SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2026, Vol. 69, Iss. 9, 192202:1–192202:12
https://doi.org/10.1007/s11432-025-4850-y

# Self-predictive Mamba for efficient multi-agent policy learning

Zhaohan FENG[1,2], Runqing WANG[1], Boxuan ZHANG[1], Jian SUN[1,2], Fang DENG[1,2] & Gang WANG[1,2*]

[1]*National Key Lab of Autonomous Intelligent Unmanned Systems, Beijing Institute of Technology, Beijing 100081, China*
[2]*Beijing Institute of Technology Chongqing Innovation Center, Chongqing 401120, China*

Received 15 February 2025/Revised 30 March 2025/Accepted 3 June 2025/Published online 26 May 2026

**Abstract** Environmental non-stationarity remains a fundamental challenge in multi-agent reinforcement learning (MARL), hindering the efficiency of policy learning. Existing approaches primarily mitigate this issue by incorporating historical information into decision-making. However, widely adopted sequence modeling architectures, such as recurrent neural networks (RNNs) and Transformers, exhibit inherent limitations: RNNs struggle to capture long-range temporal dependencies, while Transformers, despite their superior encoding capabilities, suffer from the inflexibility imposed by a fixed-size context window and the quadratic computational complexity of self-attention. Motivated by the belief that self-supervised feature learning can enhance reinforcement learning (RL) efficiency, we propose self-predictive Mamba (SPMamba), a novel architecture that integrates Mamba's superior sequence reasoning capabilities with a self-supervised auxiliary learning objective to facilitate the optimization of decentralized individual policies. Multiple challenging evaluations demonstrate that SPMamba is significantly superior to several state-of-the-art baselines.

**Keywords** multi-agent reinforcement learning, self-supervised learning, multi-agent collaboration, structured state space model

**Citation** Feng Z H, Wang R Q, Zhang B X, et al. Self-predictive Mamba for efficient multi-agent policy learning. Sci China Inf Sci, 2026, 69(9): 192202, https://doi.org/10.1007/s11432-025-4850-y

## 1 Introduction

Multi-agent reinforcement learning (MARL) has gained significant attention due to its successes in diverse domains [1,2], including autonomous driving [3] and robot swarm navigation [4,5]. Compared to traditional control methods that heavily rely on manual design and task-specific tuning [6–8], MARL demonstrates superior generalization capabilities in complex multi-agent tasks. One fundamental challenge in MARL lies in the inherent non-stationarity of the environment, arising from two major factors [9]. First, each agent operates with a limited observation range, and its perceived environment evolves dynamically as both the global and local states transition. Second, the continuous adaptation of agents' policies results in non-stationary observation transition dynamics from the perspective of individual agents, making it difficult for agents to model the environment. Granting agents access to the global state could alleviate this issue to some extent, but the associated communication overhead often renders this approach impractical. Consequently, state-of-the-art (SOTA) MARL methods commonly employ recurrent neural networks (RNNs) [10–12] as well as Transformers [13,14] to extract spatial-temporal features from historical observations [15]. Despite their utility, RNNs often struggle to capture complex environmental dynamics due to their simplistic architectures, and Transformers, despite their stronger encoding capabilities, suffer from quadratic computational complexity and rigid context window constraints. Thus, environmental non-stationarity remains a persistent challenge in MARL.

Self-supervised representation learning has recently been introduced in reinforcement learning (RL) to promote policy learning in both model-free [16,17] and model-based approaches [18,19]. However, applying self-supervised learning to MARL presents unique challenges. Due to environmental non-stationarity, predicting future observations solely from an agent's local observations and actions is inherently unreliable. While enabling agents to exchange observations can mitigate this issue by providing agents with access to global information [20–22], it also introduces additional communication overhead, leading to increased training and decision-making costs, as well as undermines the practicality of the resulting algorithms. Inspired by the critical role of historical information in distributed

* Corresponding author (email: gangwang@bit.edu.cn)

 

multi-agent decision-making, we leverage historical observations for decentralized self-supervised feature learning, enabling efficient policy learning in multi-agent settings.

In this paper, we propose self-predictive Mamba (SPMamba), a novel architecture that integrates historical sequence encoding with self-supervised feature learning to enhance distributed multi-agent policy optimization. SPMamba relies on the Mamba model [23] to efficiently extract spatial-temporal features from agents' historical observations. To enforce consistency and predictability, we introduce a multi-layer perceptron variational auto-encoder (MLP-VAE) which aggregates dense representations from raw observations, along with a transition decoder that predicts the next encoded observation based on the model's output. Unlike typical world model learning, the MLP-VAE is trained jointly with the policy, focusing solely on reward maximization without input reconstruction or representation learning. Numerical results show that SPMamba significantly outperforms RNN-based policies on challenging tasks from the StarCraft II multi-agent challenge (SMAC) [24], while also providing substantial improvements over the direct application of Mamba as the policy network.

## 2 Related work

**Representation encoding in MARL.** Early MARL algorithms adopted MLP-based policy and value networks, which were commonly used in single-agent RL settings [25, 26]. Conversely, when dealing with partially observable tasks, the necessity of utilizing historical information becomes non-negligible [27]. The most popular recurrent network structure is gated-recurrent unit (GRU). Counterfactual multi-agent (COMA) policy gradients utilize GRU in the actor network and multi-layer perceptron (MLP) consisting of multiple layers with ReLU as the critic network [28]. In QMIX and its various variants, all agents utilize a GRU as the core component [10, 29]. SOTA policy-based MARL algorithms also widely adopt GRUs as policy and critic networks [12].

**Self-supervised RL.** Self-supervised representation learning methods have demonstrated remarkable data efficiency in sequence modeling tasks such as vision and language processing, particularly in low-data regimes [30, 31]. Inspired by VAE, self-supervised representation learning has recently been widely used in world modeling [18, 19, 32] to extract relevant features for transition prediction. Through reconstructing the raw input and predicting the future latents or inputs, world models can learn complicated environment dynamics for training competitive agents with a small amount of data. Self-supervised representation learning can also improve data efficiency in deep Q-learning [16, 33]. To address the non-stationary nature of MARL settings, some studies replace the global states with the joint observations of all the agents and attempt to learn transition dynamics based on it [22, 34]. Although such methods are inefficient as joint observations are not a reliable substitute for the global state, they often contain redundancy from agents observing each other while potentially missing critical environment-specific information. Some approaches enable communication among agents [20, 35] to acquire the global state. Nevertheless, this introduces additional communication overhead in the execution phase, which goes beyond the scope of the CTDE framework and compromises the flexibility and scalability of algorithms.

**Structured state space models.** Structured state space models (dubbed S4 models) are a family of efficient models for sequence modeling tasks such as sequence signal compression, audio generation, image and audio modeling, to name just a few. S4 models have also demonstrated promising performance in partially observable RL [36], in-context RL [37], and imitation learning tasks [38]. Mamba is an advanced variant of the S4 model with selection and a scan [23] which demonstrates superior performance compared to Transformers with similar scales. Recently, Mamba has garnered widespread attention due to its promising long-sequence modeling capabilities, coupled with its linearly scalable computational resource consumption.

## 3 Preliminaries

### 3.1 Partially observable Markov decision process

In MARL, the goal is to learn policies for multiple agents that interact within a shared environment, aiming to maximize a scalar reward signal. This problem is often modeled as a partially observable Markov decision process (POMDP), represented by the tuple $G = \langle S, U, P, R, O, N, \gamma \rangle$, where $s \in S$ represents the global state of the entire multi-agent system and the environment; $u_i \in U$ is the action taken by the $i$-th agent forming the joint action $\boldsymbol{u} = [u_1\ u_2\ \cdots\ u_N] \in U^N$; $P(s'|s, \boldsymbol{u}) : S \times U^N \times S \to [0, 1]$ is the state transition function; $R$ is the reward function $r = R(s, \boldsymbol{u})$ that gives a scalar reward signal according to the global state $s$ and the joint action $\boldsymbol{u}$; $O$ describes the observation function that generates observation $o_i$ for the $i$-th agent; $N$ is the total number of agents; and $\gamma$ is the discount factor in calculating returns. At each time step, each agent $i \in N$ receives an observation $o_i \in O$

and selects an action $u_i$, resulting in a joint action $\boldsymbol{u} = [u_1\ u_2\ \cdots\ u_N] \in U^N$. The global states then transition to $s'$ according to $\boldsymbol{u}$ and the reward $r$ is given. In fully collaborative settings, the reward is shared by the entire multi-agent system. The objective of policy training is to learn a policy $\pi(u_i|o_i)$ to maximize the total expected return $L(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{u}_t)]$.

### 3.2 The Mamba model

The Mamba model is a class of SSMs that include a selection mechanism to efficiently handle sequential data [23]. At each time step $t$, the Mamba model takes an input $x_t$, updates a latent state $h_t$, and produces an output $y_t$. The discrete form of the Mamba model is defined as

$$h_t = \bar{\boldsymbol{A}}_t h_{t-1} + \bar{\boldsymbol{B}}_t x_t, \tag{1a}$$

$$y_t = \boldsymbol{C}_t h_t + D x_t, \tag{1b}$$

where, $\bar{\boldsymbol{A}}_t = f_A(\Delta_t, \boldsymbol{A}_t)$ and $\bar{\boldsymbol{B}}_t = f_B(\Delta_t, \boldsymbol{A}_t, \boldsymbol{B}_t), \boldsymbol{C}_t$ are discrete model parameters, $D$ is a learnable weight that controls the contribution of the original input $x_t$ through a shortcut connection. In the standard Mamba implementation, discretization follows the zero-order hold (ZOH) method:

$$\bar{\boldsymbol{A}}_t = \exp(\Delta_t \boldsymbol{A}_t), \tag{2a}$$

$$\bar{\boldsymbol{B}}_t = (\Delta_t \boldsymbol{A}_t)^{-1}(\exp(\Delta_t \boldsymbol{A}_t) - I)\Delta_t \boldsymbol{B}_t. \tag{2b}$$

While $\boldsymbol{A}_t$ is initialized and updated independently, the Mamba model employs a selection mechanism, using linear projections to compute the matrices $\boldsymbol{B}_t = f_B(x_t)$, $\boldsymbol{C}_t = f_C(x_t)$, and $\Delta_t = f_\Delta(x_t)$. To operate over an input sequence $x$ of batch size $B$ with $L$ channels, the SSM in (1) is applied independently to each channel with model parameters $\boldsymbol{A}_t \in \mathbb{R}^{B\times L\times N\times N}$, $\boldsymbol{B}_t \in \mathbb{R}^{B\times L\times N\times 1}$, $\boldsymbol{C}_t \in \mathbb{R}^{B\times L\times 1\times N}$ and $h_t \in \mathbb{R}^{B\times L\times N\times 1}$, the projections are defined as follows:

$$\begin{aligned} f_B(x) &= \mathrm{Linear}_N(x), \\ f_C(x) &= \mathrm{Linear}_N(x), \\ f_\Delta(x) &= \mathrm{Broadcast}_L(\mathrm{Linear}_1(x)), \end{aligned} \tag{3}$$

where the operator $\mathrm{Linear}_N(\cdot)$ projects to an $N$-dimensional space, and $\mathrm{Broadcast}_L(\cdot)$ broadcasts a scalar to an $L$-dimensional space. The term $\Delta_t$ acts analogously to the gating mechanism $g_t$ in $h_t = (1 - g_t)h_{t-1} + g_t x_t$, controlling whether to reset the state based on the current input. Specifically, $\Delta_t$ decides whether to "select" and incorporate the current input $x_t$ while forgetting the previous state, or to retain the current state and ignore the input. Meanwhile, $\boldsymbol{B}_t$ and $\boldsymbol{C}_t$ serve as more fine-grained controllers, determining how $x_t$ influences $h_t$ and how $h_t$ contributes to $y_t$.

## 4 The self-predictive Mamba model

We aim to improve the learning stability and efficiency of decentralized RL policy learning in multi-agent collaboration tasks, which are commonly modeled as POMDPs. Due to the partial observability of the environment, we attempt to conduct self-prediction on agents' historical observations. The overall prediction scheme can be formulated as

$$o^i_{t+1} := f^i_{dyn}\big(\tau^i_t, u^i_t\big), \quad \tau^i_t = [o^i]_{0:t}, \tag{4}$$

where $i$ represents the $i$-th agent, $\tau_t$ is the historical observation sequence from time 0 to $t$, and $f_{dyn}(\cdot)$ is the prediction module that predicts the next observation according to the historical information and current action. The key to self-prediction lies in efficiently modeling from the agent's historical observations. We leverage the Mamba model to achieve this goal. As SPMamba is designed for promoting decentralized decision-making of agents, we omit agent-specific indices in our description whenever clear from context. For example, the observation of the $i$-th agent at time $t$ is denoted by $o_t$, corresponding to $o^i_t$ in full notation. Additionally, for simplicity, variables related to a single agent or time step are denoted with regular symbols, while those associated with multiple agents or time steps are represented using bold symbols. For instance, the observation sequence of the $i$-th agent from time $t$ to $t + k$ is denoted by $\boldsymbol{o}_{t:t+k} = [o_t^\top, o_{t+1}^\top, \ldots, o_{t+k}^\top]$.

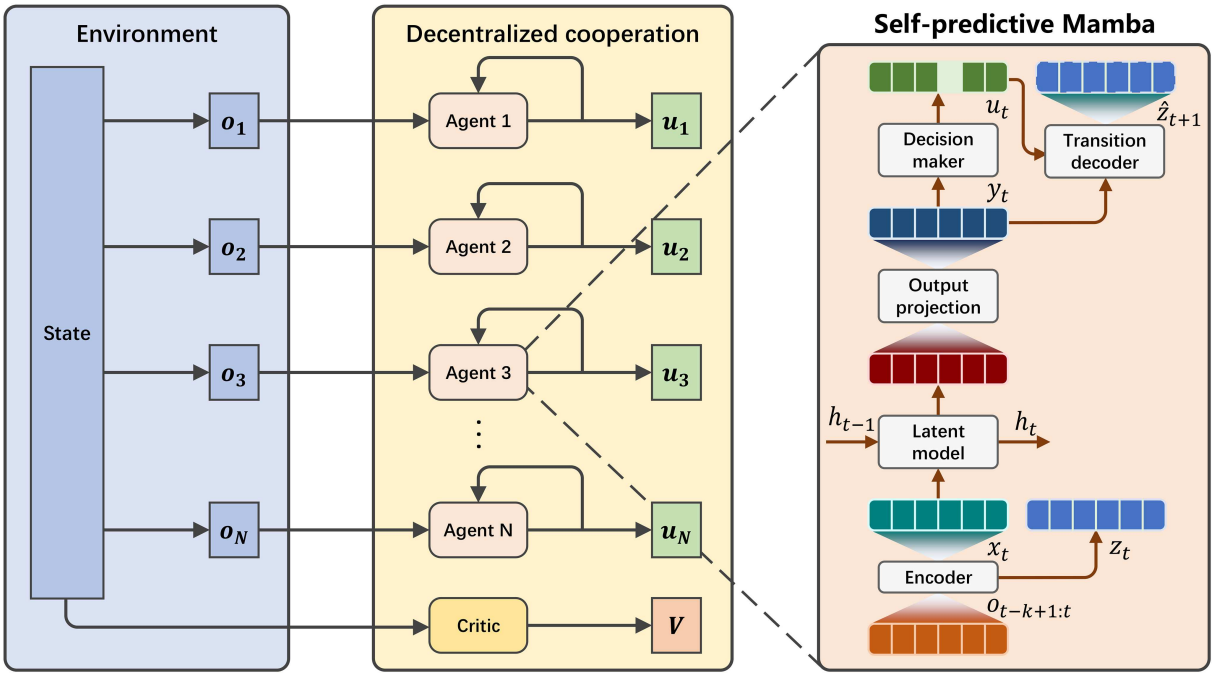


**Figure 1** (Color online) Overall architecture of the proposed SPMamba-based decentralized multi-agent framework. The right panel illustrates the internal structure of the SPMamba model used by each agent. SPMamba recurrently updates the hidden state $h_t$ from the encoded input $z_t$. For computational efficiency, the length of $\boldsymbol{o}_{t-k+1:t}$ is fixed to $k$. At each time $t$, $o_t$ is stored in, and $o_{t-k}$ is popped out.

**Model components.** SPMamba extends the Mamba model to decentralized multi-agent decision-making. Unlike the standard implementation, which processes 1D sequences in parallel, we adapt Mamba for sequential decision-making in a step-wise fashion. The entire paradigm of SPMamba is shown in Figure 1. The architecture consists of 5 components:

$$\begin{aligned}
&\text{Encoder} && x_t = \mathrm{CONV}_\phi(\boldsymbol{z}_{t-k+1:t}), z_t = \mathrm{MLP}_\phi(o_t), && \text{(5a)}\\
&\text{Latent model} && h_t = \mathrm{SSM}_\phi(h_{t-1}, x_t), && \text{(5b)}\\
&\text{Output projection} && y_t = \mathrm{PROJ}_\phi(h_t, x_t), && \text{(5c)}\\
&\text{Decision maker} && u_t \sim p_\phi(u_t|y_t) = \mathcal{U}_t, \mathcal{U}_t = \mathrm{ACT}_\phi(y_t), && \text{(5d)}\\
&\text{Transition decoder} && \hat{z}_{t+1} = \mathrm{DEC}_\phi(y_t, u_t), && \text{(5e)}
\end{aligned}$$

where $\phi$ represents the learnable parameters of SPMamba. Each component is implemented as either neural networks or probability density variables for categorical distributions. At each time $t$, SPMamba encodes the observation $o_t$ to a latent variable $z_t$ and aggregates historical latents $[z]_{t-k+1:t}$ into $x_t$, then updates the hidden state $h_{t-1}$ to $h_t$ and organizes the output $y_t$ for both decision-making and self-prediction.

**Input sequence encoding.** We propose an MLP-VAE in (5a) to encode raw input observations. To reduce redundancy in the raw observations, the observation $o_t$ is first encoded into a latent representation $z_t \in \mathbb{R}^{1\times d\epsilon}$. Here, $d$ denotes the dimension of the model's output $y_t$, and $\epsilon$ is the expansion coefficient. The historical sequence of encoded observations $\boldsymbol{z}_{t-k+1:t}$ is then aggregated into $x_t \in \mathbb{R}^{1\times d\epsilon}$ through a convolution neural network (CNN). The encoder is implemented as a combination of a single-layer MLP and a CNN.

**Mamba latent model.** The key point of decentralized decision-making and self-supervised learning revolves around efficiently modeling over the observation trajectory $\tau_t$. Recent studies primarily employ RNNs and Transformers for this purpose. However, RNNs struggle to capture long-range dependencies, while the long-sequence modeling capability of Transformers depends on the length of the context window, which lacks flexibility in the step-by-step observing-deciding-interacting loop. Accordingly, we implement the latent model described in (5b) as a Mamba model. Specifically, the $\mathrm{SSM}_\phi$ in (5b) denotes (1a). The latent model updates the hidden state $h_{t-1} \in \mathbb{R}^{1\times d\epsilon\times 1}$ to $h_t$ using the aggregated encoded observation $x_t$. All linear projections for calculating $\bar{\boldsymbol{A}}_t$ and $\bar{\boldsymbol{B}}_t$ are implemented as single-layer MLPs.

**Nonlinear output projection.** A parameterized nonlinear transformation is applied for enhancing the latent model's output in (5c). It applies a nonlinear transformation using a coefficient tensor that is parameterized by $x_t$

and activated by SiLU [39]:

$$y_t = \mathrm{NORM}\big[(\boldsymbol{C}_t h_t + Dx_t) \odot \mathrm{SiLU}[\mathrm{MLP}_\phi(x_t)]\big], \tag{6}$$

where NORM$(\cdot)$ refers to the layer normalization and $\odot$ is the Hadamard product. The nonlinear output projection serves as a sophisticated combination of the residual connection and the gating mechanism, which not only facilitates model convergence but also dynamically focuses on relevant parts of historical information based on the current input. Just like $\bar{\boldsymbol{A}}_t$ and $\bar{\boldsymbol{B}}_t$ in the latent model, the linear projection used in computing $\boldsymbol{C}_t$ is also implemented as a single-layer MLP.

**Categorical activation.** To facilitate decision-making in a discrete action space, the decision output head described in (5d) is parameterized as a categorical distribution. The latent model's output $y_t$ is activated to be a categorical distribution $\mathcal{U}$, and the action $u_t$ is selected by sampling once from this categorical distribution.

**Transition decoder.** The transition decoder is implemented similarly to the encoder as a single-layer MLP. Unlike common world models that regularize the latent representation towards a prior distribution, we train the decoder solely to predict the next encoded observation $z_{t+1}$ based on the model's output $y_t$ and the action taken by the agent $u_t$. The rationale is that the model's output will be automatically regularized through end-to-end policy training, eliminating the need for explicit regularization toward a prior.

## 5 SPMamba for policy learning

In this work, we implement SPMamba into MAPPO [12] and describe the training process for the resulting actor and critic. While we focus on MAPPO in our simulations, it is worth noting that SPMamba is designed as a general policy module, making it easily integrable into other policy-based algorithms that adopt distributed execution as well.

### 5.1 Experience buffer

A buffer storing the past experiences of executing the policy is maintained to train the policy. The buffer contains sequences of agents' observations $\boldsymbol{o}_{1:N,0:T-1}$, global states $\boldsymbol{s}_{0:T-1}$, actions $\boldsymbol{u}_{1:N,0:T-1}$ along with the action log probabilities, the team rewards $\boldsymbol{r}_{0:T-1}$ and the next observations $\boldsymbol{o}_{1:N,1:T}$, where $T$ is the maximum step per episode and $N$ is the total number of agents. We randomly shuffle the data in the whole buffer and pack them into one single batch to ensure the stability of training. At the end of each training phase, the buffer is cleared in preparation for storing new experience data.

### 5.2 Actor learning

The actor $u_t = \pi_\phi(o_t)$ is realized using the SPMamba designed in Section 4, and is simultaneously guided by both the reward signal and the self-supervised signal. We enable the experience data sharing among all the agents, and therefore, the experiences $\{o_t, u_t, o_{t+1}, r_t\}$ can be sampled randomly regardless of the agent. The total loss for the actor is defined as follows:

$$L(\phi) = \frac{1}{NT}\sum_{n=1}^{N}\sum_{t=0}^{T-1}\left[L_{n,t}^{\mathrm{act}}(\phi) + \alpha L_{n,t}^{\mathrm{pred}}(\phi)\right], \tag{7}$$

where $\alpha$ is the weight coefficient for self-supervised representation learning. We share the policy among all the agents; thus the agent index $n$ is omitted. The policy loss $L_t^{\mathrm{act}}(\phi)$ and the self-predicting loss $L_t^{\mathrm{pred}}(\phi)$ are given by

$$L_t^{\mathrm{act}}(\phi) = \min\left[\frac{\pi_\phi(u_t|o_t)}{\pi_\phi^{\mathrm{old}}(u_t|o_t)}A_\psi^{\pi_\phi}(s_t,\boldsymbol{u}_t), \mathrm{clip}\left(\frac{\pi_\phi(u_t|o_t)}{\pi_\phi^{\mathrm{old}}(u_t|o_t)}, 1-\eta, 1+\eta\right)A_\psi^{\pi_\phi}(s_t,\boldsymbol{u}_t)\right], \tag{8a}$$

$$L_t^{\mathrm{pred}}(\phi) = \max\big(\mu, \mathrm{KL}\big[\mathrm{sg}(\mathrm{softmax}(z_t))||\mathrm{softmax}(\hat{z}_t)\big]\big), \tag{8b}$$

where $\pi_\phi^{\mathrm{old}}$ is the previous policy used in collecting the data $\{o_t, u_t, o_{t+1}, r_t\}$, $\pi_\phi$ is the current policy, $A_\psi^{\pi_\phi}(s_t,\boldsymbol{u}_t)$ is the advantage function, $\psi$ represents the learnable parameters of the critic function, $\eta$ is the clip parameter that controls the deviation between the current policy $\pi_\phi$ and the old policy $\pi_\phi^{\mathrm{old}}$, sg$(\cdot)$ is the operation of stop-gradients, and $\mu$ is the early-stop coefficient of self-supervised representation training. We compute $A_\psi^{\pi_\phi}$ utilizing the $K$-step general advantage estimation (GAE),

$$A_\psi^{\pi_\phi}(s_t,\boldsymbol{u}_t) = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{K-1}\delta_{t+K-1}, \tag{9}$$

where $\delta_t = \gamma V_\psi^{\pi_\phi}(s_{t+1}) + r(s_t, \boldsymbol{u}_t) - V_\psi^{\pi_\phi}(s_t)$, $r(s_t, \boldsymbol{u}_t)$ is the shared team reward, and $V_\psi^{\pi_\phi}(s_t)$ is the critic function with learnable parameters collectively denoted by $\psi$.

The self-predicting loss $L_t^{\text{pred}}(\phi)$ is the KL divergence between the discrete categorical distributions parameterized by the encoded observation $z_t$ and the predicted transition $\hat{z}_t$. We stop propagating the gradients of $\text{softmax}(z_t)$ in $L_t^{\text{pred}}(\phi)$. We also halt the training of the transition decoder early by setting $L_t^{\text{pred}}(\phi)$ to $\mu$ if the self-predicting loss falls below a constant $\mu$ to reduce its interference in policy training when the transition prediction is relatively accurate.

### 5.3 Critic learning

The critic function $V_\psi^{\pi_\phi}(s_t)$ shared among all the agents is implemented as a gated recurrent unit (GRU), and trained in a self-supervised manner. We choose the advantage function in (9) as the critic target and use the mean squared error (MSE) as the critic loss:

$$L(\psi) = \frac{1}{T}\sum_{t=0}^{T-1} \text{MSE}\big[\text{sg}\big(A_\psi^{\pi_\phi}(s_t, \boldsymbol{u}_t) + V_\psi^{\pi_\phi}(s_t)\big) - V_\psi^{\pi_\phi}(s_t)\big], \tag{10}$$

where $\text{sg}(\cdot)$ is the stop-gradient manipulation. The $V_\psi^{\pi_\phi}(s_t)$ is updated using semi-gradient TD target for stabilizing the training process.

## 6 Experiments

We evaluate the performance of SPMamba on six of the most challenging SMAC tasks, characterized by asymmetric force distributions and heterogeneous agents. Our experiments aim to address the following key questions.

- Q1. How effective is the SPMamba policy compared to a GRU-based policy? Additionally, how effective is self-supervised representation learning compared to directly using the Mamba model as the policy?
- Q2. Is the MLP-VAE structure effective compared to other proven effective architectures, specifically the categorical-VAE employed by Dreamer [18] and the SimNorm-VAE employed by TDMPC2 [32]?
- Q3. Is the current design of SPMamba effective enough?

To address Q1, we conduct comparative evaluations among SPMamba, the vanilla Mamba policy, and the GRU-based policy. Details of these evaluations are provided in Subsection 6.2. To address Q2, we compare the MLP-VAE in SPMamba with the categorical-VAE from Dreamer and the SimNorm-VAE from TDMPC2. These comparisons are outlined in Subsection 6.3. To address Q3, we perform ablation studies that explore alternative structures designed for SPMamba, as described in Subsection 6.4.

### 6.1 Experimental settings

#### 6.1.1 *Benchmark*

We conduct experiments on SMAC and SMACv2 [40], two benchmarks including various multi-agent combat tasks that require defeating enemy forces controlled by the built-in AI script, which is well known for their complexity and scalability. We carefully select the six most challenging tasks in SMAC for evaluating the performance of the SPMamba against baselines, specifically 5m_vs_6m, 8m_vs_9m, 6h_vs_8z, 3s5z_vs_3s6z, MMM2 and 3s_vs_5z. All the selected tasks include asymmetric force distributions, implying that the enemy force is always stronger than the ally force. Specifically, 5m_vs_6m and 8m_vs_9m are basic asymmetric combat tasks with different scales, while 5m_vs_6m presents a greater challenge due to the higher force ratio between opposing sides; 8m_vs_9m is more difficult to learn because it involves a larger number of agents, and 3s5z_vs_3s6z and MMM2 are asymmetric tasks with multiple types of heterogeneous agents which significantly increases their complexity. Moreover, 3s_vs_5z is a task requiring long-term skill learning. We also evaluate our approach on three tasks in SMACv2, each featuring three distinct types of heterogeneous agents.

#### 6.1.2 *Hyperparameters*

We build SPMamba upon the implementation of MAPPO [12], and maintain consistent hyperparameter settings across all the tasks and the compared algorithms, except in 3s_vs_5z which requires long-term skill learning, where the historical observation sequence length of SPMamba is set to $k = 8$ and the early-stop coefficient of the self-predicting loss is fixed at $\mu = 0$. The maximum environment steps for SPMamba and all the on-policy baselines

**Table 1** Performance comparison of SPMamba and baseline methods on SMAC benchmark tasks in terms of win rate (%), reported as mean ± std. On-policy methods are trained for up to $1 \times 10^7$ environment steps. Off-policy methods marked with $*$ are trained under the same runtime budget as SPMamba. The best results are in bold.

| Task | SPMamba | Mamba | MAPPO | MAT | HAPPO | *RODE | *QMIX |
|---|---|---|---|---|---|---|---|
| 3s_vs_5z | **89.5±7.9** | 83.2±29.7 | 23.4±42.7 | 0 | 25.2±41.9 | 64.2±23.8 | 41.7±43.4 |
| 5m_vs_6m | **41.0±24.7** | 36.7±11.4 | 23.0±31.3 | 55.1±0.9 | 65.4±8.3 | 32.4±6.6 | 21.8±6.4 |
| 8m_vs_9m | **95.6±0.5** | 88.2±3.6 | 78.6±8.9 | 77.8±3.6 | 80.7±6.8 | 62.8±4.5 | 43.0±16.3 |
| 6h_vs_8z | **47.3±24.5** | 25.5±7.3 | 24.9±7.9 | 34.8±15.4 | 3.0±0.3 | 4.6±8.0 | 12.6±11.8 |
| 3s5z_vs_3s6z | **83.7±2.7** | 62.4±40.4 | 56.5±23.0 | 6.8±11.3 | 5.1±6.4 | 4.0±7.0 | 32.7±28.3 |
| MMM2 | **84.0±7.9** | 74.0±6.2 | 77.5±15.2 | 29.4±16.6 | 73.2±3.2 | 79.3±9.2 | 73.0±1.4 |

**Table 2** Comparison of SPMamba and baseline methods on SMACv2 tasks in terms of average episode reward (mean ± std). The maximum number of environment steps is $1 \times 10^7$. The best results are in bold.

| Task | SPMamba | Mamba | MAPPO | MAT | HAPPO |
|---|---|---|---|---|---|
| zerg 5_vs_5 | **13.0±0.5** | 12.5±1.1 | 11.2±0.6 | 14.3±1.1 | 13.8±0.3 |
| terran 5_vs_5 | **13.4±0.3** | 13.4±0.9 | 12.0±0.6 | 14.0±1.7 | 15.4±0.8 |
| protoss 5_vs_5 | **16.5±0.3** | 15.7±0.8 | 15.9±0.9 | 16.3±1.4 | 16.7±0.2 |

are set to 1e7. As off-policy methods are generally more sample-efficient but slower than on-policy approaches, we compare off-policy baselines with SPMamba under equal runtime. Each experiment is conducted with 4 different random seeds in a Python 3.7 environment using StarCraft II 2.4.6. All experiments are performed on a PC equipped with an RTX 3090 GPU.

### 6.2 Comparing with baselines

We conduct a comprehensive performance evaluation of SPMamba against SOTA baselines. Given that SPMamba is built upon MAPPO, we select MAPPO-FP [12] as a primary baseline. To isolate the impact of our proposed self-supervised feature learning module, we replace the recurrent policy network in MAPPO-FP with a Mamba model, forming the Mamba policy as an additional baseline. Furthermore, to assess SPMamba's effectiveness in a broader context, we compare it with two strong off-policy algorithms: QMIX [10] and RODE [41]. QMIX employs GRUs to encode local observation histories of agents and constructs the global Q-function out of local utilities, while RODE extends QMIX with explicit role assignment to enhance agent cooperation. For a more comprehensive comparison, we also include MAT [14] and HAPPO [42], two SOTA baselines that adopt sequential policy updates. In SMAC, we compare the average win rates achieved by different algorithms. In SMACv2, each episode begins with both teams being randomly initialized, making it considerably more challenging to secure consistent wins. To ensure a more reliable and insightful comparison, we adopt the average episode reward as our primary evaluation metric rather than relying on the average win rate. The average runtime of SPMamba across all tasks is 22.73 h, compared to 22.12 h for RNN and 23.66 h for Transformers.

As shown in Tables 1 and 2, SPMamba consistently outperforms all baselines across all tasks in SMAC. While the Mamba policy benefits from Mamba's advanced sequence modeling capabilities and achieves improvements over the vanilla MAPPO with GRUs, its lack of a self-supervised learning module prevents it from reaching the performance level of SPMamba. In SMACv2, despite the extreme randomness in the environment and task settings causing SPMamba, which follows MAPPO's original update scheme, to be outperformed by MAT and HAPPO, it still achieves a significant improvement over MAPPO. Furthermore, SPMamba demonstrates a clear advantage over the Mamba policy that does not incorporate self-predictive feature learning. In general, we attribute its superior performance to three key advancements: (1) a carefully designed distributed self-supervised learning module based on agents' historical observations, (2) sequence modeling of agent observation histories using the Mamba model, and (3) observation encoding via the MLP-VAE. The following sections provide an in-depth analysis of these components and further examine the advantages of the SPMamba architecture.

### 6.3 Encoder structure

#### 6.3.1 *Alternative structures*

Recently, there are two widely employed alternatives of the proposed MLP-VAE in RL, namely categorical-VAE [18] and SimNorm-VAE [32]. Specifically, categorical-VAE samples multiple categorical variables as the encoded features:

$$z_t \sim q_\phi(z_t|o_t) = \mathcal{Z}_t, \tag{11}$$

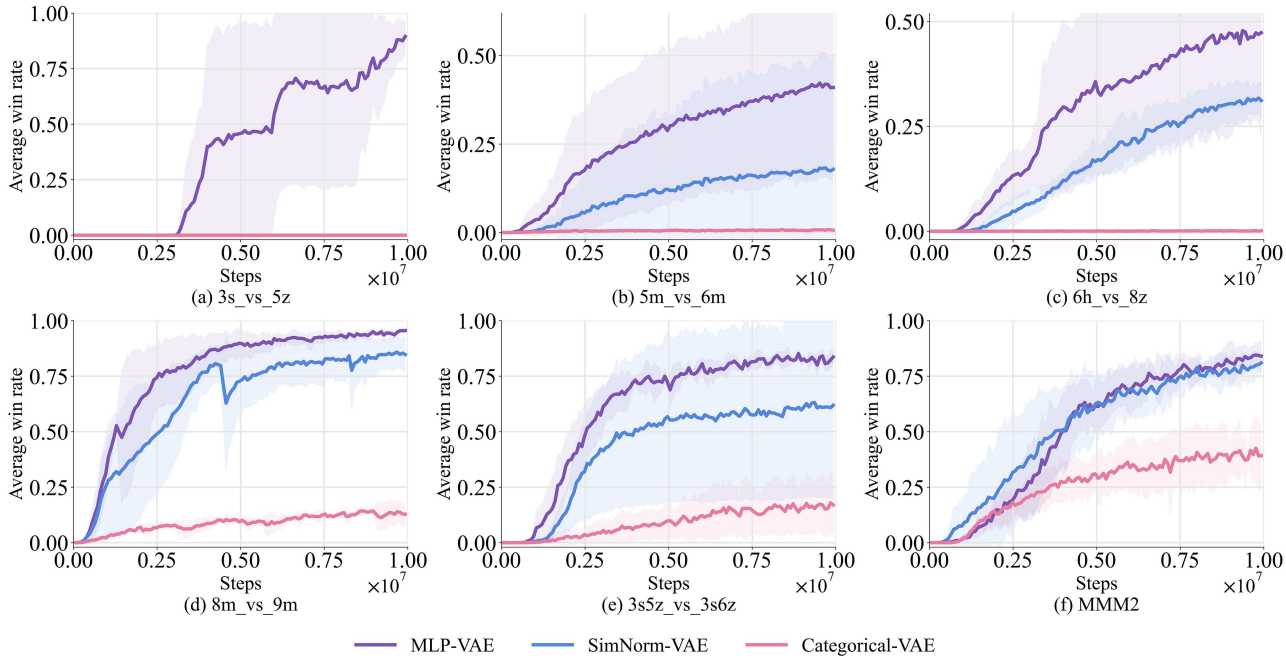


**Figure 2** (Color online) Average win rate comparison among SPMamba models using MLP-VAE, SimNorm-VAE, and categorical-VAE.

where $\mathcal{Z}_t$ is a distribution comprising multiple categories. The output $z_t$ is sampled from $\mathcal{Z}_t$ to represent the raw input $o_t$. Categorical-VAE is implemented in, e.g., DreamerV3 [18] and its various variants [19, 43]. Nevertheless, we consider multiple categorical variables may lose too much information when handling non-image observations encountered in e.g., SMAC. Furthermore, Dreamerv3 utilizes multiple categorical variables with $32 \times 32 = 1024$ dimensions, which are necessary for processing the image inputs in Atari, but it is overly redundant for compressing input features in SMAC. Considering the characteristics of non-image observations, TDMPC2 [32] introduces the SimNorm-VAE, which encodes raw inputs into multiple categorical variables without performing any sampling. By defining $d = M^2$ for any $M \in \mathbb{N}_+$ and $\tilde{z}_t = f_{\text{MLP}}(o_t) \doteq [x_1, x_2, ..., x_d]_t \in \mathbb{R}^{1\times d}$, the $\tilde{z}_t$ could be divided into $M$ partitions (groups) $[g_1, g_2, ..., g_M]_t$, and the $i$-th partition $g_i$ contains $M$ elements $[x_{(i-1)\times M+1}, x_{(i-1)\times M+2}, ..., x_{i\times M}]_t$. Thus, the SimNorm encoding is held as

$$z_t \doteq [\tilde{g}_1, \tilde{g}_2, ..., \tilde{g}_M]_t, \quad \tilde{g}_i = \frac{e^{x_{(i-1)\times M+j}/\tau}}{\sum_{j=1}^{j=M} e^{x_{(i-1)\times M+j}/\tau}}, \tag{12}$$

where $\tau$ is the temperature parameter that modulates the sparsity of $\tilde{g}_i$. SimNorm-VAE shows competitive performance in TDMPC2 in continuous control tasks in DMControl [44], Meta-World [45], ManiSkill2 [46], and MyoSuite [47] with non-image observations. However, the softmax operation in (12) constrains the model's capacity for feature learning in the latent state, which may lead to suboptimal performance in SMAC, where observation inputs exhibit rapid and discontinuous changes.

Due to the described limitations of the aforementioned encoder structures, we advocate MLP-VAE in (5a) tailored to the characteristics of agent observation inputs in the SMAC environment. Figure 2 shows the comparison among SPMamba models using MLP-VAE, categorical-VAE, and SimNorm-VAE. As the hidden state of the model is set to be 64 dimensions, the distribution $\mathcal{Z}_t$ of the categorical-VAE is set to include 8 categoricals which include 8 classes each. The scale $M$ of the SimNorm-VAE is also set to 8, while the temperature coefficient $\tau$ is set to 1.

According to the learning curves, the SPMamba with categorical-VAE performs the worst due to the excessive information loss from discrete encoding. The self-predicting loss curves in Figure 3 demonstrate that the features encoded by the categorical-VAE fail to correctly capture the self-predictive representations. The SPMamba with SimNorm-VAE outperforms the version with categorical-VAE in self-predicting, and this advantage is also reflected in the overall performance. Conversely, its performance is inferior to MLP-VAE due to the expressiveness limitations imposed by the softmax operation used in encoding. This highlights the superior encoding capability of the proposed MLP-VAE, which extracts the self-predictive features efficiently with greater accuracy and completeness, and thus leads to the strongest performance.

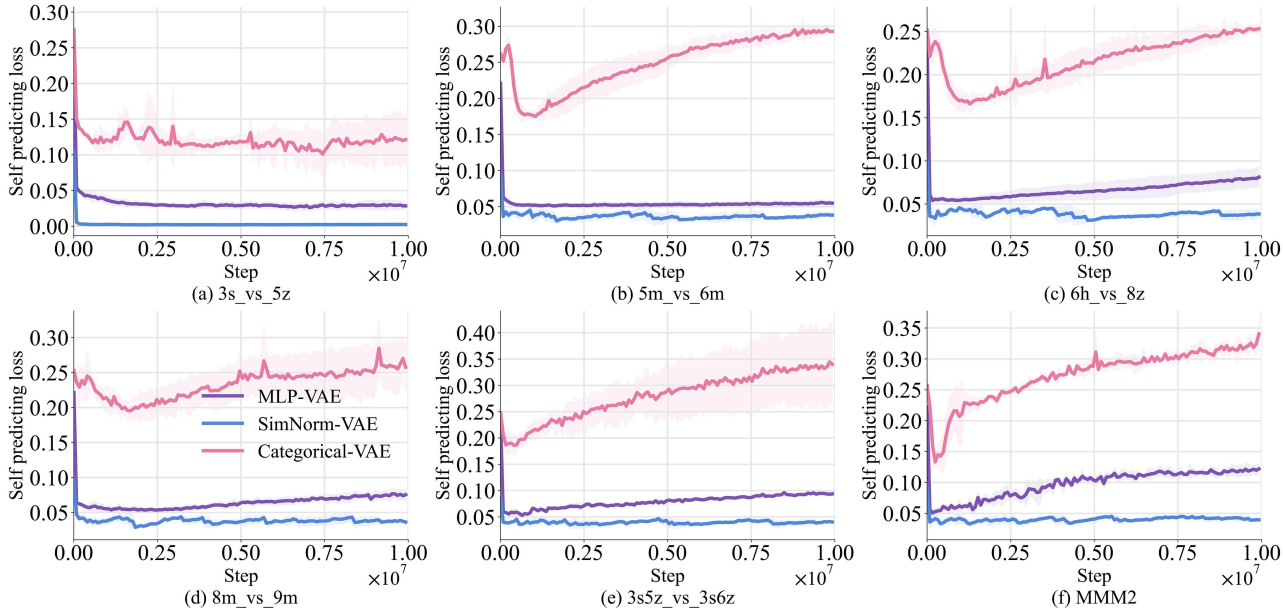


**Figure 3** (Color online) The self-predicting loss curves of MLP-VAE, categorical-VAE, and SimNorm-VAE.

#### 6.3.2 *Learning objectives*

The most widely-used learning objectives for VAE variants in RL include minimizing the self-predicting loss $L^{\text{pred}}(\phi)$ along with the representation loss $L^{\text{rep}}$ and the reconstruction loss $L^{\text{rec}}$:

$$L_t^{\text{rep}}(\phi) = \max\big(\mu, \text{KL}\big[\text{softmax}(z_t)||\text{sg}(\text{softmax}(\hat{z}_t))\big]\big), \tag{13a}$$

$$L_t^{\text{rec}}(\phi) = ||o_t - \text{REC}_\phi(z_t)||_2^2, \tag{13b}$$

where $\text{REC}_\phi$ is a decoder for reconstructing $o_t$, $L^{\text{rep}}(\phi)$ slightly aligns the output with the predicted results to make the encoded features more amenable to prediction, and $L^{\text{rec}}(\phi)$ ensures that the encoder output retains as much of the original input's features as possible. The total VAE loss is given by

$$L^{\text{VAE}}(\phi) = \alpha L^{\text{pred}}(\phi) + \beta L^{\text{rep}}(\phi) + \sigma L^{\text{rec}}(\phi), \tag{14}$$

where $\alpha, \beta, \sigma$ are weight coefficients. Nonetheless, we suppose that the primary objective of policy learning is to extract features highly relevant to decision-making. Preserving all input features may introduce irrelevant information, which could hinder the learning performance.

We compare the loss $L^{\text{SPMamba}}(\phi) = \alpha L^{\text{pred}}(\phi)$ designed for SPMamba against 2 alternative loss functions $L^{\text{rep}}(\phi) = \alpha L^{\text{pred}}(\phi) + \beta L^{\text{rep}}(\phi)$ and $L^{\text{VAE}}(\phi)$ in (14). The hyperparameters follow the same settings as in Dreamer [48], with only a global scaling adjustment, set to $\alpha = 0.1$, $\beta = 0.02$, and $\sigma = 0.2$. The results can be found in Figure 4. Clearly, incorporating the reconstruction objective significantly degrades policy performance, indicating that learning fully reconstructable features interferes with decision-making. Meanwhile, the representation loss has minimal impact on the performance.

### 6.4 Self-supervised learning structure

As an advanced self-supervised learning model, SPMamba should be designed carefully. We present two alternative self-predictive representation learning structures. The 'Predicting from $h_t$' configuration makes the transition decoder predict $\hat{z}_{t+1}$ from SPMamba's hidden state $\hat{z}_{t+1} = \text{DEC}_\phi(h_t, u_t)$, and the transition decoder in 'Raw output' configuration predicts $\hat{z}_{t+1}$ from $y_t$ that has not been normalized. The results in Figure 5 indicate that neither predicting from $h_t$ nor from the un-normalized $y_t$ is beneficial to policy learning. Since Mamba expands the dimensionality of $h_t$ by a factor of $\epsilon$ relative to the input-output, directly predicting $\hat{z}_{t+1}$ from $h_t$ may be inefficient. However, prediction based on un-normalized features performs slightly worse, as the un-normalized features exhibit numerical instability.

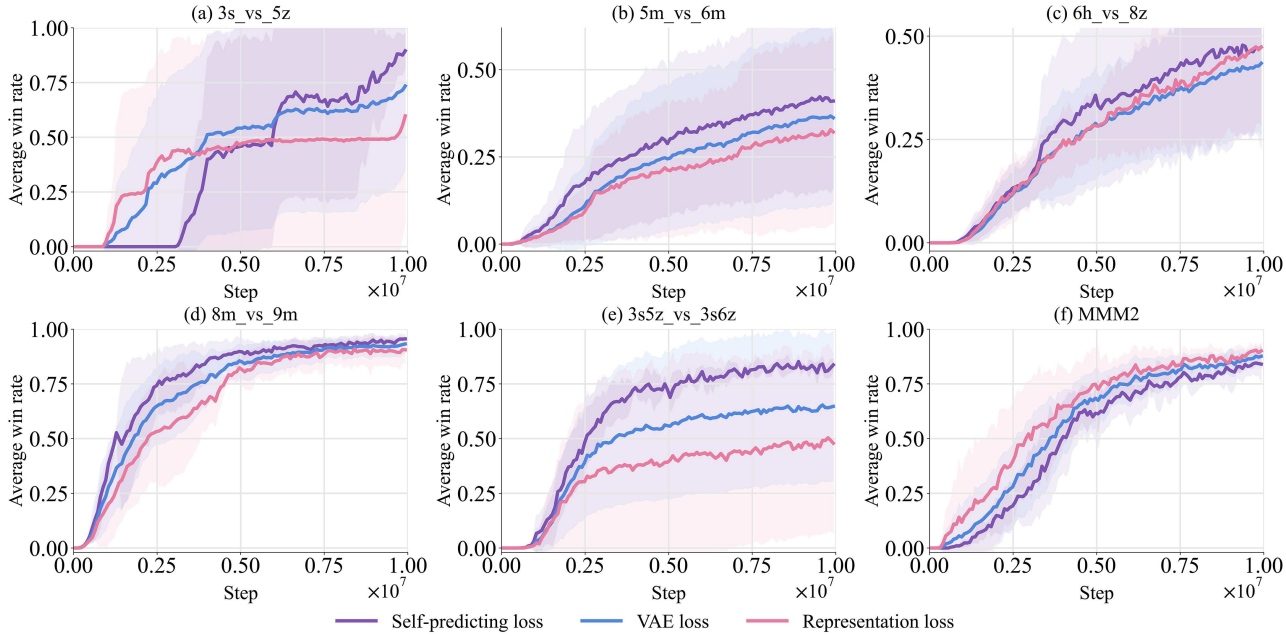


**Figure 4** (Color online) Average win rate comparison among the $L^{\text{SPMamba}}(\phi)$, $L^{\text{rep}}(\phi)$, and $L^{\text{VAE}}(\phi)$ losses.

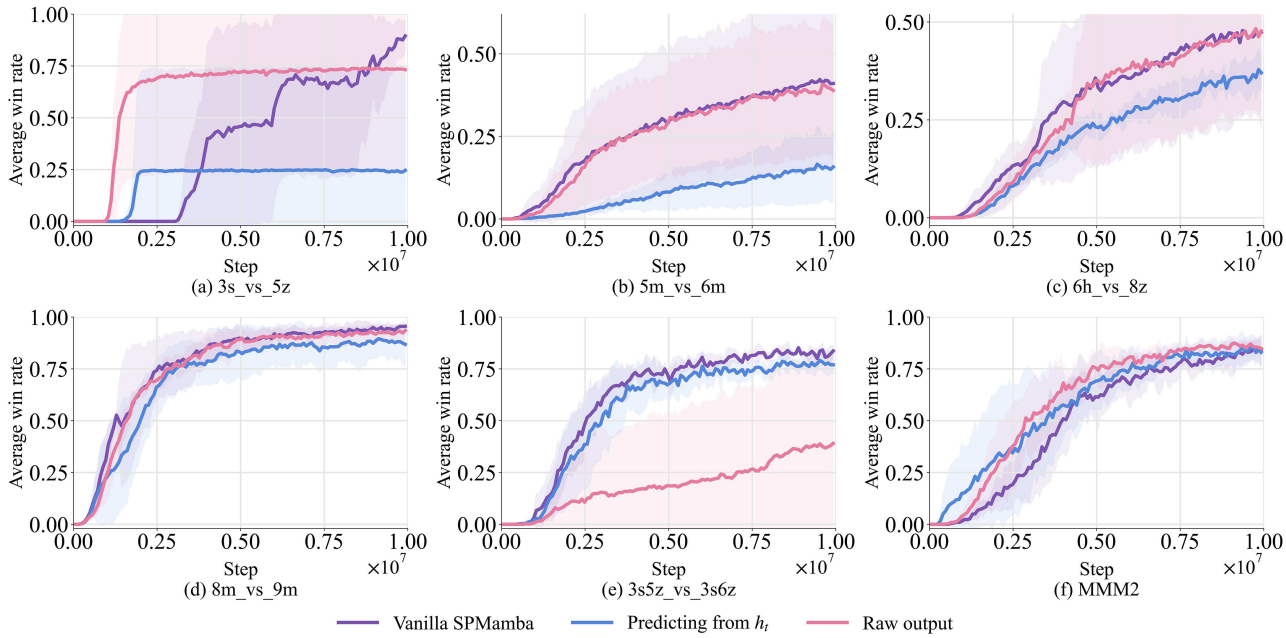


**Figure 5** (Color online) Average win rate comparison between vanilla SPMamba and 2 alternative structures.

## 7 Conclusion

In this work, we introduced SPMamba, a novel framework that integrates the Mamba model with self-predictive representation learning to improve decentralized multi-agent policy optimizations. SPMamba utilizes a Mamba model as the core part of processing agents' historical observations, and regresses the model's output on future observations in a latent space. SPMamba outperforms traditional RNN-based policies in extensive experiments without the need for task-specific hyperparameter tuning. The success of SPMamba highlights the potential of the Mamba model as a powerful module for sequential decision-making tasks, offering a scalable and efficient approach for handling complex multi-agent environments. Despite its demonstrated effectiveness, there are still several limitations. First, the self-predictive learning objective may be less effective in environments with highly stochastic dynamics, where future observations are inherently difficult to predict. Moreover, although Mamba offers

improved efficiency over Transformer-based models, its computational overhead remains non-negligible, potentially limiting scalability. Future research will investigate incorporating self-predictive learning based on global features within the CTDE paradigm, as well as exploring further optimizations to reduce runtime complexity.

**Acknowledgements** This work was supported by National Natural Science Foundation of China (Grant Nos. U23B2059, 61925303, 62088101).

**References**

1 Feng Z H, Xue R Q, Yuan L, et al. Multi-agent embodied AI: advances and futuredirections. Sci China Inf Sci, 2026, 69: 151202
2 Wang G, Liu W J, Li Y F, et al. Data-drivencontrol of network systems: accounting for communication adaptivityand security. Sci China Inf Sci, 2026, 69: 121201
3 Zhou M, Luo J, Villella J, et al. SMARTS: an open-source scalable multi-agent RL training school for autonomous driving. In: Proceedings of the Conference on Robot Learning, 2021. 264–285
4 Zheng Z Q, Wei C, Duan H B. UAV swarm air combat maneuver decision-making method based on multi-agent reinforcement learning and transferring. Sci China Inf Sci, 2022, 67: 180204
5 Zhou X H, Xiong J, Zhao H T, et al. Joint UAV trajectory and communication design with heterogeneous multi-agent reinforcement learning. Sci China Inf Sci, 2024, 67: 132302
6 Li Y F, Wang X, Sun J, et al. Data-driven consensus control of fully distributed event-triggered multi-agent systems. Sci China Inf Sci, 2023, 66: 152202
7 Liu W, Sun J, Wang G, et al. Data-driven resilient predictive control under denial-of-service. IEEE Trans Automat Contr, 2023, 68: 4722–4737
8 Zhou Z, Wang G, Sun J, et al. Efficient and robust time-optimal trajectory planning and control for agile quadrotor flight. IEEE Robot Autom Lett, 2023, 8: 7913–7920
9 Yuan L, Zhang Z, Li L, et al. A survey of progress on cooperative multi-agent reinforcement learning in open environment. 2023. ArXiv:2312.01058
10 Rashid T, Samvelyan M, Christian Schroeder D W, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning. J Mach Learn Res, 2020, 21: 7234–7284
11 Cao J H, Yuan L, Wang J H, et al. LINDA: multi-agent local information decomposition for awareness of teammates. Sci China Inf Sci, 2023, 66: 182101
12 Yu C, Velu A, Vinitsky E, et al. The surprising effectiveness of PPO in cooperative multi-agent games. In: Proceedings of Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022
13 Hu S, Zhu F, Chang X, et al. UPDeT: universal multi-agent RL via policy decoupling with transformers. In: Proceedings of International Conference on Learning Representations, 2021
14 Wen M, Kuba J, Lin R, et al. Multi-agent reinforcement learning is a sequence modeling problem. In: Proceedings of Advances in Neural Information Processing Systems, 2022. 16509–16521
15 Huang J, Xu Y, Wang Q, et al. Foundation models and intelligent decision-making: progress, challenges, and perspectives. Innovation, 2025, 6: 100948
16 Schwarzer M, Obando Ceron J S, Courville A, et al. Bigger, better, faster: human-level Atari with human-level efficiency. In: Proceedings of the 40th International Conference on Machine Learning, 2023. 30365–30380
17 Fang C, Stachenfeld K. Predictive auxiliary objectives in deep RL mimic learning in the brain. In: Proceedings of the 12th International Conference on Learning Representations, 2024
18 Hafner D, Pasukonis J, Ba J, et al. Mastering diverse domains through world models. 2023. ArXiv:2301.04104
19 Zhang W, Wang G, Sun J, et al. STORM: efficient stochastic transformer based world models for reinforcement learning. In: Proceedings of Advances in Neural Information Processing Systems, 2023. 27147–27166
20 Guan C, Chen F, Yuan L, et al. Efficient multi-agent communication via self-supervised information aggregation. In: Proceedings of Advances in Neural Information Processing Systems, 2022
21 Liang Q, Liu J, Jiang Z, et al. Limited information aggregation for collaborative driving in multi-agent autonomous vehicles. IEEE Robot Autom Lett, 2024, 9: 6624–6631
22 Song H, Feng M, Zhou W, et al. Ma2cl: masked attentive contrastive learning for multi-agent reinforcement learning. In: Proceedings of the 38th International Joint Conference on Artificial Intelligence, 2023
23 Gu A, Dao T. Mamba: linear-time sequence modeling with selective state spaces. 2023. ArXiv:2312.00752
24 Samvelyan M, Rashid T, Christian Schroeder D W, et al. The starcraft multi-agent challenge. 2019. ArXiv:1902.04043
25 Lowe R, Wu Y, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Proceedings of Advances in Neural Information Processing Systems, 2017
26 Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning. 2017. ArXiv:1706.05296
27 Gupta J K, Egorov M, Kochenderfer M. Cooperative multi-agent control using deep reinforcement learning. In: Proceedings of Autonomous Agents and Multiagent Systems, 2017. 66–83
28 Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2018

29 Wang J, Ye D, Lu Z. More centralized training, still decentralized execution: multi-agent conditional policy factorization. In: Proceedings of International Conference on Learning Representations, 2023

30 Xie Q, Dai Z, Hovy E, et al. Unsupervised data augmentation for consistency training. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 6256–6268

31 Henaff O. Data-efficient image recognition with contrastive predictive coding. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 4182–4192

32 Hansen N, Su H, Wang X. TD-MPC2: scalable, robust world models for continuous control. In: Proceedings of the 12th International Conference on Learning Representations, 2024

33 Schwarzer M, Anand A, Goel R, et al. Data-efficient reinforcement learning with self-predictive representations. 2020. ArXiv:2007.05929

34 Feng M, Zhou W, Yang Y, et al. Joint-predictive representations for multi-agent reinforcement learning. In: Proceedings of ICLR, 2023

35 Huh D, Mohapatra P. Representation learning for efficient deep multi-agent reinforcement learning. 2024. ArXiv:2406.02890

36 Lu C, Schroecker Y, Gu A, et al. Structured state space models for in-context reinforcement learning. In: Proceedings of Advances in Neural Information Processing Systems, 2023. 47016–47031

37 Chen L, Wang L, Dong H, et al. Introspective tips: large language model for in-context decision making. 2023. ArXiv:2305.11598

38 Jia X, Wang Q, Donat A, et al. MaIL: improving imitation learning with Mamba. 2024. ArXiv:2406.08234

39 Elfwing S, Uchibe E, Doya K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Netws, 2018, 107: 3–11

40 Ellis B, Cook J, Moalla S, et al. SMACv2: an improved benchmark for cooperative multi-agent reinforcement learning. In: Proceedings of Advances in Neural Information Processing Systems, 2023

41 Wang T, Gupta T, Mahajan A, et al. RODE: learning roles to decompose multi-agent tasks. In: Proceedings of International Conference on Learning Representations, 2021

42 Kuba J G, Chen R, Wen M, et al. Trust region policy optimisation in multi-agent reinforcement learning. In: Proceedings of International Conference on Learning Representations, 2022

43 Robine J, Höftmann M, Uelwer T, et al. Transformer-based world models are happy with 100k interactions. In: Proceedings of the 11th International Conference on Learning Representations, 2023

44 Tassa Y, Doron Y, Muldal A, et al. DeepMind control suite. 2018. ArXiv:1801.00690

45 Yu T, Quillen D, He Z, et al. Meta-world: a benchmark and evaluation for multi-task and meta reinforcement learning. In: Proceedings of the Conference on Robot Learning, 2020. 1094–1100

46 Gu J, Xiang F, Li X, et al. Maniskill2: a unified benchmark for generalizable manipulation skills. 2023. ArXiv:2302.04659

47 Caggiano V, Durandau G, Wang H, et al. Myochallenge 2022: learning contact-rich manipulation using a musculoskeletal hand. In: Proceedings of the NeurIPS 2022 Competitions Track, 2022. 233–250

48 Hafner D, Lillicrap T P, Norouzi M, et al. Mastering Atari with discrete world models. In: Proceedings of International Conference on Learning Representations, 2021