains an action-observation history $\tau^i \in \mathcal{T} \equiv (\Omega \times \mathcal{A})^*$ and constructs its individual policy $\pi^i(a|\tau^i)$. The objective is to find a joint policy $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_n \rangle$ that maximizes the expected joint value function: $V^{\boldsymbol{\pi}}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \boldsymbol{\pi}\right]$. The joint action-value function is given by: $Q^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s'}\left[V^{\boldsymbol{\pi}}(s')\right]$.

2.2 Centralized Training with Decentralized Execution (CTDE)

Learning optimal policies in the Dec-POMDP framework is challenging due to non-stationarity and limited information available to each agent. CTDE has emerged as a promising paradigm for MARL, especially in cooperative settings [38, 32, 37, 45, 43]. CTDE allows agents to share information

during training to facilitate centralized decision-making, while during execution, agents operate using only their local observations and action histories.

A key principle in value-based CTDE is the *Individual-Global-Max (IGM)* [37], which ensures consistency between joint and individual action-value functions. The IGM principle mandates that the joint greedy action selection under the joint action-value function $Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a})$, where $\boldsymbol{\tau}$ represents the joint histories of all agents, aligns with individual greedy selections under the individual action-value functions $\{Q_i(\tau^i, a^i)\}_{i=1}^n$, as expressed:

$$\forall \boldsymbol{\tau} \in \mathcal{T}^n, \arg\max_{\boldsymbol{a} \in \mathcal{A}^n} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \left(\arg\max_{a^i \in \mathcal{A}} Q_i(\tau^i, a^i) \right)_{i=1}^n.$$

To achieve this, factorization structures are employed to decompose the joint action-value function. Popular approaches include: VDN [38], which represent the joint action-value as a sum of individual action-values: $Q_{\text{tot}}^{\text{VDN}}(\boldsymbol{\tau}, \boldsymbol{a}) = \sum_{i=1}^n Q_i(\tau^i, a^i)$. QMIX [32], which enforces a monotonic relationship between the joint and individual action-values: $\forall i \in \mathcal{N}, \quad \frac{\partial Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q_i(\tau^i, a^i)} > 0$. However, both VDN and QMIX enforce sufficient but not necessary conditions for the IGM constraint, limiting their expressiveness [24]. Extensions such as Qatten [45] integrate multi-head attention mechanisms [42]. In contrast, QTRAN [37] reformulates IGM as a linear condition and applies it as a soft regularization term. W-QMIX [30] introduces weighting mechanisms to emphasize high-quality joint actions.

2.3 Deep Multi-Agent Q-Learning in Dec-POMDP

Q-learning algorithms are widely used for optimizing joint action-value functions in value-based MARL. The optimal joint action-value function is defined as:

$$Q^*(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s'} \left[\max_{\boldsymbol{a'}} Q^*(s', \boldsymbol{a'}) \right].$$

Deep Q-learning approximates the action-value function using a neural network parameterized by $\boldsymbol{\theta}$ [25, 41, 10]. In the Dec-POMDP setting, $Q(\boldsymbol{\tau}, \boldsymbol{a}; \boldsymbol{\theta})$ replaces $Q(s, \boldsymbol{a}; \boldsymbol{\theta})$ due to partial observability, and is often decomposed into local utilities, as discussed in Section 2.2. The parameters $\boldsymbol{\theta}$ are learned by minimizing the expected temporal difference (TD) error:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{\tau}, \boldsymbol{a}, r, \boldsymbol{\tau'}) \in D} \left[\left(r + \gamma V(\boldsymbol{\tau'}; \boldsymbol{\theta}^-) - Q(\boldsymbol{\tau}, \boldsymbol{a}; \boldsymbol{\theta}) \right)^2 \right],$$

where $V(\boldsymbol{\tau'}; \boldsymbol{\theta}^-) = \max_{\boldsymbol{a'}} Q(\boldsymbol{\tau'}, \boldsymbol{a'}; \boldsymbol{\theta}^-)$ is the TD target, and $\boldsymbol{\theta}^-$ are the parameters of the target network, updated periodically with $\boldsymbol{\theta}$. Replay memory D is used to store transitions $(\boldsymbol{\tau}, \boldsymbol{a}, r, \boldsymbol{\tau'})$.

3 Related Work

3.1 Multi-Agent Reinforcement Learning

CTDE is a widely adopted framework in cooperative MARL [31, 23]. It facilitates coordination through centralized joint value estimation [4], while enabling efficient decentralized execution [40]. Actor-critic methods that leverage a centralized critic to guide decentralized policies, including RIIT [17], have demonstrated notable effectiveness [23, 9, 17]. Variants of PPO [35], including IPPO [5], MAPPO [46], and HAPPO [22], have further improved coordination in complex tasks.

Value-based methods within CTDE have evolved primarily by factorizing global value functions into individual utilities. Early linear approaches (VDN [38]) were succeeded by nonlinear mixing networks (QMIX [32]) and richer Individual-Global-Max (IGM) principles (QTRAN [37]). Further enhancements included attention mechanisms (Qatten [45]), monotonicity constraints on advantage values (QPLEX [43]), and relaxed weighting schemes (Weighted QMIX [30]).

To address partial observability, many approaches have adopted explicit inter-agent communication [40]. However, these methods face limitations in practice, including applicability, unreliability, cost, and scalability concerns, highlighting the necessity for communication-free alternatives.

Despite advances, popular CTDE methods typically rely on local observations and temporal-difference signals, without explicitly modeling the underlying latent environmental states. This often results in suboptimal representations under severe partial observability. Our method addresses this gap by explicitly incorporating recurrent, self-supervised representation learning to robustly infer latent states, significantly enhancing local utility estimations and overall value decomposition.

3.2 Representation Learning in Reinforcement Learning

Representation learning has played a pivotal role in advancing single-agent reinforcement learning, particularly through world models that utilize variational autoencoders (VAEs) and recurrent networks [16] to encode and predict future observations [13]. However, extending these approaches to MARL presents unique challenges. MARL settings exhibit severe partial observability and inherent non-stationarity due to the co-adaptation of agents, making it difficult to infer even the current state.

Recent efforts have sought to adapt representation learning to multi-agent settings. PAC [48] introduces variational encodings for counterfactual prediction, improving value factorization. Grover et al.[12] and Igl et al.[18] propose using Gaussian embeddings and recurrent VAE-based models to manage uncertainty and partial observability. Papoudakis et al. [28] develop inference models that explicitly reason about opponent strategies. COLA [44] learns decentralized representations via contrastive learning, aligning agent trajectories without modeling latent dynamics or using centralized supervision. In contrast, our approach infers a latent global state through self-supervised recurrent modeling, leveraging centralized training for improved performance under partial observability.

A closely related method, MA^{2}E [19], employs a transformer-based masked autoencoder to reconstruct other agents' trajectories from local observations, leveraging self-attention mechanisms [42]. While effective, MA^{2}E requires pretraining on data collected from random policies, which can be sample-inefficient. Its computational complexity scales with both episode length and agent count, and its performance may degrade in settings where agent behaviors are homogeneous.

In contrast, our approach directly infers latent representations of the global state, without reconstructing other agents' trajectories or observations. It does not require pre-training or random policy implementation. Using a recurrent encoder-decoder trained with self-supervised objectives, our method efficiently captures latent state dynamics. Leveraging centralized training, variational inference, and recurrent modeling, we infer compact, fixed-size global representations from local agent histories—scalable across episode lengths and agent counts.

4 Motivation: Two-Agent, Three-Action, Three-State Matrix Game

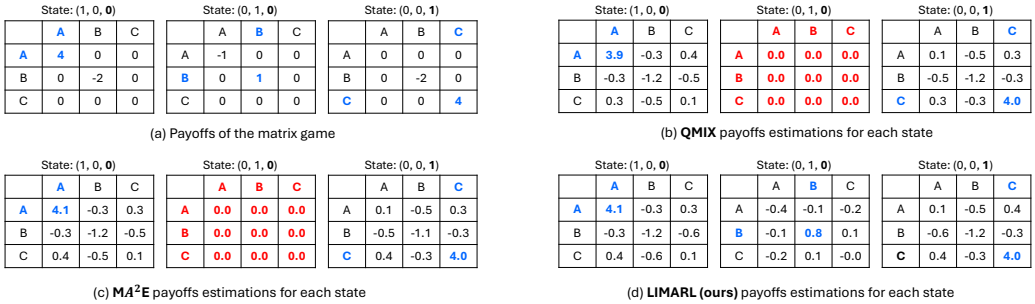


Figure 1: Two-Agent, Three-State, Three-Action Matrix Game with Deterministic Rewards and Transitions. Each table in (a) represents the true payoffs of a state, while the others—(b), (c), and (d)—show the estimations, after 50k steps, of QMIX, MA^{2}E, and LIMARL (ours). Each state is represented by a 3D one-hot vector. The agents observe only the last dimension of the state vector, making the first two states indistinguishable.

We introduce a simple yet insightful matrix game (Figure 1 (a)) to highlight the limitations of existing MARL algorithms—even in settings with deterministic dynamics and a limited number of agents, actions, and states. The game is carefully designed to satisfy the *monotonicity* assumption, where the joint action-value function is a monotonic function of the individual agents' utilities. Moreover, both the state transitions and reward structure are deterministic and independent of the actions. Despite the simplicity of this setup, we observe that several advanced MARL algorithms fail to learn the optimal policy. In contrast, our method consistently learns the optimal policy within just a few thousand iterations, accurately assigning the highest value to the optimal joint action in each state.

The environment models a two-agent setting with three states. Each agent selects from three possible actions per state (A, B, or C), and each episode consists of a fixed horizon of 10 steps. State transitions follow a deterministic cycle: $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_1$. Each state is encoded as a 3D one-hot vector; however, agents observe only the last dimension of this vector, introducing partial observability.

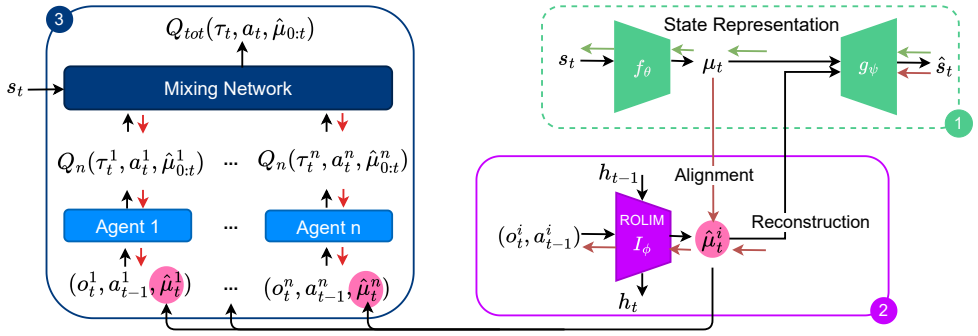


Figure 2: Illustration of the **training process** for LIMARL. At each timestep, given the current local observation o_t^i and the previous action a_{t-1}^i, each agent i uses the ROLIM module (I_ϕ) to infer a latent representation $\hat{\mu}_t^i$. ROLIM is trained by aligning its output with the state-based latent encoding μ_t produced via an encoder f_θ using the full state s_t, and by minimizing reconstruction loss via a decoder g_ψ. Black arrows indicate forward computation paths; colored arrows indicate gradient flows during training.

As a result, both S_1 and S_2 produce the same observation $\mathbf{0}$, while S_3 produces a $\mathbf{1}$. This partial observability renders S_1 and S_2 indistinguishable based on the current observation alone. However, due to the deterministic and cyclic nature of the transitions, agents can infer the states by combining their current and previous observations. For instance, if an agent observes a $\mathbf{1}$ followed by a $\mathbf{0}$, it must currently be in S_1; if it observes two consecutive $\mathbf{0}$s, it must be in S_2.

Each state is associated with a payoff matrix that defines the rewards for all joint actions, as shown in Figure 1 (a). We evaluate a suite of state-of-the-art algorithms, including QMIX [32, 31], QPLEX [43], OW-QMIX and CW-QMIX [30], COLA [44], and MA^{2}E [19]. While many of these methods correctly assign the highest utility in S_1 (A, A) and S_3 (C, C), none of them are able to correctly assign the highest utility to the optimal action (B, B) in S_2. For example, as shown in Figure 1 (b) and (c), the learned value distributions over actions in S_2 are flat, failing to identify the optimal action. In contrast, our approach LIMARL—depicted in Figure 1 (d)—overcomes this by leveraging self-supervised representation learning to infer latent state representations (detailed in Section 5). These representations enable the agents to distinguish between states despite identical instantaneous observations, allowing our method to learn and assign the correct utilities and ultimately recover the optimal policy in each state. Further experimental details and algorithm comparisons are provided in Appendix D.

5 Methodology

LIMARL introduces a novel framework for cooperative MARL under partial observability. LIMARL allows agents to infer compact global-state representations from local observation histories, following the CTDE paradigm. The key idea is to learn compact, latent representations of the global state during centralized training and enable agents to infer these representations from their local trajectories during decentralized execution. As illustrated in Figure 2, LIMARL consists of three main components:

(1) *State Representation Module (SRM):* Learns a compact, structured latent encoding of the global state during centralized training using a variational autoencoder (VAE) (Section 5.1).

(2) *Recurrent Observation-to-Latent Inference Module (ROLIM):* Enables each agent to infer the latent representation of the global state from its local action-observation history during decentralized execution, trained to align with the SRM's encoding (Section 5.2).

(3) *Latent-Augmented Policy:* Uses the inferred latent (from ROLIM) together with local history to compute utility estimates (Section 5.3).

5.1 State Representation Module (SRM)

During centralized training, we assume access to the true global state $s_t \in \mathcal{S}$ at each timestep. The SRM encodes each state into a compact, d-dimensional latent vector using a VAE [20, 21, 33].

Specifically, we define the encoder as: $(\mu_t, \log \sigma_t^2) = f_\theta(s_t)$, where $\mu_t, \sigma_t \in \mathbb{R}^d$ parameterize a diagonal Gaussian distribution $p_\theta(z_t \mid s_t) = \mathcal{N}(z_t; \mu_t, \mathrm{diag}(\sigma_t^2))$. We sample latent variables using the reparameterization trick: $z_t = \mu_t + \sigma_t \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. The decoder $g_\psi : \mathbb{R}^d \to \mathcal{S}$ reconstructs the state: $\hat{s}_t = g_\psi(z_t)$. The SRM is trained to minimize the following VAE objective:

$$\mathcal{L}_{\text{SRM}}(\theta, \psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\underbrace{\|g_\psi(z_t) - s_t\|_2^2}_{\text{Reconstruction Loss}} + \lambda_{\text{KL}} \underbrace{D_{\text{KL}}\left(p_\theta(z_t \mid s_t) \,\|\, \mathcal{N}(0, I)\right)}_{\text{KL Regularization}} \right] \tag{1}$$

where $\mathcal{D}$ is the distribution of visited states, and $\lambda_{\text{KL}} \in \mathbb{R}_+$ balances reconstruction and regularization.

We train a single SRM shared across all agents. During centralized training, joint rollouts provide states s_t, and the encoder-decoder pair is optimized to minimize the loss in Equation (1).

5.2 Recurrent Observation-to-Latent Inference Module (ROLIM)

To enable agents to infer latent representations $\hat{z}_t^i \in \mathbb{R}^d$ of the global state during decentralized execution, we introduce the ROLIM. Each agent i maintains a hidden state h_t^i based on its local observation-action trajectory $\tau_t^i = (o_0^i, a_0^i, \ldots, o_{t-1}^i, a_{t-1}^i, o_t^i)$, and infers:

$$(\hat{\mu}_t^i, \log \hat{\sigma}_t^{i2}, h_t^i) = I_\phi([o_t^i, a_{t-1}^i], h_{t-1}^i),$$

where $\hat{z}_t^i \sim \hat{q}_\phi(. \mid \tau_t^i) = \mathcal{N}(\hat{\mu}_t^i, \mathrm{diag}(\hat{\sigma}_t^{i2}))$, and sampled via: $\hat{z}_t^i = \hat{\mu}_t^i + \hat{\sigma}_t^i \odot \epsilon, \epsilon \sim \mathcal{N}(0, I)$.

The ROLIM objective aligns these inferred latents with the SRM's reference latents and promotes semantic structure. Specifically, the loss is:

$$\mathcal{L}_{\text{ROLIM}}(\phi) = \mathbb{E}_{(s_t, \tau_t^i) \sim \mathcal{D}} \left[\lambda_{\text{align}} \underbrace{\|\hat{\mu}_t^i - \mu_t\|_2^2}_{\text{Alignment to SRM}} + \lambda_{\text{rec}} \underbrace{\|g_\psi(\hat{z}_t^i) - s_t\|_2^2}_{\text{State Reconstruction}} + \lambda_{\text{KL}} \underbrace{D_{\text{KL}}(\hat{q}_\phi(\hat{z}_t^i \mid \tau_t^i) \,\|\, \mathcal{N}(0, I))}_{\text{Latent Regularization}} \right] \tag{2}$$

where $(\mu_t, .) = f_\theta(s_t)$ is the latent mean from the SRM. The decoder g_ψ is frozen to ensure consistency with the SRM latent space, and $\lambda_{\text{align}}, \lambda_{\text{rec}}, \lambda_{\text{KL}} \in \mathbb{R}_+$ are hyperparameters controlling the trade-offs between accuracy, alignment, and regularization. This objective ensures that the inferred latent vectors approximate their SRM counterparts while remaining semantically meaningful.

During ROLIM training, the encoder f_θ and decoder g_ψ are frozen to preserve consistency in the SRM latent space. The inference module I_ϕ is shared across all agents, promoting parameter efficiency and robustness. This design leverages multiple agents' diverse observations to improve latent inference.

5.3 LIMARL: Latent-Inference for MARL

5.3.1 Centralized Training

The training of LIMARL consists of two interleaved phases: (1) learning latent representations via SRM and ROLIM, and (2) policy learning using value-based MARL with latent augmentation. This modular decomposition improves training stability and facilitates efficient credit assignment.

Phase I: Learning Latent Representations. We first train the SRM to encode each global state s_t into a compact latent vector z_t. The SRM is implemented as a VAE with encoder-decoder pair (f_θ, g_ψ) trained using the objective in Equation (1), which balances reconstruction accuracy and KL regularization. This step provides a latent space, serving as a reference for subsequent inference.

Next, we train the ROLIM to infer latent estimates $\hat{\mu}_t^i$ from each agent's local trajectory τ_t^i. ROLIM minimizes the loss in Equation (2), aligning its outputs with the SRM's latent codes. This alignment ensures that decentralized agents can approximate global state information without explicit access. The decoder g_ψ is frozen during this step to preserve a consistent latent reference.

To minimize redundant updates, we track the mean ROLIM loss $\bar{L}_{\text{ROLIM}}$ on a dynamic validation set and continuously monitor the online inference loss over a sliding buffer of recent episodes. When the instantaneous loss exceeds $\bar{L}_{\text{ROLIM}}$ by more than a predefined percentage $\delta \in [0, 1]$, we retrain both the SRM and ROLIM modules from their most recent best checkpoint. This conditional update strategy ensures that retraining is only triggered when the online performance degrades significantly relative to the validation baseline, thereby avoiding unnecessary fine-tuning and improving overall training stability. An ablation study in Appendix F analyzes the effect of varying δ.

Phase II: MARL with Inferred Latent Augmentation. Once SRM and ROLIM are trained, we freeze their parameters and proceed with policy learning. At each timestep, agent i computes its latent estimate $\hat{\mu}_t^i$, as function of its local action-observation history τ_t^i, using ROLIM I_ϕ. The local utility function is conditioned on both the trajectory and the latent: $Q_i(a_t^i \mid \tau_t^i, \hat{\mu}_t^i)$. These individual utilities are combined using a monotonic mixing network Mix_ω to yield the global action-value: $Q_{\text{tot}}(\tau_t, \hat{\mu}_{0:t}, a_t) = \text{Mix}_\omega(Q_1, \ldots, Q_n)$ where $\hat{\mu}_{0:t}$ denotes the sequence of inferred latents.

Standard temporal-difference learning is used to train Q_i and Mix_ω, with frozen SRM and ROLIM modules. Replay buffers store full trajectories to support both value updates and latent inference.

5.3.2 Decentralized Execution

At execution time, agents no longer access the state or the SRM encoder-decoder. Each agent maintains its own trajectory τ_t^i and uses ROLIM to infer its latent $\hat{\mu}_t^i$. The agent then selects actions via $a_t^i = \arg\max_{a \in \mathcal{A}} Q_i(a \mid \tau_t^i, \hat{\mu}_t^i)$. This enables decentralized decision-making under partial observability, without requiring communication or synchronization.

6 Theoretical Analysis

We provide theoretical foundations for LIMARL, establishing how latent representations support MARL under partial observability in the CTDE framework. Our analysis yields three key results: (1) Latent policies can recover optimal actions despite imperfect reconstruction (Theorem 6.4), (2) Decentralized greedy policies over latent utilities can achieve globally optimal coordination (Theorem 6.5), (3) Utility estimates remain stable even under inference errors at execution time (Theorem 6.6). All proofs are provided in Appendix C.

6.1 Latent-State Sufficiency for Control

We begin by asking: *Can latent representations retain all necessary information for optimal decision-making?* The following result shows that if the latent space perfectly captures the full state, then value functions can be equivalently expressed in the latent space without loss.

Lemma 6.1 (Latent Sufficiency for Optimal Utility)**.** *Suppose $g_\psi(f_\theta(s)) = s$ for all $s \in \mathcal{S}$. Then, for each agent i, there exists a function $\tilde{Q}_i^\star : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}$ such that*

$$Q_i^\star(s, a) = \tilde{Q}_i^\star(f_\theta(s), a), \quad \forall s \in \mathcal{S},\ a \in \mathcal{A}.$$

Perfect reconstruction is idealized; real-world models only approximate the true state. To study the effect of the reconstruction error on policy quality, we assume the following regularity condition [15, 8]:

Assumption 6.2 (Lipschitz Continuity of the Optimal Utility)**.** The ground-truth utility function $Q_i^\star(s, a)$ is L-Lipschitz in s, i.e., for all $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$,

$$|Q_i^\star(s, a) - Q_i^\star(s', a)| \leq L \cdot \|s - s'\|_2.$$

Lemma 6.3 (Approximate Latent Sufficiency)**.** *Let Assumption 6.2 hold, the reconstruction error at state $s \in \mathcal{S}$ be: $\varepsilon_{\text{rec}}(s) = \|g_\psi(f_\theta(s)) - s\|_2$, and define the latent utility as $\tilde{Q}_i^\star(\mu, a) := Q_i^\star(g_\psi(\mu), a)$. Then, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$,*

$$\left| Q_i^\star(s, a) - \tilde{Q}_i^\star(f_\theta(s), a) \right| \leq L \cdot \varepsilon_{\text{rec}}(s).$$

We now ask: *Can we still select optimal actions from the latent space even when reconstruction isn't perfect?*

Theorem 6.4 (Argmax Consistency Under Imperfect Reconstruction)**.** *Let $a^\star = \arg\max_{a \in \mathcal{A}} Q_i^\star(s, a)$, and let $\tilde{a} = \arg\max_{a \in \mathcal{A}} \tilde{Q}_i^\star(f_\theta(s), a)$. Define the advantage margin*

$$\Delta(s) := Q_i^\star(s, a^\star) - \max_{a \neq a^\star} Q_i^\star(s, a).$$

If $\Delta(s) > 2L \cdot \varepsilon_{\text{rec}}(s)$, then $\tilde{a} = a^\star$; that is, the latent-greedy policy matches the state-optimal policy.

Interpretation: This result provides a margin-based guarantee that optimal decisions can still be made from imperfect latent representations, as long as reconstruction errors are sufficiently small. This establishes the robustness of LIMARL's latent-space control in practice.

6.2 Decentralized Optimality under Monotonicity

We now turn to coordination: *Can decentralized agents acting on local latent estimates recover the globally optimal joint action?*

Theorem 6.5 (Decentralized Greedy Optimality [31])**.** *Under the monotonic mixing assumption, the decentralized greedy policy* $a_t^{(i)*} = \arg\max_{a \in \mathcal{A}} Q_i\big(\tau_t^i, \hat{\mu}_{0:t}^i, a\big), \quad \forall i$*, recovers the globally optimal joint action:* $\boldsymbol{a}_t^* = \arg\max_{\boldsymbol{a}} Q_{tot}\big(\boldsymbol{\tau}_t, \hat{\boldsymbol{\mu}}_{0:t}, \boldsymbol{a}\big)$.

Implication: Even without communication, agents can act greedily over their local trajectories and inferred latents to achieve optimal joint coordination. This result is critical for LIMARL's scalability and decentralized execution under partial observability.

6.3 Robustness to Latent Inference Errors

Finally, we consider the realistic case where latents are inferred from partial observations. The question is: *How sensitive are utility estimates to inference inaccuracies?*

The following result quantifies how inference errors at test time impact utility estimation.

Theorem 6.6 (Bounded Utility Error under Latent Misalignment)**.** *Let Assumption 6.2 hold, the reconstruction error at state* $s \in \mathcal{S}$ *be:* $\varepsilon_{\text{rec}}(s) = \|g_\psi(f_\theta(s)) - s\|_2$*, each agent* i *infer its latent via* $\hat{\mu}_t^i = I_\phi(\tau_t^i)$*, and define the alignment error as:* $\varepsilon_{\text{align}}^i(\mu) = \left\|\hat{\mu}^i - \mu\right\|_2$*, and assume* g_ψ *is* L_ψ*-Lipschitz. Then for any action* $a \in \mathcal{A}$*,*

$$\frac{1}{L}\left|Q_i^\star(s_t, a) - Q_i^\star\left(g_\psi(\hat{\mu}_t^i), a\right)\right| \leq L_\psi \varepsilon_{\text{align}}^i(\mu_t) + \varepsilon_{\text{rec}}(s_t).$$

Therefore, as long as inferred latents remain well-aligned and reconstruction is accurate, the impact on decision quality is provably bounded. These theoretical guarantees underpin LIMARL's design, where training objectives explicitly minimize both alignment and reconstruction error.

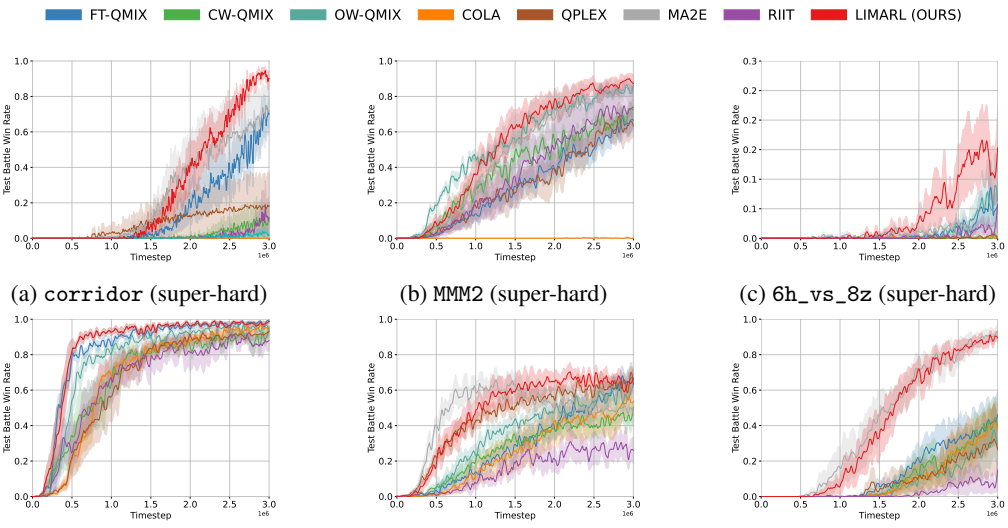
7 Experiments

We empirically evaluate LIMARL across three complementary settings. First, we revisit the matrix coordination game (introduced in Figure 1), where all considered baselines fail to learn the optimal policy, while LIMARL consistently converges to it. Second, we benchmark LIMARL on the StarCraft Multi-Agent Challenge (SMAC) [34] and its more recent variant SMACv2 [7], both of which pose significant coordination and observability challenges. Third, to further isolate the impact of the ROLIM module, we design a partially observable classification game that enables visual inspection and quantitative evaluation of latent reconstructions. Complete experimental specifications, hyperparameters, and implementation details are deferred to Appendix B.

Setup. We evaluate LIMARL against several state-of-the-art cooperative MARL baselines across environments with varying levels of partial observability. Specifically, we compare against Fine-tuned QMIX [32, 31, 17]; QPLEX [43], OW-QMIX [30], and CW-QMIX [30]. Furthermore we compare against RIIT [17] (policy-based); COLA [44]; as well as against MA^{2}E [19]. All methods are evaluated on 6 challenging scenarios from the SMAC benchmark [34], which differ in team composition, observability, and coordination complexity. In addition, we test on SMACv2 [7], a more challenging benchmark that introduces environment shifts across three different unit types: `Protoss`, , and `Zerg`. All methods are trained for 3 million environment steps, and evaluated using the average win rate across 6 random seeds.

7.1 Performance on SMAC and SMACv2 Benchmarks.

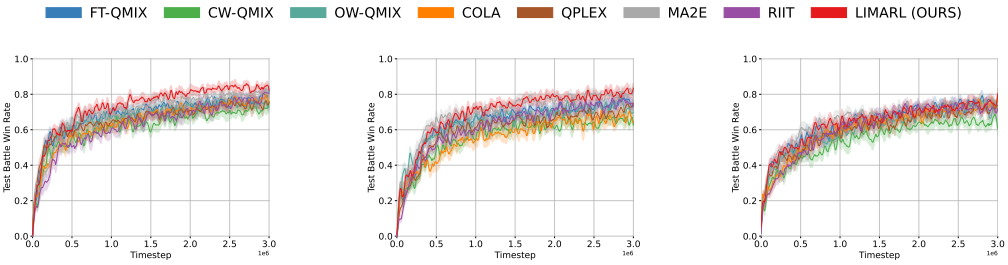
Figures 3 and 4 show the win rate in each scenario over time steps. LIMARL consistently outperforms all baselines across the considered scenarios, with the most pronounced gains in partially observable, long-horizon tasks such as `corridor` and `6h_vs_8z` (both super-hard), validating the efficacy of ROLIM under decentralized execution. Furthermore, we evaluated the statistical significance of LIMARL's improvements over FT-QMIX using Wilcoxon signed-rank tests across 6 seeds. Significant gains were observed on corridor (p = 0.03) and 3s_vs_5z (p = 0.03), while MMM2 (p = 0.07) and 6h_vs_8z (p = 0.08) showed marginal significance.



(a) `corridor` (super-hard) (b) `MMM2` (super-hard) (c) `6h_vs_8z` (super-hard)

(d) `2c_vs_64zg` (hard) (e) `5m_vs_6m` (hard) (f) `3s_vs_5z` (hard)

Figure 3: Training performance (win rate %) of LIMARL vs. baselines across SMAC scenarios. LIMARL demonstrates consistently faster convergence and higher final performance.



(a) `Terran` (b) `Protoss` (c) `Zerg`

Figure 4: Training performance (win rate %) of LIMARL vs. baselines across SMACv2 scenarios. LIMARL demonstrates consistently faster convergence and higher final performance.

7.2 Visualization and Ablation: Inference in a Partially Observable Classification Game

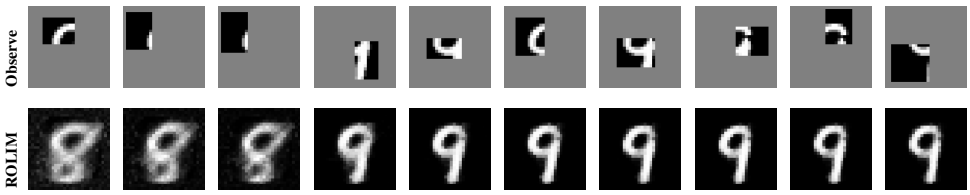


Figure 5: **Qualitative reconstructions.** Top: partial inputs (**Observe**) available to agents. Bottom: reconstructions from **ROLIM**, conditioned on the same observation sequence.

To isolate the contribution of each training objective, we evaluate our latent inference module in a controlled multi-agent environment: the *MNIST classification game*. In this setting, agents receive only partial observations of an MNIST digit, as illustrated in Figure 5 (top row), which depicts a sequence of masked inputs from a fixed episode with a consistent underlying state (digit "9"). The goal is to classify digits under partial observability.

We show the ability of ROLIM to reconstruct the true state with just partial observation, as illustrated in Figure 5 (bottom row). Morover we ablate two core components of the training loss: (i) **alignment loss**, and (ii) **regularization**. Results demonstrate that alignment is important, while a moderate KL penalty further improves alignment without causing degeneration. Full qualitative and quantitative results across inference architectures and latent visualizations via t-SNE are provided in Appendix E.

7.3 Ablation Study on SMAC

Ablation studies on key hyperparameters, including the intrinsic dimensionality d and the fine-tuning threshold δ, are provided in Appendix F and G. These results show that the approach is relatively robust to variations in these parameters.

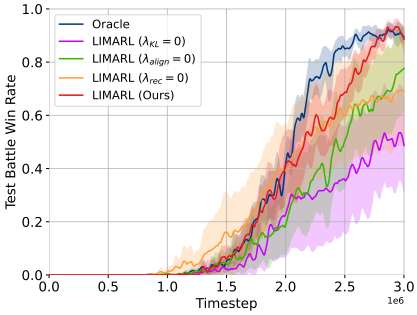


Figure 6: Ablation results on the SMAC `corridor` (super-hard) scenario.

Besides, to assess the contribution of each component to LIMARL, we evaluate different variants of our method, removing different terms from the latent inference objective (Equation (2)): **(1) LIMARL w/o Grounding** — removes the reconstruction loss term. **(2) LIMARL w/o Alignment** — disables the alignment loss term, which encourages the inferred latent to approximate the ground-truth encoding; **(3) LIMARL w/o KL** — omits the KL divergence term, allowing the latent space to evolve without a distributional prior. Furthermore, we consider **(4) Oracle** - where agents are provided with the true latent state instead of using the inferred one. Each ablation is evaluated under the same experimental settings as the full LIMARL method, using the SMAC benchmark with identical hyperparameters.

As shown in Figure 6, removing any single component of the latent inference objective leads to a measurable drop in performance. This highlights the complementary roles of inference alignment, semantic grounding, and latent regularization. Notably, the combination of the three components significantly approaches that of the Oracle baseline-without having access to the true latent state.

8 Conclusion

We introduced LIMARL, a novel framework for MARL under partial observability that combines self-supervised state representation learning with recurrent latent inference. LIMARL uses centralized training to learn compact global-state embeddings and enables decentralized agents to infer them from local observation histories. Our theoretical analysis supports the design with justifications and guarantees. Extensive experiments on diagnostic tasks and SMAC/SMACv2 scenarios show that LIMARL outperforms state-of-the-art baselines, demonstrating its effectiveness in cooperative, partially observable settings.

References

[1] Abubakr O Al-Abbasi, Arnob Ghosh, and Vaneet Aggarwal. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4714–4727, 2019.

[2] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024.

[3] Chang-Lin Chen, Hanhan Zhou, Jiayu Chen, Mohammad Pedramfar, Tian Lan, Zheqing Zhu, Chi Zhou, Pol Mauri Ruiz, Neeraj Kumar, Hongbo Dong, et al. Learning-based two-tiered online optimization of region-wide datacenter resource allocation. *IEEE Transactions on Network and Service Management*, 2024.

[4] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

[5] Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[6] Joris Dinneweth, Abderrahmane Boubezoul, René Mandiau, and Stéphane Espié. Multi-agent reinforcement learning for autonomous vehicles: A survey. *Autonomous Intelligent Systems*, 2(1):27, 2022.

[7] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:37567–37593, 2023.

[8] Mathieu Fehr, Olivier Buffet, Vincent Thomas, and Jilles Dibangoye. rho-pomdps have lipschitz-continuous epsilon-optimal value functions. *Advances in neural information processing systems*, 31, 2018.

[9] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[10] Fares Fourati, Vaneet Aggarwal, and Mohamed-Slim Alouini. Stochastic q-learning for large discrete action spaces. In *International Conference on Machine Learning*, pages 13734–13759. PMLR, 2024.

[11] Nan Geng, Tian Lan, Vaneet Aggarwal, Yuan Yang, and Mingwei Xu. A multi-agent reinforcement learning perspective on distributed traffic engineering. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE, 2020.

[12] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *International conference on machine learning*, pages 1802–1811. PMLR, 2018.

[13] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[14] Marina Haliem, Ganapathy Mani, Vaneet Aggarwal, and Bharat Bhargava. A distributed model-free ride-sharing approach for joint matching, pricing, and dispatching using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7931–7942, 2021.

[15] Karl Hinderer. Lipschitz continuity of value functions in markovian decision processes. *Mathematical Methods of Operations Research*, 62(1):3–22, 2005.

[16] S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[17] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2102.03479*, 2021.

[18] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International conference on machine learning*, pages 2117–2126. PMLR, 2018.

[19] Sehyeok Kang, Yongsik Lee, Gahee Kim, Song Chong, and Se-Young Yun. MA^{2}e: Addressing partial observability in multi-agent reinforcement learning with masked auto-encoder. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[21] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[22] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251*, 2021.

[23] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[24] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *Advances in neural information processing systems*, 32, 2019.

[25] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[26] Navid Naderializadeh, Jaroslaw J Sydir, Meryem Simsek, and Hosein Nikopour. Resource management in wireless networks via multi-agent deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 20(6):3507–3523, 2021.

[27] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[28] Georgios Papoudakis and Stefano V Albrecht. Variational autoencoders for opponent modeling in multi-agent systems. *arXiv preprint arXiv:2001.10829*, 2020.

[29] Ashley Peake, Joe McCalmon, Benjamin Raiford, Tongtong Liu, and Sarra Alqahtani. Multi-agent reinforcement learning for cooperative adaptive cruise control. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 15–22. IEEE, 2020.

[30] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020.

[31] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[32] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

[33] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[34] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[36] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[37] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019.

[38] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2085–2087, 2018.

[39] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. 2018.

[40] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[41] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[43] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

[44] Zhiwei Xu, Bin Zhang, Dapeng Li, Zeren Zhang, Guangchong Zhou, Hao Chen, and Guoliang Fan. Consensus learning for cooperative multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11726–11734, 2023.

[45] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020.

[46] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[47] Chao Yu, Xinyi Yang, Jiaxuan Gao, Jiayu Chen, Yunfei Li, Jijia Liu, Yunfei Xiang, Ruixin Huang, Huazhong Yang, Yi Wu, et al. Asynchronous multi-agent reinforcement learning for efficient real-time multi-robot cooperative exploration. *arXiv preprint arXiv:2301.03398*, 2023.

[48] Hanhan Zhou, Tian Lan, and Vaneet Aggarwal. Pac: Assisted value factorization with counterfactual predictions in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:15757–15769, 2022.

[49] Wei Zhou, Dong Chen, Jun Yan, Zhaojian Li, Huilin Yin, and Wanchen Ge. Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic. *Autonomous Intelligent Systems*, 2(1):5, 2022.

A Table of Notation

Symbol	Description
$\mathcal{N}$	Number of agents
$\mathcal{S}$	Set of global environment states
$\mathcal{A}$	Action space for each agent
Ω	Observation space for each agent
$P(s' \mid s, a)$	Transition probability from s to s' given joint action a
$O(o^i \mid s, a^i)$	Observation distribution for agent i
$r(s, a)$	Shared reward function for joint action a in state s
γ	Discount factor
τ_t^i	Action-observation history for agent i up to time t
$\pi^i(a \mid \tau^i)$	Policy of agent i based on its history
$Q^i(\cdot)$	Local utility function for agent i
$Q_{\text{tot}}(\cdot)$	Global joint action-value function
$f_\theta(\cdot)$	Encoder mapping global state to latent embedding (SRM)
$g_\psi(\cdot)$	Decoder mapping latent to reconstructed state
$I_\phi(\cdot)$	ROLIM inference module for estimating latent from local history
z_t	Latent representation of global state at time t
$\hat{z}_t^i$	Inferred latent estimate by agent i at time t
μ_t	Mean of latent distribution from SRM at time t
$\hat{\mu}_t^i$	Inferred latent mean for agent i at time t with ROLIM
σ_t^2	Variance of latent distribution from SRM
$\hat{\sigma}_t^{i2}$	Inferred latent variance for agent i from ROLIM
h_t^i	Hidden state of agent i's RNN in ROLIM
$\mathcal{L}_{\text{SRM}}$	SRM loss (reconstruction + KL divergence)
$\mathcal{L}_{\text{ROLIM}}$	ROLIM loss (alignment + reconstruction + KL)
λ_{KL}	KL regularization weight
λ_{align}	Alignment loss weight
λ_{rec}	Reconstruction loss weight
ϵ	Gaussian noise sample in latent reparameterization
$\text{Mix}_\omega(\cdot)$	Monotonic mixing network with parameters ω
D	Replay buffer / dataset distribution
$V(\cdot)$	Value function (TD target)
θ^-	Parameters of the target Q-network

Table 1: Notation used throughout the paper.

B Experimental Details

B.1 Environments

We evaluate LIMARL on both controlled synthetic environments and standard multi-agent benchmarks:

- **Matrix Game (Figure 1):** A two-agent, three-state deterministic game designed to highlight the limitations of existing MARL methods under partial observability.
- **MNIST Classification Game (Section Experiments):** A diagnostic task where agents receive independent masked views of a digit and must classify the underlying image based on their local trajectories.
- **SMAC** [34]: The StarCraft Multi-Agent Challenge provides a suite of decentralized micromanagement tasks based on the StarCraft II game engine. We use the implementation available at `https://github.com/oxwhirl/smac`, which is released under the MIT license.
- **SMACv2** [7]: An updated and more challenging version of SMAC that addresses some of the limitations of the original benchmark. We adopt the implementation from `https://github.com/oxwhirl/smacv2`, also under the MIT license.

B.1.1 SMAC Scenarios

The StarCraft Multi-Agent Challenge (SMAC) is a widely adopted benchmark for evaluating MARL algorithms. In this benchmark, agents control units from the real-time strategy game StarCraft II to cooperate and defeat opposing units. The environment offers a variety of scenarios, each categorized into difficulty levels such as `Easy`, `Hard`, and `Super Hard`, depending on the complexity of coordination required. In our study, we focus on the `Hard` and `Super Hard` settings, which pose significant challenges due to unit imbalance, coordination demands, or heterogeneous agent types. Table 2 provides an overview of the scenarios considered in our experiments.

Table 2: Detailed description of the SMAC scenarios used in our experiments.

Scenario	Difficulty	Ally Units	Enemy Units	Type
3s_vs_5z	Hard	3 Stalkers	5 Zealots	Micro-trick: kiting
2c_vs_64zg	Hard	2 Colossi	64 Zerglings	Micro-trick: positioning
MMM	Hard	1 Medivac 2 Marauders 7 Marines	1 Medivac 2 Marauders 7 Marines	Heterogeneous & symmetric
corridor	Super Hard	6 Zealots	24 Zerglings	Micro-trick: wall off
6h_vs_8z	Super Hard	6 Hydras	8 Zealots	Micro-trick: focus fire
MMM2	Super Hard	1 Medivac 2 Marauders 7 Marines	1 Medivac 3 Marauders 8 Marines	Heterogeneous & asymmetric

B.1.2 SMACv2

SMACv2 was introduced to address key limitations in the original SMAC benchmark, particularly its limited stochasticity and overly simplified partial observability [7]. It introduces three core enhancements to better reflect the challenges of real-world multi-agent coordination:

First, unit compositions are no longer fixed. Instead, in SMACv2, unit types are sampled probabilistically for each episode, introducing diversity in team configurations (unit generation is detailed in Table 3). Second, agent observability is made more realistic. Unlike SMAC—where if one agent spots an enemy, others within range are guaranteed to do the same—SMACv2 introduces uncertainty: agents within observation range may perceive different local views, even when observing the same enemy. Finally, SMACv2 randomizes spawn locations using two spatial initialization strategies: surround, where allied units are positioned encircling enemies, and reflect, where opposing teams are arranged in mirrored, head-on formations. These changes collectively make SMACv2 a more

challenging and stochastic environment for MARL research. In our experiments we focused on the surround setting as it offers less information about the full state making it more challenging in partially observable settings.

Race	Unit	Generation Probability
Terran	Marine	0.45
	Marauder	0.45
	Medivac	0.10
Protoss	Stalker	0.45
	Zealot	0.45
	Colossus	0.10
Zerg	Zergling	0.45
	Hydralisk	0.45
	Baneling	0.10

Table 3: Detailed generation probabilities of the three types of units for the three races (Protoss, Terran, and Zerg).

B.2 Training and Evaluation Protocol

- **Training steps:** For SMAC all models are trained for at least 3 million environment steps.
- **Evaluation:** We evaluate every 10,000 steps using 32 test episodes without exploration noise and report average win rates over 6 random seeds.
- **Episode limits:** Follow default SMAC time limits (e.g., 120–200 steps per episode depending on the scenario).
- **Computing resources:** All experiments were run on NVIDIA A100 GPUs with 40 GB memory.

B.3 Hyperparameters

Shared across all models:

- Replay buffer size: 5000 episodes
- Batch size: 128
- Optimizer: Adam, learning rate 1×10^{-3}
- Discount factor $\gamma = 0.99$
- Target network update interval: every 200 episodes
- Exploration: ϵ-greedy with linear decay from 1.0 to 0.05 over 100k steps

LIMARL-specific:

- Latent dimension $d = 64$
- Encoder/decoder: 2-layer MLP with ReLU activations
- Inference module: 1-layer GRU with hidden size 128
- KL regularization weight $\lambda_{\text{KL}} = \min\left(1.0, \frac{\text{steps}}{\text{KL}_{\text{warmup}}}\right)$
- Alignment loss weight $\lambda_{\text{align}} = 1.0$
- Reconstruction loss weight $\lambda_{\text{rec}} = 1.0$

B.4 Baselines

We compare against the following baselines, reimplemented or adapted from open-source libraries:

- **FT-QMIX** [32, 31, 17]
- **QPLEX** [43]

- **OW-QMIX** and **CW-QMIX** [30]
- **COLA** [44]
- **RIIT** [17]
- **MA^{2}E** [19]

All baselines use their recommended hyperparameters and are trained under the same CTDE setup and training budget for fair comparison.

B.5 Implementation and Reproducibility

Our code is implemented in PyTorch and builds on the open-source PyMARL2 framework [17].

C Missing Proofs

C.1 Proof of Lemma 6.1

Proof. Define $\tilde{Q}_i^\star(x, a) := Q_i^\star(g_\psi(x), a)$. Then for any $s \in \mathcal{S}$, let $(\mu, .) = f_\theta(s)$

$$\tilde{Q}_i^\star(\mu, a) = Q_i^\star(g_\psi(\mu), a) = Q_i^\star(s, a),$$

by the assumption that $g_\psi(\mu) = s$. Hence, the claim holds. □

C.2 Proof of Lemma 6.3

Proof. Let $s' = g_\psi(f_\theta(s))$. By the Lipschitz property of $Q_i^\star$,

$$|Q_i^\star(s, a) - Q_i^\star(s', a)| \leq L \cdot \|s - s'\|_2 = L \cdot \varepsilon_{\text{rec}}(s).$$

□

C.3 Proof of Theorem 6.4

Proof. By Lemma 6.3, for all $a \in \mathcal{A}$,

$$|Q_i^\star(s, a) - \tilde{Q}_i^\star(f_\theta(s), a)| \leq L \cdot \varepsilon_{\text{rec}}(s).$$

Because $\tilde{a}$ maximizes $\tilde{Q}_i^\star$, we have

$$\tilde{Q}_i^\star(f_\theta(s), \tilde{a}) \geq \tilde{Q}_i^\star(f_\theta(s), a^\star).$$

Combining these two facts,

$$Q_i^\star(s, a^\star) - Q_i^\star(s, \tilde{a}) \leq 2L \cdot \varepsilon_{\text{rec}}(s).$$

If $\Delta(s) > 2L \cdot \varepsilon_{\text{rec}}(s)$, this implies $\tilde{a} = a^\star$. □

C.4 Proof of Theorem 6.5

We follow the argument from [31] and include it here for completeness. Assume that the total value function $Q_{\text{tot}}\big(Q_1, \ldots, Q_n\big)$ is monotonic in each of its inputs:

$$\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0, \quad \forall i \in \{1, \ldots, n\}.$$

Then, for any joint action $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^n)$, we can iteratively improve the joint value by replacing each agent's action with its local maximizer:

$$\begin{aligned} &Q_{\text{tot}}\big(Q_1(\tau_t^1, \mu_{0:t}^1, a_t^1), \ldots, Q_n(\tau_t^n, \mu_{0:t}^n, a_t^n)\big) \\ &\quad \leq Q_{\text{tot}}\big(\max_{a^1} Q_1(\tau_t^1, \mu_{0:t}^1, a^1), \ldots, Q_n(\tau_t^n, \mu_{0:t}^n, a_t^n)\big) \\ &\quad \leq \cdots \leq Q_{\text{tot}}\big(\max_{a^1} Q_1(\tau_t^1, \mu_{0:t}^1, a^1), \ldots, \max_{a^n} Q_n(\tau_t^n, \mu_{0:t}^n, a^n)\big). \end{aligned}$$

Define each agent's greedy local action as:

$$a_t^{(i)*} = \arg\max_{a \in \mathcal{A}} Q_i(\tau_t^i, \mu_{0:t}^i, a),$$

and let the joint greedy action be $\boldsymbol{a}_t^* = (a_t^{(1)*}, \ldots, a_t^{(n)*})$. Then:

$$\begin{aligned} &Q_{\text{tot}}\big(Q_1(\tau_t^1, \mu_{0:t}^1, a_t^{(1)*}), \ldots, Q_n(\tau_t^n, \mu_{0:t}^n, a_t^{(n)*})\big) \\ &= Q_{\text{tot}}\big(\max_{a^1} Q_1(\tau_t^1, \mu_{0:t}^1, a^1), \ldots, \max_{a^n} Q_n(\tau_t^n, \mu_{0:t}^n, a^n)\big) \\ &= \max_{\boldsymbol{a}} Q_{\text{tot}}(\boldsymbol{\tau}_t, \boldsymbol{\mu}_{0:t}, \boldsymbol{a}). \end{aligned}$$

Therefore, the joint action composed of individually greedy actions is globally optimal:

$$\boldsymbol{a}_t^* = \arg\max_{\boldsymbol{a}} Q_{\text{tot}}(\boldsymbol{\tau}_t, \boldsymbol{\mu}_{0:t}, \boldsymbol{a}).$$

□

C.5 Proof of Theorem 6.6

We can decompose the error as:

$$\left|Q_i^\star(s_t, a) - Q_i^\star(g_\psi(\hat{\mu}_t^i), a)\right| \leq |Q_i^\star(s_t, a) - Q_i^\star(g_\psi(\mu_t), a)| + \left|Q_i^\star(g_\psi(\mu_t), a) - Q_i^\star(g_\psi(\hat{\mu}_t^i), a)\right|.$$

Under Assumption 6.2, where $Q_i^\star$ is L-Lipschitz in its first argument, we obtain:

$$|Q_i^\star(s_t, a) - Q_i^\star(g_\psi(\mu_t), a)| \leq L \cdot \|s_t - g_\psi(\mu_t)\|_2.$$

Again using Assumption 6.2, we have:

$$\left|Q_i^\star(g_\psi(\mu_t), a) - Q_i^\star(g_\psi(\hat{\mu}_t^i), a)\right| \leq L \cdot \|g_\psi(\mu_t) - g_\psi(\hat{\mu}_t^i)\|_2.$$

Assuming the decoder g_ψ is L_ψ-Lipschitz, we further bound:

$$\|g_\psi(\mu_t) - g_\psi(\hat{\mu}_t^i)\|_2 \leq L_\psi \cdot \|\mu_t - \hat{\mu}_t^i\|_2.$$

Combining the two terms, we conclude:

$$\left|Q_i^\star(s_t, a) - Q_i^\star(g_\psi(\hat{\mu}_t^i), a)\right| \leq LL_\psi \cdot \|\mu_t - \hat{\mu}_t^i\|_2 + L \cdot \|g_\psi(\mu_t) - s_t\|_2.$$

□

D Extended Results: Matrix Game Performance Across Algorithms

This section presents extended results and analysis for the matrix game described in Section 4, offering a detailed comparison across several state-of-the-art MARL algorithms.

State: (1, 0, **0**)

	A	B	C
A	4	0	0
B	0	-2	0
C	0	0	0

State: (0, 1, **0**)

	A	B	C
A	-1	0	0
B	0	1	0
C	0	0	0

State: (0, 0, **1**)

	A	B	C
A	0	0	0
B	0	-2	0
C	0	0	4

(a) Payoffs of the matrix game

State: (1, 0, **0**)

	A	B	C
A	3.9	-0.3	0.4
B	-0.3	-1.2	-0.5
C	0.3	-0.5	0.1

State: (0, 1, **0**)

	A	B	C
A	0.0	0.0	0.0
B	0.0	0.0	0.0
C	0.0	0.0	0.0

State: (0, 0, **1**)

	A	B	C
A	0.1	-0.5	0.3
B	-0.5	-1.2	-0.3
C	0.3	-0.3	4.0

(b) **QMIX** payoffs estimations for each state

State: (1, 0, **0**)

	A	B	C
A	4.0	0.0	0.1
B	0.0	-0.1	-0.1
C	0.1	-0.1	-0.1

State: (0, 1, **0**)

	A	B	C
A	0.2	0.2	0.2
B	0.2	0.2	0.2
C	0.2	0.2	0.2

State: (0, 0, **1**)

	A	B	C
A	-0.0	-0.1	0.2
B	-0.1	-0.2	0.0
C	0.2	0.0	4.0

(c) **OW-QMIX** payoffs estimations for each state

State: (1, 0, **0**)

	A	B	C
A	4.0	-0.2	-0.2
B	-0.2	-0.8	-0.6
C	0.2	-0.6	-0.3

State: (0, 1, **0**)

	A	B	C
A	-0.1	-0.1	-0.1
B	-0.1	-0.1	-0.1
C	-0.1	-0.1	-0.1

State: (0, 0, **1**)

	A	B	C
A	-0.3	-0.6	0.2
B	-0.6	-0.9	-0.2
C	0.2	-0.1	4.0

(d) **CW-QMIX** payoffs estimations for each state

State: (1, 0, **0**)

	A	B	C
A	4.0	1.6	2.0
B	1.7	-0.8	-0.3
C	1.9	-0.5	-0.1

State: (0, 1, **0**)

	A	B	C
A	-0.0	-0.0	-0.0
B	-0.0	-0.0	-0.0
C	-0.0	-0.0	-0.0

State: (0, 0, **1**)

	A	B	C
A	0.4	-0.3	2.2
B	-0.2	-0.9	1.6
C	2.3	1.5	4.0

(e) **QPLEX** payoffs estimations for each state

State: (1, 0, **0**)

	A	B	C
A	3.9	-0.2	0.3
B	-0.2	-1.0	-0.6
C	0.3	-0.6	-0.2

State: (0, 1, **0**)

	A	B	C
A	-0.1	-0.1	-0.0
B	-0.1	-0.1	-0.1
C	-0.0	-0.1	-0.0

State: (0, 0, **1**)

	A	B	C
A	-0.1	-0.6	0.3
B	-0.6	-0.9	-0.2
C	0.3	-0.2	3.9

(f) **COLA** payoffs estimations for each state

State: (1, 0, **0**)

	A	B	C
A	4.1	-0.3	0.3
B	-0.3	-1.2	-0.5
C	0.4	-0.5	0.1

State: (0, 1, **0**)

	A	B	C
A	0.0	0.0	0.0
B	0.0	0.0	0.0
C	0.0	0.0	0.0

State: (0, 0, **1**)

	A	B	C
A	0.1	-0.5	0.3
B	-0.5	-1.1	-0.3
C	0.4	-0.3	4.0

(g) **MA^{2}E** payoffs estimations for each state

State: (1, 0, **0**)

	A	B	C
A	4.1	-0.3	0.3
B	-0.3	-1.2	-0.6
C	0.4	-0.6	0.1

State: (0, 1, **0**)

	A	B	C
A	-0.4	-0.1	-0.2
B	-0.1	0.8	0.1
C	-0.2	0.1	-0.0

State: (0, 0, **1**)

	A	B	C
A	0.1	-0.5	0.4
B	-0.6	-1.2	-0.3
C	0.4	-0.3	4.0

(h) **LIMARL (ours)** payoffs estimations for each state

Figure 7: Two-Agent, Three-State, Three-Action Matrix Game with Deterministic Rewards and Transitions. Table (a) shows the ground-truth payoffs for each state. Tables (b)–(g) show estimated Q-values produced by various baseline algorithms. Table (h) shows the results from our proposed method. Due to partial observability, S_1 and S_2 yield identical observations, which causes many methods to misrepresent their true values.

D.1 Experimental Setup

Each algorithm is trained and evaluated using the same matrix game environment. Key experimental parameters include:

- **Exploration:** We fix the exploration rate $\epsilon = 1.0$ throughout training for all algorithms. This setup focuses purely on the value estimation challenge, ensuring that all state-action pairs are visited sufficiently often.
- **Discount Factor:** We use $\gamma = 0.0$, so that the estimated values depend only on the payoff of the joint action played and not on the future states, since the transitions are deterministic and independent of actions.
- **Episode Length:** Each episode spans 10 steps, and transitions cycle through states in a fixed order: $S_1 \to S_2 \to S_3 \to S_1$.
- **Partial Observability:** Only the last dimension of each state's one-hot vector is observable by the agents. As a result, S_1 and S_2 yield identical observations.
- **Evaluation:** Each method is run using 3 random seeds. The Q-value tables shown in Figure 7 are the element-wise average over these seeds.

D.2 Evaluated Algorithms

We evaluate the following MARL algorithms: **QMIX** [32, 31], **QPLEX** [43], **OW-QMIX** [30], **CW-QMIX** [30], **COLA** [44], and **MA^{2}E** [19].

D.3 Findings and Analysis

Across all baseline algorithms, as shown in Figure 7 a consistent pattern emerges:

- **Accurate Value Estimation in S_1 and S_3:** All baseline methods are able to assign high utility to the correct joint actions in states S_1 and S_3, where either the reward is clearly distinct (e.g., action A in S_1) or where the state is uniquely identifiable (e.g., S_3 with observation 1).
- **Failure in S_2:** No baseline method successfully identifies action B as the optimal joint action in S_2. Instead, these methods mainly average over all possible actions.
- **Our Method's Success:** In contrast, our method consistently recovers the optimal policy, even in S_2. By leveraging self-supervised representation learning, it learns a latent encoding that correctly differentiates between S_1 and S_2, enabling accurate payoff estimation under partial observability.

D.4 Implementation Details

For fairness, all algorithms use the same configuration where applicable. An example configuration file for QMIX is shown below:

```
# QMIX hyperparameters
action_selector: "epsilon_greedy"
epsilon_start: 1.0
epsilon_finish: 1.0
gamma: 0.0
batch_size_run: 8
buffer_size: 5000
batch_size: 500
optimizer: 'adam'
t_max: 50000
target_update_interval: 200
mac: "n_mac"
agent: "n_rnn"
learner: "nq_learner"
mixer: "qmix"
mixing_embed_dim: 32
hypernet_embed: 64
lr: 0.001
td_lambda: 0.6
```

D.5 Evaluation Method

To evaluate the estimated payoff matrices, we fix the state and sweep through all possible joint actions $(a_1, a_2) \in \{A, B, C\}^2$. The Q-values are computed for each pair, and the resulting 3×3 matrix is averaged over batches and seeds. Our evaluation script ensures proper alignment between the global state and individual Q-values from agents, passing them through the appropriate mixer module to compute global Q-values.

D.6 Conclusion

This matrix game poses a deceptively simple yet fundamentally hard challenge for MARL algorithms under partial observability. Many well-established methods fail due to partial observability, while our method successfully overcomes this via representation learning, clearly highlighting the importance of learned latent states in decentralized multi-agent scenarios.

E Extended Results: Classification Game and Inference Ablation

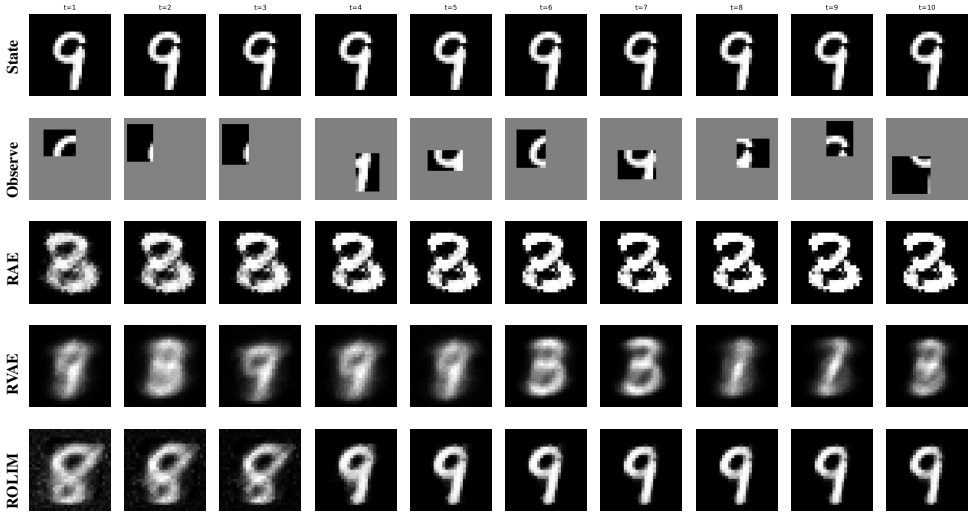


Figure 8: **Qualitative reconstructions across model variants.** Top: full input (**State**) seen only by the oracle encoder. Second: partial inputs (**Observe**) available to agents. Bottom: reconstructions from (RAE), (RVAE), and (ROLIM), conditioned on the same observation sequence.

Setup. To evaluate our latent inference module in a controlled setting, we introduce the *MNIST classification game*, a simplified cooperative multi-agent environment. At each timestep, a global image s_t (an MNIST digit) is partially and independently masked for each agent, producing local observations that evolve over time: $\tau_t^i = (o_0^i, a_0^i, \ldots, o_t^i)$. Agents aim to classify the digit using only their own observation histories—without communication.

The core challenge is to infer a latent representation $\hat{z}_t^i$ that reflects the true global state s_t under severe partial observability. Ideally, this decentralized latent should align with a centralized reference encoding $z_t = f_\theta(s_t)$, learned from full-state supervision. This setting captures key challenges in real-world MARL: partial observability, decentralized inference, and distributed learning dynamics.

Ablation Design. We ablate the inference objective by varying two critical loss components:

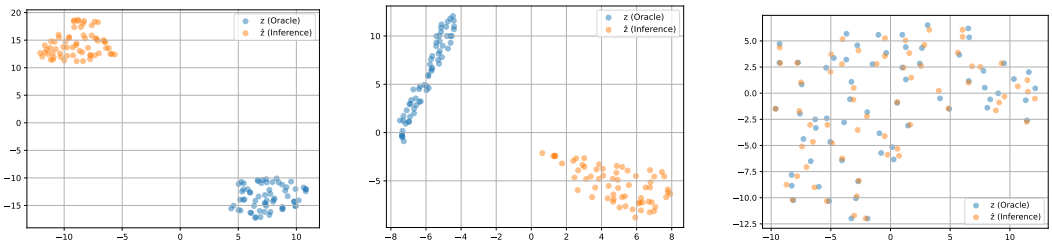
- **Alignment loss** (λ_{align}): Encourages $\hat{z}_t^i$ to align with the centralized latent z_t.
- **KL regularization** (λ_{KL}): Promotes compact and regular latent spaces.

We evaluate six combinations with $\lambda_{\text{align}} \in \{0, 1.0\}$ and $\lambda_{\text{KL}} \in \{0, 0.001, 0.01, 0.1\}$, allowing us to isolate the impact of each component.

Evaluation Metrics. To assess inference quality, we report:

- **Reconstruction loss:** Mean squared error between $g_\psi(\hat{z}_t^i)$ and s_t.
- **Alignment loss:** Squared L_2 distance between $\hat{z}_t^i$ and z_t.
- **KL divergence:** Regularization against a unit Gaussian prior.
- **Classification accuracy:** Task performance from a decoder applied to $\hat{z}_t^i$.
- **Latent variance:** Diversity of $\hat{z}_t^i$ across the dataset.

Inference Results. Table 4 summarizes the final inference metrics. Notably, the configuration with $\lambda_{\text{KL}} = 0.01$ and $\lambda_{\text{align}} = 1.0$ achieves the best overall tradeoff between reconstruction fidelity and alignment, while maintaining regularized latents.



(a) RAE ($\lambda_{\text{KL}} = 0, \lambda_{\text{align}} = 0$) (b) RVAE ($\lambda_{\text{KL}} = 0.1, \lambda_{\text{align}} = 0$) (c) ROLIM($\lambda_{\text{KL}} = 0, \lambda_{\text{align}} = 1$)

Figure 9: t-SNE projections of oracle latent $\mathbf{z}$ and inferred latent $\hat{\mathbf{z}}$. Each point represents a sample colored by digit class.

Configuration	Recon ↓	Align ↓	KL ↓	Acc ↑ (%)	$\hat{z}$ Var ↑
RAE ($\lambda_{\text{KL}} = 0, \lambda_{\text{align}} = 0$)	0.8416	105.7743	9399.46	24.00	**4.6160**
RVAE ($\lambda_{\text{KL}} = 0.001, \lambda_{\text{align}} = 0$)	0.9227	0.2427	5.11	15.00	0.0130
RVAE ($\lambda_{\text{KL}} = 0.01, \lambda_{\text{align}} = 0$)	0.9347	0.0178	**0.06**	12.00	0.0001
ROLIM($\lambda_{\text{KL}} = 0.01, \lambda_{\text{align}} = 1.0$)	0.2336	**0.0018**	**1.64**	47.00	0.0196
ROLIM($\lambda_{\text{KL}} = 0.001, \lambda_{\text{align}} = 1.0$)	**0.1439**	**0.0135**	18.26	**50.50**	0.1589
ROLIM($\lambda_{\text{KL}} = 0, \lambda_{\text{align}} = 1.0$)	**0.0930**	0.8125	1405.80	**68.00**	**7.7520**

Table 4: Inference and evaluation metrics (epoch 49) for TrainDecoder ablation configurations.

Key Takeaways.

- **Alignment is essential:** Without alignment, even low KL values fail to prevent drift in $\hat{z}_t^i$, leading to poor reconstructions and unstructured latents.
- **KL alone is insufficient:** Higher regularization improves compactness but harms semantic expressiveness (e.g., KL=0.1, Align=0).
- **Best tradeoff:** Moderate KL with alignment achieves the most structured, predictive, and transferable representations.

Conclusion. These ablations demonstrate that partial observations alone are inadequate for learning useful representations. Alignment with a centralized reference is crucial for semantic structure, while KL regularization improves generalization but must be balanced carefully.

F Ablation Study on Fine-tuning Threshold δ

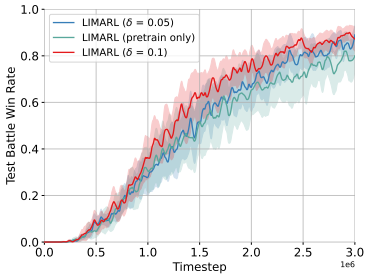
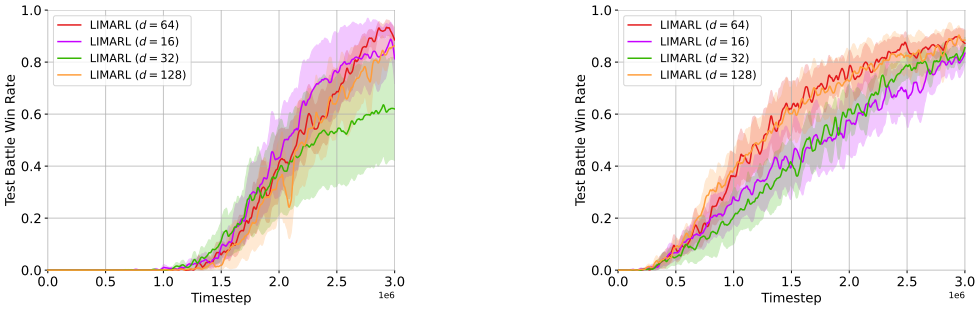


Figure 10: Performance impact of varying the fine-tuning threshold δ on the SMAC `MMM2` (super-hard) scenario. The results highlight the sensitivity of the model to δ, illustrating the trade-off between adaptation and overfitting.

We investigate the effect of the fine-tuning percentage threshold δ, which determines the percentage increase of the tracked mean to launch fine-tuning. Figure 10 presents the ablation results on the challenging `MMM2` scenario in SMAC. The study reveals that both very low ($\delta = 0.05$) and very high (i.e., pretrain only) values of δ can lead to suboptimal performance, while a moderate setting ($\delta = 0.1$) yields the best balance perfromance.

G Ablation Study on Fine-tuning the Intrinsic Dimension d



(a) `corridor` (super-hard)

(b) `MMM2` (super-hard)

Figure 11: Impact of varying the latent dimension d on LIMARL performance in two super-hard SMAC scenarios.

To understand the sensitivity of our method to the dimensionality of the latent space, we conducted an ablation study varying the intrinsic dimension d used in both modules (SRM and ROLIM). Specifically, we evaluated $d \in \{16, 32, 64, 128\}$ on two super-hard SMAC scenarios: `corridor` and `MMM2`.

Throughout the main paper, we fix $d = 64$, motivated by its strong empirical performance. This choice is supported by our findings here: $d = 64$ consistently achieves the highest final win rate across both environments. Interestingly, $d = 128$ performs comparably well, suggesting that a slightly larger latent space can retain performance.

In contrast, lower-dimensional latents ($d = 16$ and $d = 32$) often degraded performance. In `corridor`, $d = 32$ performs notably worse, whereas in `MMM2`, $d = 16$ is the least effective. Across both tasks, $d = 16$ and $d = 32$ are the weakest two configurations, demonstrating the importance of having sufficient capacity in the latent space for capturing the complex, partially observable dynamics.

Despite this, LIMARL remains surprisingly robust to changes in d: even the smallest latent size ($d = 16$) allows for competitive learning and non-trivial win rates. This resilience indicates that the core inference mechanism is not overly sensitive to the exact choice of d, as long as it remains within a reasonable range.

Overall, this study highlights that while $d = 64$ offers a reliable trade-off between expressiveness and efficiency, LIMARL exhibits a degree of robustness to this hyperparameter, reinforcing the general applicability of our latent inference framework.