Multiagent Continual Coordination via Progressive Task Contextualization

Lei Yuan, Lihe Li, Ziqian Zhang, Fuxiang Zhang, Cong Guan, and Yang Yu

***Abstract*— Cooperative multiagent reinforcement learning (MARL) has attracted significant attention and has the potential for many real-world applications. Previous arts mainly focus on facilitating the coordination ability from different aspects (e.g., nonstationarity and credit assignment) in single-task or multitask scenarios, ignoring the stream of tasks that appear in a continual manner. This ignorance makes the continual coordination an unexplored territory, neither in problem formulation nor efficient algorithms designed. Toward tackling the mentioned issue, this article proposes an approach, multiagent continual coordination via progressive task contextualization (MACPro). The key point lies in obtaining a factorized policy, using shared feature extraction layers but separated independent task heads, each specializing in a specific class of tasks. The task heads can be progressively expanded based on the learned task contextualization. Moreover, to cater to the popular centralized training with decentralized execution (CTDE) paradigm in MARL, each agent learns to predict and adopt the most relevant policy head based on local information in a decentralized manner. We show in multiple multiagent benchmarks that existing continual learning methods fail, while MACPro is able to achieve close-to-optimal performance. More results also disclose the effectiveness of MACPro from multiple aspects, such as high generalization ability.**

***Index Terms*— Continual learning, cooperation and coordination, multiagent system, reinforcement learning.**

I. Introduction

Cooperative multiagent reinforcement learning (MARL) has attracted prominent attention in recent years [1] and achieved great progress in multiple aspects, such as path finding [2], active voltage control [3], and dynamic algorithm configuration [4]. Among the multitudinous methods, researchers, on the one hand, focus on facilitating coordination ability via solving specific challenges, including nonstationarity [5], credit assignment [6], and scalability [7]. Other works, on the other hand, investigate the cooperative MARL from multiple aspects, such as efficient communication [8], zero-shot coordination (ZSC) [9], and policy robustness [10]. A lot of methods emerge as promising solutions for different scenarios, including policy-based ones [11], [12], value-based series [13], [14], and many other variants, showing remarkable coordination ability in a wide range of tasks, such as StarCraft Multiagent Challenge (SMAC) [15]. Despite the great success, the mainstream cooperative MARL methods are still restricted to being trained in a single task or multiple tasks simultaneously, assuming that the agents have access to data from all tasks at all times.

Unfortunately, in some real-world applications, this assumption is unrealistic. On the one hand, for multiagent systems deployed to solve real-world tasks, the dynamic or the goal of the task may change over time [16], due to the changing nature of the real-world environment, or different man-made goals in different periods. In such real-world scenarios, the agents are required to maintain the ability to solve all seen tasks but have no access to previous tasks since the environment has changed. On the other hand, for agents that are required to coordinate with diverse teammates [17], particularly humans in human-AI coordination scenarios, they need to train with different teammates to improve their coordination ability. When they encounter new unseen teammates, learning from scratch with all seen teammates (especially human participants) would be unrealistic since the previous teammates could be unavailable. In this scenario, ego agents should continually adapt to new teammates while retaining knowledge from previous interactions. In these real-world applications, simply applying the mainstream cooperative MARL methods to train the agents will cause catastrophic forgetting, which is unacceptable.

Continual reinforcement learning plays a promising role in the mentioned problem [16], where the agent aims to avoid catastrophic forgetting, as well as enable knowledge transfer to new tasks (also known as stability-plasticity dilemma [18]), while maintaining scalable to a large number of tasks. Multiple approaches have been proposed to address one or more of these challenges, including regularization-based ones [19], [20], experience maintaining techniques [21], [22], task structure sharing categories [23], [24], and so on. However, the multiagent setting is much more complex than the single-agent one, as the interaction among agents might cause additional considerations [25]. Also, coordinating with multiple teammates is proved to be tough [17]. Previous works model this problem as multitask [9] or just unimodal coordination among

Manuscript received 8 May 2023; revised 22 November 2023 and 29 January 2024; accepted 18 April 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0107200, and in part by the National Science Foundation of China under Grant 61921006 and Grant 62022039. *(Corresponding author: Yang Yu.)*

Lei Yuan, Fuxiang Zhang, and Yang Yu are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China, also with the School of Artificial Intelligence, Nanjing University, Nanjing 210023, China, and also with Polixir Technologies, Nanjing 211106, China (e-mail: yuanl@lamda.nju.edu.cn; zhangfx@lamda.nju.edu.cn; yuy@nju.edu.cn).

Lihe Li, Ziqian Zhang, and Cong Guan are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China, and also with the School of Artificial Intelligence, Nanjing University, Nanjing 210023, China (e-mail: lilhzq76@gmail.com; 191240076zzq@gmail.com; gaunc@lamda.nju.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2024.3394513

teammates [26]. In light of the significance and ubiquity of cooperative MARL, it is, thus, imperative to consider the continual coordination in both the problem formulation and the algorithm design to tackle this issue.

In this work, we develop such a continual coordination framework in cooperative MARL where tasks appear sequentially. Concretely, we first develop a multiagent task context extraction module, where information of each state in a specific task is extracted and integrated by a product-of-expert (POE) mechanism into a latent space to capture the task dynamic information, and a contrastive regularizer is further applied to optimize the learned representation, with which similar task representations are pulled together, while dissimilar ones are pushed apart. Next, we apply an expandable multihead policy architecture whose separate independent heads are synchronously expanded with the newly instantiated context, along with a carefully designed shared feature extraction module. Finally, considering the popular centralized training with decentralized execution (CTDE) paradigm in mainstream cooperative MARL, we leverage the local information of each agent to approximate the policy head selection process via policy distillation in the centralized training process, with which agents can select the optimal ones to coordinate with other teammates in a decentralized manner.

For the evaluation of the proposed approach, multiagent continual coordination via progressive task contextualization (MACPro), we conduct extensive experiments on various cooperative multiagent benchmarks in the continual setting, including level-based foraging (LBF) [27], predator-prey (PP) [11], and the SMAC benchmark [28], and compare MACPro against previous approaches, strong baselines, and ablations. Experimental results show that MACPro considerably improves upon existing methods. More results demonstrate its high generalization ability and its potential to be integrated with different value-based methods to enhance its continual learning ability. Visualization experiments provide additional insight into how MACPro works.

II. Related Work

A. Cooperative MARL

Many real-world problems are made up of multiple interactive agents, which could usually be modeled as an MARL problem [25], [29]. Furthermore, when the agents hold a shared goal, this problem refers to cooperative MARL [1], showing great progress in diverse domains such as path finding [2], active voltage control [3], and dynamic algorithm configuration [4] (removed). Many methods are proposed to facilitate coordination among agents, including policy-based ones (e.g., MADDPG [11], MAPPO [12], and FD-MARL [30]), value-based series, such as VDN [13], QMIX [14], and SMIX (λ) [31], or other techniques, such as transformer [32]; these approaches have demonstrated remarkable coordination ability in a wide range of benchmarks (e.g., SMAC [28], Hanabi [12], and GRF [32]). Besides the mentioned approaches and the corresponding variants, many other methods are also proposed to investigate the cooperative MARL, including efficient communication [8] [33] to relieve the partial observability caused by decentralized policy execution, policy deployment in an offline manner [34], model learning in MARL [35], policy robustness when some perturbations exist [10], and training paradigm such as CTDE [36], coordination between heterogeneous agents [37], and ad hoc teamwork [17]. With all these well-developed techniques, some researchers apply MARL methods to solve challenging real-world tasks, including multiagent navigation [38], [39] and path finding [2], active voltage control [3], backstepping control [40], dynamic algorithm configuration [4], and multirobot control [41], [42], showcasing the great potential of MARL in real-world applications.

Despite this progress, the majority of current approaches either focus on training the MARL policy on a single task or the multitask setting where all tasks appear simultaneously, lacking attention to the continual coordination problem. In these methods, MATTAR [43] assumes that there are some basic tasks, training with which can accelerate the training process in other similar tasks, and develops a multiagent multitask training framework. TrajeDi [9] and some variants (or improved versions), such as MAZE [44], concentrate on coordinating with different teammates or even unseen ones like a human; these methods are also under the assumption that we can access all the training tasks all the time. Nekoei et al. [26] introduce a multiagent learning testbed that supports both zero- and few-shot settings based on Hanabi, but it only considers the unimodal coordination among tasks, and the experimental results demonstrate that methods such as VDN [13] trained in the proposed testbed can coordinate well with unseen agents, without any additional assumptions made by previous works.

B. Continual Reinforcement Learning

Continual learning is conceptually related to incremental learning and lifelong learning as they all assume that tasks or samples are presented in a sequential manner [18], [45]. For continual reinforcement learning [16], EWC [46] learns new Q-functions by regularizing the l_2 distance between the optimal weights of the new task and previous ones. It requires additional supervision information such as task changes to update its objective and then selects a specific Q-function head and a task-specific exploration schedule for different tasks. CLEAR [47] is a task-agnostic method that does not require task information during the continual learning process and leverages big experience replay buffers to prevent forgetting. Coreset [48] prevents catastrophic forgetting by choosing and storing a significantly smaller subset of the previous task's data, which is used to rehearse the model during or after finetuning. Some other works, such as HyperCRL [49], [50], utilize a learned world model to promote continual learning efficiency. Considering the scalability issue along with the task number, CN-DPM [51], DaCoRL [52], and LLIRL [53] decompose the whole task space into several subsets of the data (tasks) and then utilize techniques such as Dirichlet process mixture or Chinese restaurant process to expand the neural network for efficient continual supervised learning and reinforcement learning tasks, respectively. OWL [23] is a recently proposed approach that learns a

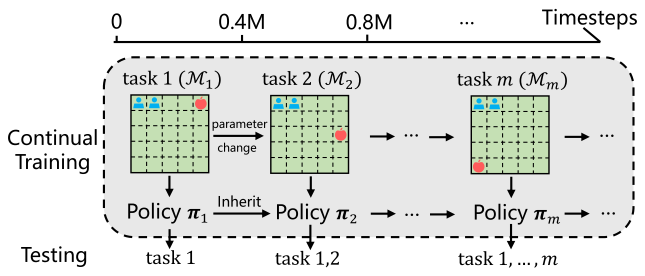


Fig. 1. Example of multiagent continual coordination, where tasks (e.g., the position of food changes in the LBF [27]) change along with the timeline. We, thus, need to train a policy $\boldsymbol{\pi}_m$ to solve the concurrent task and maintain the knowledge of previous tasks (i.e., avoid catastrophic forgetting).

multihead architecture and achieves high learning efficiency, and CSP [24] incrementally builds a subspace of policies for training a reinforcement learning agent on a sequence of tasks. These works are capable of solving dynamic tasks such as graph representation learning [54], community detection [55], identification of critical drones [56], and so on. Other researchers also design benchmarks such as continual world [57], or baselines [58] to verify the effectiveness of different methods in single-agent reinforcement learning. Nekoei et al. [26] investigate whether agents can coordinate with unseen agents by introducing a multiagent learning testbed based on Hanabi, only considering the unimodal coordination among tasks. Our work takes a further step in this direction for this problem.

III. Problem Formulation

This work considers a cooperative MARL problem under partial observation, which can be formalized as a Dec-POMDP [59], with tuple $\mathcal{M} = \langle N, \mathcal{S}, \mathcal{A}, \Omega, P, O, R, \gamma \rangle$, where $N = \{1, \dots, n\}$, $\mathcal{S}$, $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^n$, and Ω are the set of agents, states, joint actions, and local observation, respectively. $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ stands for the transition probability function; $O : \mathcal{S} \times N \to \Omega$ and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ are the corresponding observation function and reward function; and $\gamma \in [0, 1)$ is the discounted factor. Multiple interactive agents in a Dec-POMDP coordinate with teammates to complete a task under a shared reward R; at each time step, agent i receives the local observation $o^i = O(s, i)$ and outputs the action $a^i \in \mathcal{A}^i$. The formal objective of the agents is to maximize the expected cumulative discounted reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \boldsymbol{a}_t)]$.

In this work, we focus on a continual coordination problem where agents in a team are exposed to a sequence of (infinite) tasks $\mathcal{Y} = (\mathcal{M}_1, \dots, \mathcal{M}_m, \dots)$. Each task involves a sequential decision-making problem and can be formulated as a Dec-POMDP $\mathcal{M}_m = \langle N_m, \mathcal{S}_m, \mathcal{A}_m, \Omega_m, P_m, O_m, R_m, \gamma \rangle$, as shown in Fig. 1. These agents are continually evaluated on all previous tasks (but cannot be trained with these tasks) and the present task. Therefore, the agent's policy needs to be transferred to new tasks while maintaining the ability to perform previous tasks. Concretely, agents that have learned M tasks are expected to maximize the MARL objective for each task in $\mathcal{Y}_M = \{\mathcal{M}_1, \dots, \mathcal{M}_M\}$. We consider the setting where task boundaries are known during the centralized training phase. During the decentralized execution phase, agents cannot access global but only local information to finish the tasks sampled from $\mathcal{Y}_M$ in a decentralized manner.

IV. Method

In this section, we will describe the detailed design of our proposed method, MACPro. First, we propose a novel training paradigm, including a shared feature extraction part and an adaptive policy heads expansion module based on the learned contexts [see Fig. 2(a)]. Next, we design an efficient multiagent task contextualization learning module to capture the uniqueness of each emerging task [see Fig. 2(b)]. Finally, considering the CTDE property in mainstream cooperative MARL, we train each agent to utilize its local information to approximate the actual task head [see Fig. 2(c)].

A. Multiagent Task Contextualization Learning

In continual reinforcement learning where tasks keep altering sequentially, it is crucial to capture the unique context of each emerging new task. However, the behavioral descriptor of the multiagent task is much more complex than the single-agent setting due to the interactions among agents [1]. Thus, this section aims to tackle this issue by developing an efficient multiagent task contextualization learning module.

Specifically, considering a trajectory $\tau = (s_0, \dots, s_T)$ with horizon T rollout by any policies, we utilize a global trajectory encoder g_θ parameterized by θ to encode τ into a latent space. Concretely, the trajectory representation is represented by a multivariate Gaussian distribution $\mathcal{N}(\mu_\theta(\tau), \sigma_\theta^2(\tau))$ whose parameters are computed by $g_\theta(\tau)$. As the trajectory horizon T may alter for different tasks (e.g., 3m and 5m in SMAC [28]), we here apply a transformer [60] architecture (see Appendix B) to extract feature from each trajectory; thus, the latent context of a whole trajectory can be represented as T Gaussian distributions $\mathcal{N}(\mu_0, \sigma_0^2), \dots, \mathcal{N}(\mu_T, \sigma_T^2)$, where $\mathcal{N}(\mu_i, \sigma_i^2)$ stands for the ith essential parts of the trajectory. Next, considering the importance of different states in a trajectory, we apply the POE technique [61] to acquire the joint representation of a trajectory, which is also a Gaussian distribution $\mathcal{N}(\mu_\theta(\tau), \sigma_\theta^2(\tau))$, where

$$\mu_\theta(\tau) = \left(\sum_{t=0}^{T} \mu_t (\sigma_t^2)^{-1}\right)\left(\sum_{t=0}^{T} (\sigma_t^2)^{-1}\right)^{-1}$$
$$\sigma_\theta^2(\tau) = \left(\sum_{t=0}^{T} (\sigma_t^2)^{-1}\right)^{-1}. \tag{1}$$

The detailed derivative process between the joint distribution and each single one can be seen in [62].

The previous part can obtain representation for each trajectory. Nevertheless, the learned representation lacks any dynamic information about a specific multiagent task. As the difference between any dynamic model lies in transition and reward functions [63], we here apply a loss function to force the learned trajectory representation to capture the dynamic information of each task. Specifically, we learn a context-aware forward model h including three predictors:

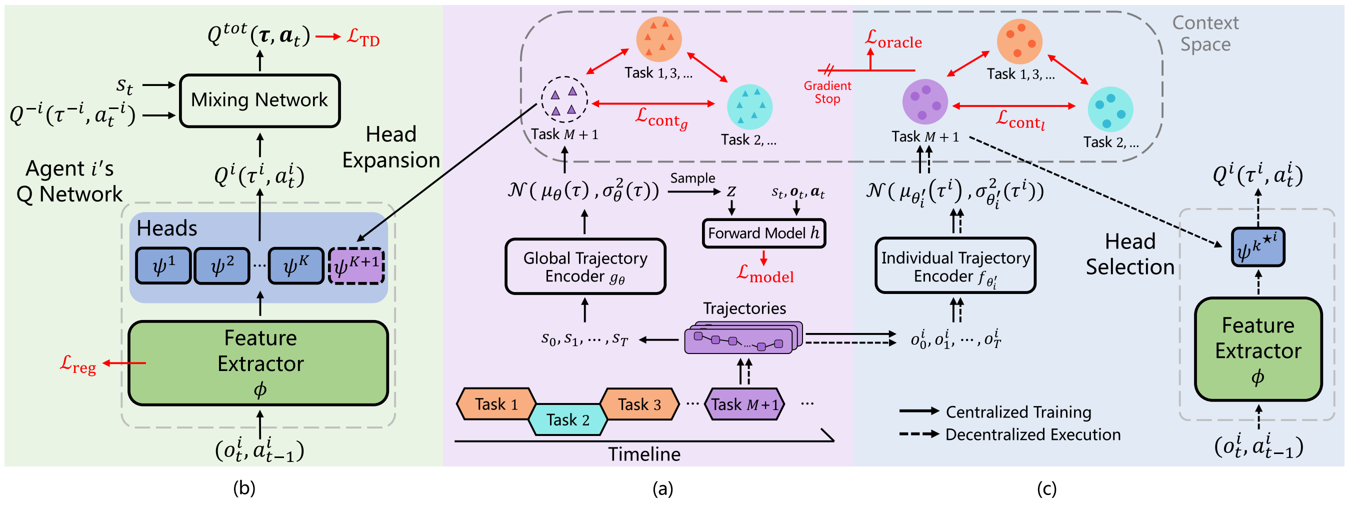


Fig. 2. Overall framework of MACPro. (a) We design an efficient multiagent task contextualization learning module to capture the uniqueness of each emerging task (dynamic network expansion). (b) Training paradigm, including a shared feature extraction part and an adaptive policy heads expansion module based on the learned contexts (task contextualization learning). (c) Each agent utilizes its local information to approximate the actual task head in a decentralized way (decentralized task approximation and execution).

h_s, h_o, and h_r, which are responsible for predicting the next state, local observations, and reward given the current state, local observations, actions, and task contextualization, respectively,

$$\mathcal{L}_{\text{model}} = \mathbb{E}_{\tau\in\mathcal{D}'}\Bigg[\sum_{t=0}^{T}\|h_s[s_t, \boldsymbol{o}_t, \boldsymbol{a}_t, z] - s_{t+1}\|_2^2 + \|h_o[s_t, \boldsymbol{o}_t, \boldsymbol{a}_t, z] - \boldsymbol{o}_{t+1}\|_2^2 + (h_r[s_t, \boldsymbol{o}_t, \boldsymbol{a}_t, z] - r_t)^2\Bigg] \tag{2}$$

where z is the task contextualization sampled from the joint task distribution and $\mathcal{D}'$ is the replay buffer for task contextualization learning, which stores a small number of trajectories for each task. However, as there are tasks with different correlations, the mentioned optimization object $\mathcal{L}_{\text{model}}$ might be insufficient for differentiable context acquisition. Therefore, we apply another auxiliary contrastive loss [64] by pulling together semantically similar data points (positive data pairs) while pushing apart the dissimilar ones (negative data pairs)

$$\mathcal{L}_{\text{cont}_g} = \mathbb{E}_{\tau_j,\tau_k\in\mathcal{D}'}\Bigg[\mathbf{1}\{y_j = y_k\}D_J(g_\theta(\tau_j)\|g_\theta(\tau_k)) + \mathbf{1}\{y_j \neq y_k\}\frac{1}{D_J(g_\theta(\tau_j)\|g_\theta(\tau_k)) + \varepsilon}\Bigg] \tag{3}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function, y_j and y_k are the label(s) of the task(s) from which τ_j and τ_k are sampled, respectively, and ε is a small positive constant added to avoid division by zero. $D_J(P\|Q) = D_{\text{KL}}(P\|Q) + D_{\text{KL}}(Q\|P)$ is Jeffrey's divergence [65] used to measure the distance between two distributions, and D_{KL} denotes the Kullback–Leibler divergence. We choose Jeffrey's divergence because it is symmetric and easy to calculate. Thus, the overall loss term is defined as follows:

$$\mathcal{L}_{\text{context}} = \mathcal{L}_{\text{model}} + \alpha_{\text{cont}_g}\mathcal{L}_{\text{cont}_g} \tag{4}$$

where α_{cont_g} is the coefficient balancing the loss terms.

B. Adaptive Dynamic Network Expansion

With the previously learned global trajectory encoder g_θ, we can obtain a unique contextualization for each task. Now, this section comes to the design of a context-based continual learning mechanism, which incrementally clusters a stream of stationary tasks in the dynamic environment into a series of contexts and optimizes for the optimal policy head from the expandable multihead neural network.

Formally, for multiple tasks appearing sequentially, we design a policy network consisting of a shared feature extractor ϕ with multiple layers of neural network (agent index is omitted for simplicity), which can promote knowledge sharing among different tasks. Furthermore, as there may be some multimodal tasks, a single head for all tasks could make the policy overfit some specific tasks. One way to solve this problem is to learn a customized head for each task like OWL [23]. However, this solution has poor scalability as the number of heads increases linearly over the number of tasks that could be infinitely many.

Thus, we develop an adaptive network expansion paradigm based on the similarity between task contextualizations. Specifically, we assume that the agents have already learned M tasks and have K policy heads $\{\psi^k\}_{k=1}^K$ so far ($K \leq M$). For each head, we store bs trajectories in buffer $\mathcal{D}'$, and we use g_θ to obtain the corresponding task contextualizations with mean values $\{\{\mu_k^j\}_{j=1}^{bs}\}_{k=1}^K$.

When encountering a new task ($M + 1$), we first utilize the feature extractors ϕ and all the existing heads $\{\psi^k\}_{k=1}^K$ to derive a set of behavior policies $\{\boldsymbol{\pi}_k\}_{k=1}^K$ to collect bs trajectories each on task ($M + 1$), denoted as $\{\{\tau_k^j\}_{j=1}^{bs}\}_{k=1}^K$. Next, we use g_θ to derive the mean values $\{\{\mu_k^{\prime j}\}_{j=1}^{bs}\}_{k=1}^K$ of their contextualizations and calculate the similarities between the existing mean values $\{\{\mu_k^j\}_{j=1}^{bs}\}_{k=1}^K$ as follows:

$$l = (l_1, \ldots, l_K), \quad l' = (l_1', \ldots, l_K')$$

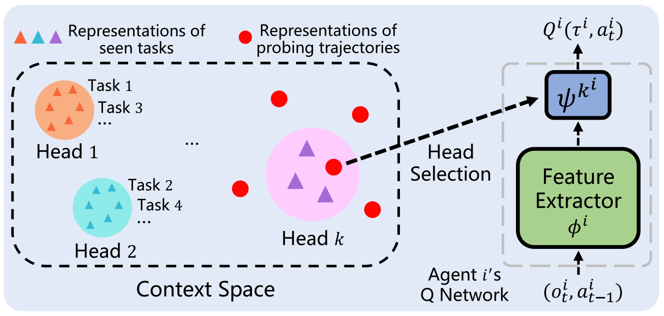


Fig. 3. Framework of decentralized task approximation and execution.

where

$$l_k = \frac{1}{bs}\sum_{j=1}^{bs}\left\|\mu_k^j - \frac{1}{bs}\sum_{i=1}^{bs}\mu_k^i\right\|_2$$
$$l'_k = \frac{1}{bs}\sum_{j=1}^{bs}\left\|\mu_k^{\prime j} - \frac{1}{bs}\sum_{i=1}^{bs}\mu_k^i\right\|_2, \quad k=1,\ldots,K \tag{5}$$

where l is the vector describing the dispersion of the K existing contextualizations and l' is the vector describing the distance between the K new contextualizations and the existing ones. Let $k_* = \arg\min_{1\le k\le K} l'_k$ such that the k_*^{th} pair of existing and new contextualizations are closest among all K pairs. With an adjustable threshold λ_{new}, if $l'_{k_*} \le \lambda_{\text{new}} l_{k_*}$, indicating task $(M+1)$ is similar to the task(s) that head ψ^{k_*} takes charge, we, thus, merge it to this/these learned task(s) and use the unified head ψ^{k_*} for them. Otherwise, none of the learned tasks is similar to the new one; a new head ψ^{K+1} is created. This phase processes along with the task sequence, enjoying high scalability and learning efficiency.

The previous part solves the head expansion issue, while a single shared feature extractor may inevitably cause forgetting. We here apply an l_2-regularizer to relieve this issue by constraining the parameters of the shared part that does not change too drastically when learning task $(M+1)$

$$\mathcal{L}_{\text{reg}} = \sum_{i=1}^{n}\left\|\phi_i - \phi_i^M\right\|_2 \tag{6}$$

where ϕ_i^M is the saved snapshot of agent i's feature extractor ϕ_i after training on task M. We choose l_2-regularizer because it is straightforward and easy to calculate. As we can apply MACPro to any value-based methods, we, thus, obtain the temporal difference error $\mathcal{L}_{\text{TD}}$ as $[r + \gamma \max_{\boldsymbol{a}'} Q^{\text{tot}}(s', \boldsymbol{a}'; \theta^-) - Q^{\text{tot}}(s, \boldsymbol{a}; \theta)]^2$, where θ^- are parameters of a periodically updated target network. The overall loss term of training agents' policies is defined as

$$\mathcal{L}_{\text{RL}} = \mathcal{L}_{\text{TD}} + \alpha_{\text{reg}}\mathcal{L}_{\text{reg}} \tag{7}$$

where α_{reg} is the coefficient balancing the two loss terms.

C. Decentralized Task Approximation

Although we have obtained an efficient continual learning approach for any tasks that appear in a sequential way, it is still far away from the MARL setting, as it requires the trajectory of global states to obtain the task representation, while agents in an MARL system can only acquire its local information.

Toward tackling the issue, we here develop a distillation solution. Concretely, for agent i with its local trajectory history $\tau^i = (o_0^i, \ldots, o_T^i)$, we design a local trajectory encoder $f_{\theta'_i}$ that is similar to the global trajectory encoder g_θ. $f_{\theta'_i}$ takes τ^i as input and outputs $\mathcal{N}(\mu_{\theta'_i}(\tau^i), \sigma^2_{\theta'_i}(\tau^i))$. We, thus, optimize $f_{\theta'_i}$ by minimizing Jeffrey's divergence between the distributions, as defined in the following equation:

$$\mathcal{L}_{\text{oracle}} = \mathbb{E}_{(\tau,\tau^i)\in\mathcal{D}'}\left[D_J\left(\overline{g_\theta(\tau)}\,\|\, f_{\theta'_i}(\tau^i)\right)\right] \tag{8}$$

where $\overline{\cdot}$ denotes gradient stop, and τ and τ^i stand for the global and local trajectories of the same task, respectively. To accelerate this learning process and make it consistent with task contextualization learning, we design a local auxiliary contrastive loss

$$\mathcal{L}_{\text{cont}_l} = \mathbb{E}_{\tau_j,\tau_k\in\mathcal{D}'}\Bigg[\mathbf{1}\{y_j = y_k\}D_J(f_{\theta'_i}(\tau_j^i)\,\|\, f_{\theta'_i}(\tau_k^i)) + \mathbf{1}\{y_j \ne y_k\}\frac{1}{D_J(f_{\theta'_i}(\tau_j^i)\,\|\, f_{\theta'_i}(\tau_k^i)) + \epsilon}\Bigg]. \tag{9}$$

The overall loss term of this part is

$$\mathcal{L}_{\text{approx}} = \mathcal{L}_{\text{oracle}} + \alpha_{\text{cont}_l}\mathcal{L}_{\text{cont}_l} \tag{10}$$

where α_{cont_l} is the coefficient balancing the loss terms.

During the decentralized execution phase (Fig. 3), agents first roll out P episodes to probe the environment. Concretely, for each probing episode $p (p = 1, \ldots, P)$, agents randomly choose one policy head to interact with the evaluating task to collect trajectory τ_p^i and calculate the mean value $\mu_{\theta'_i}(\tau_p^i)$ of the trajectory representation $f_{\theta'_i}(\tau_p^i)$. Finally, each agent i selects the most optimal task head by comparing the distance with the K existing task contextualization as follows:

$$k^{\star i} = \operatorname*{argmin}_{1\le k\le K}\min_{1\le p\le P}\left\|\mu_{\theta'_i}(\tau_p^i) - \frac{1}{bs}\sum_{j=1}^{bs}\mu_k^j\right\|_2 \tag{11}$$

and uses head $\psi^{k^{\star i}}$ with feature extractor ϕ for testing.

V. Experimental Evaluation

In this section, we design extensive experiments for the following questions:

1) Can our approach MACPro achieve high continual ability compared to other baselines in different scenarios, and how each component influences its performance (see Section V-B)?
2) What task representation is learned by our approach, and how does it influence the continual learning ability (see Section V-C)?
3) Can MACPro be integrated into multiple cooperative MARL methods, and how does each hyperparameter influence its performance (see Section V-D)?

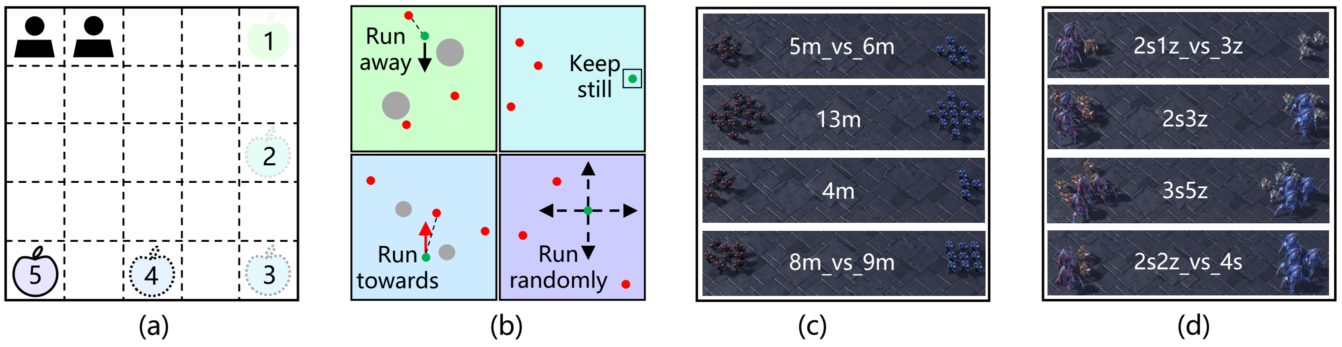


Fig. 4. Experimental environments used in this article. (a) LBF, where the position of the food changes in different tasks as indicated by the number on the food. (b) PP, where, in different tasks, the position of landmarks, the agents' acceleration, maximum speed, positions, and the fixed heuristic policies that the prey uses are different. (c) and (d) Marines and SZ from SMAC involve various numbers and types of battle agents, respectively.

A. Environments and Baselines

For the evaluation benchmarks, we select four multiagent environments (see Fig. 4), where LBF [27] is a cooperative grid world game with agents that are rewarded if they concurrently navigate to the food and collect it. The position of the food changes in different tasks as indicated by the number of the food. PP [11] is another popular benchmark where agents (predators) need to chase the adversary agent (prey) and encounter it to win the game. In different tasks, the position of landmarks, the agents' acceleration, maximum speed, positions, and the fixed heuristic policies that the prey uses are different. Marines and Stalkers and Zealots (SZ) from SMAC [28] involve various numbers of the agents.

To evaluate if MACPro can achieve good performance on these benchmarks when different tasks appear continually, we apply it to a popular valued-based method QMIX [14]. Compared baselines include Finetuning, which directly tunes the learned policy on the current task; EWC [46], a regularization-based method that constrains the whole agent network from dramatic change; and Coreset [48], which uses a shared replay buffer over all the tasks so that data on old tasks can rehearse the agents during fine-tuning on the new task. Also, OWL [23] is included as it is similar to our work but applies the bandit algorithm for head selection. To further study the head selection process, we design Random, with MACPro selecting a head randomly during testing, and Oracle, where MACPro's head selection is based on the ground-truth head information. We run each method for five distinct random seeds for a fair comparison. Also, for the combination of different methods and tasks, we test 32 episodes and report the average performance. More details about benchmarks and baselines used in this article can be seen in Appendix A.

B. Competitive Results and Ablations

1) Continual Learning Ability Comparison: At first glance, we compare MACPro against the mentioned baselines to investigate the continual learning ability, as shown in Fig. 5. We can find that Finetuning achieves the most inferior performance in different benchmarks, showing that a conventional reinforcement learning training paradigm is improper for continual learning scenarios. Other successful approaches for single-agent continual learning, such as Coreset, EWC, and OWL, also suffer from performance degradation in the involved benchmarks, demonstrating the necessity of specific consideration for MARL settings. The Oracle baseline, where we give all the ground-truth task identification when testing, can be seen as an upper bound of performance on the related benchmarks, acquiring superiority over all baselines in all benchmarks, demonstrating that multihead architecture can solve the multimodal tasks, while conventional approaches fail. Our approach, MACPro, obtains comparable performance to Oracle, indicating the efficiency of all the designed modules. Random, which selects a head randomly when testing, suffers from terrible performance degradation compared with MACPro and Oracle, showing that the success of MACPro is due to the appropriate head selection mechanism but not a larger network with multiple heads.

Furthermore, we display the performance on every single task in PP in Fig. 6; we can find that baselines, Finetuning, EWC, and Coreset, all suffer from performance degradation on one task after training on it, i.e., catastrophic forgetting, demonstrating the necessity of specific consideration for MARL continual learning. Other baselines, OWL and Random, fail to choose the appropriate head for testing and do not perform well on all tasks. Learning the new task as quickly as Finetuning without forgetting the old ones, our method MACPro obtains excellent performance. The comparable average performance to Oracle also indicates that MACPro can accurately choose the optimal head for testing.

2) Ablation Studies: As MACPro is composed of multiple components, we here design ablation studies on benchmark LBF and PP to investigate their impacts. First, for task contextualization learning, we derive *W/o model* by removing the forward model h and its corresponding loss term $\mathcal{L}_{\text{model}}$ and using the contrastive loss only to optimize the global trajectory encoder g_θ. Next, instead of extracting the representation of trajectories with POE, we use the average of the Gaussian distributions generated by the transformer network as the representation, and we call it *W/o POE*. Furthermore, we also introduce *W/o oracle*, which has a similar number of parameters as MACPro, to investigate whether the superiority of MACPro over QMIX is due to the increase in the number of parameters. Finally, we remove both global contrastive loss $\mathcal{L}_{\text{cont}_g}$ and local contrastive loss $\mathcal{L}_{\text{cont}_l}$ to derive *W/o* $\textit{cont}_{g,l}$.

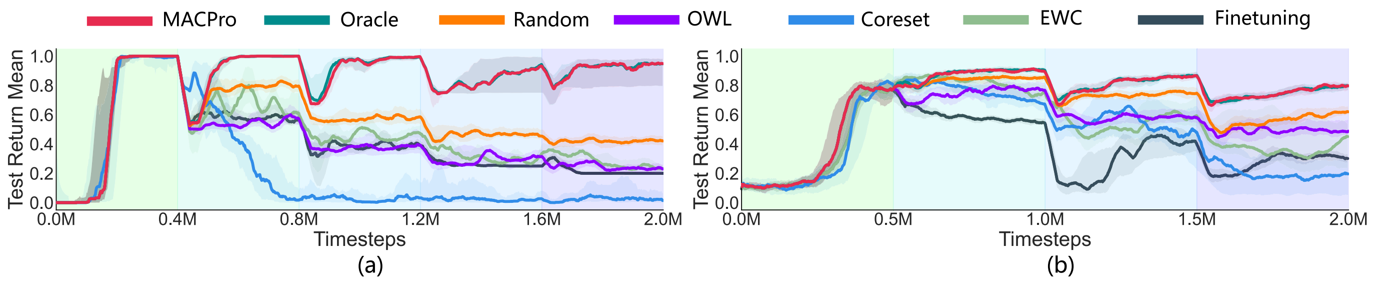
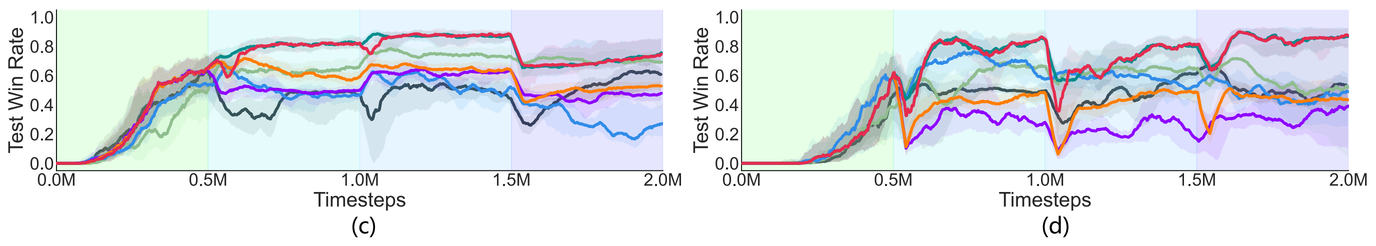



Fig. 5. Performance comparison with baselines. Each task is trained for 400k steps in LBF and 500k steps in other benchmarks, and each plot indicates the average performance across all tasks seen so far. Different background colors indicate different task periods. (a) LBF. (b) PP. (c) Marines. (d) SZ.

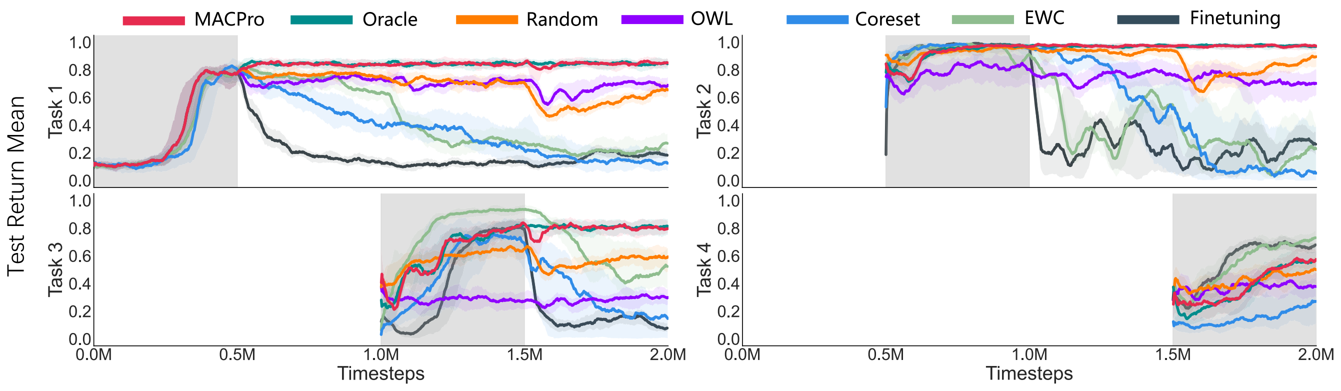


Fig. 6. Complete continual learning results on PP. Four plots for four tasks are displayed. The plot for each task has a gray background, indicating that the agents are training on this task during this period, and there exists blank space in the plots of task $n(n > 1)$ because the task has not appeared yet, so we do not test the performance on it until agents have started training on it. For example, task 2 appears at $t = 0.4$M, and it is trained for 400k steps, so $t = 0.0 \sim 0.4$M is blank and $t = 0.4\text{M} \sim 0.8$M has a gray background.

TABLE I
ABLATION STUDIES ON LBF. THE VALUES IN THE TABLE ARE THE RETURN VALUES

Method	Task 1	Task 2	Task 3	Task 4	Task 5	Average
Ours	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.80 ± 0.40	0.71 ± 0.38	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.90 \pm 0.09}$
W/o model	0.93 ± 0.10	0.68 ± 0.46	0.67 ± 0.47	0.85 ± 0.21	0.93 ± 0.10	0.81 ± 0.18
W/o cont$_g$	$\mathbf{1.00 \pm 0.00}$	0.67 ± 0.47	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.97 \pm 0.03}$	0.55 ± 0.41	0.84 ± 0.18
W/o POE	0.99 ± 0.01	0.86 ± 0.19	0.97 ± 0.03	0.74 ± 0.17	0.69 ± 0.44	0.85 ± 0.05
W/o oracle	0.21 ± 0.39	0.60 ± 0.49	0.06 ± 0.12	0.24 ± 0.28	0.24 ± 0.38	0.27 ± 0.07
W/o cont$_l$	0.98 ± 0.03	0.86 ± 0.19	0.76 ± 0.17	0.75 ± 0.20	0.97 ± 0.04	0.86 ± 0.06
W/o cont$_{g,l}$	0.79 ± 0.40	0.99 ± 0.02	$\mathbf{1.00 \pm 0.00}$	0.34 ± 0.42	0.54 ± 0.38	0.73 ± 0.12

As shown in Tables I and II, we can find that when the model loss is removed, *W/o model* suffers from performance degradation in most tasks, indicating the necessity for task representation learning. Furthermore, the POE mechanism also slightly influences the learning performance, demonstrating that the special integration of multiple representations of trajectories can facilitate representation learning. Consequently, when removing the oracle loss function, *W/o oracle* sustains great performance degradation and even fails in some tasks, indicating that a simply larger network cannot fundamentally improve the performance. We also find that contrastive learning loss has a positive effect on performance. We further design *W/o cont$_g$* and *W/o cont$_l$* by setting $\alpha_{\text{cont}_g} = 0$ and $\alpha_{\text{cont}_l} = 0$ to study the impact of contrastive loss. Both variants

TABLE II
ABLATION STUDIES ON PP. THE VALUES IN THE TABLE ARE THE RETURN VALUES

Method	Task 1	Task 2	Task 3	Task 4	Average
Ours	0.75 ± 0.05	$\mathbf{0.97 \pm 0.02}$	$\mathbf{0.77 \pm 0.09}$	$\mathbf{0.66 \pm 0.05}$	$\mathbf{0.79 \pm 0.11}$
W/o model	0.73 ± 0.1	0.86 ± 0.08	0.61 ± 0.19	0.54 ± 0.1	0.69 ± 0.12
W/o cont$_g$	0.38 ± 0.11	0.89 ± 0.07	0.16 ± 0.18	0.65 ± 0.05	0.52 ± 0.28
W/o POE	0.70 ± 0.13	0.84 ± 0.12	0.39 ± 0.23	0.55 ± 0.07	0.62 ± 0.17
W/o oracle	0.39 ± 0.16	0.34 ± 0.25	0.45 ± 0.12	0.27 ± 0.19	0.36 ± 0.06
W/o cont$_l$	$\mathbf{0.76 \pm 0.06}$	0.91 ± 0.07	0.64 ± 0.2	0.48 ± 0.16	0.7 ± 0.16
W/o cont$_{g,l}$	0.41 ± 0.28	0.83 ± 0.08	0.25 ± 0.25	0.33 ± 0.2	0.45 ± 0.22

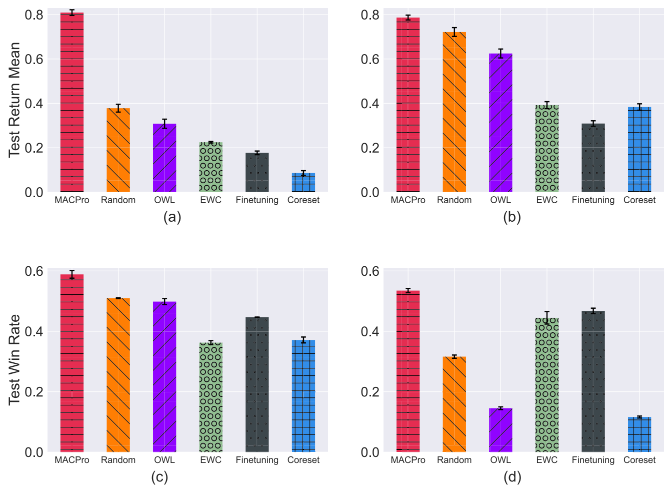


Fig. 7. Generalization results. (a) LBF. (b) PP. (c) Marines. (d) SZ.

suffer from performance degradation, indicating the necessity of contrastive learning.

3) Generalization Results: As we focus on training each emerging task sequentially, it produces a significant risk of overfitting. What is more, the ultimate goal of continual learning agents is to not only perform well on seen tasks but also utilize the learned experiences to complete future unseen tasks. Here, we design experiments to test the generalization ability of MACPro compared with multiple baselines. Concretely, we design 20 additional tasks (details can be seen in Appendix A) for each benchmark that agents have not encountered before to conduct zero-shot experiments.

As can be seen from Fig. 7, MACPro demonstrates the most superior performance compared to the multiple compared baselines, indicating that it has strong generalization ability due to the multiagent task contextualization learning module and the decentralized task approximation procedure. Note that the baseline Oracle is not tested here because there is no ground-truth head selection on unseen tasks.

C. Task Contextualization Analysis

Then, we visualize the development of continual learning performance, along with changes in task representation and factored heads, to demonstrate how our method works. Concretely, we build a task sequence with ten tasks of benchmark PP. As shown in Fig. 8, when $t = 1.0$M, the incoming task 3 is similar to task 1, and their latent variables are distributed in the same area (the green ellipse). Task 3 shares the same head as task 1, leading to an unchanged number of task heads. When a dissimilar task is encountered at $t = 2.5$M, none of the learned tasks is similar to the incoming task 6. The latent variables of task 6 are distributed in a new area (the red ellipse), and MACPro expands a new head accordingly. This process continuously proceeds, until the learning procedure ends at $t = 5.0$M, when the latent variables of all ten tasks are distributed in four separate clusters, and MACPro has four heads, respectively. The latent variables of the representations $f_{\theta_i'}(\tau^i)$ encoded by individual trajectory encoders, denoted as $\circ$, are also displayed (we omit them in the first two 3-D figures for simplicity). It shows that the representations learned by $f_{\theta_i'}(\tau^i)$ are close to $g_\theta(\tau)$, enabling accurate decentralized task approximation and good performance.

Consequently, we can further find the learning curve in the top row of Fig. 8, along with the number of separate heads that change according to the corresponding task representations. We also compare two extreme-case methods, where Finetuning holds a single head for all tasks, enjoying high scalability but strong catastrophic forgetting. On the contrary, MACPro w/o adaptive expansion maintains one head for each task and can achieve high learning efficiency, but the heads' storage cost may impede it when facing a large number of tasks. Our method, MACPro, achieves comparable or even better learning ability but consumes fewer heads, showing high learning efficiency and scalability.

D. Integrative Abilities and Sensitive Studies

MACPro is agnostic to specific value-based cooperative MARL methods. Thus, we can use it as a plug-in module and integrate it with existing MARL methods, such as VDN [13], QMIX [14], and QPLEX [67]. As shown in Table III, when integrating with MACPro, the performance of the baselines vastly improves, indicating that MACPro has a high generality ability for different methods to facilitate continual learning.

As MACPro includes multiple hyperparameters, here, we conduct experiments on benchmark PP to investigate how each one influences continual learning ability. First, α_{reg} controls the extent of the restriction on changing the parameters of the shared feature extractor ϕ_i. If it is too small, the dramatic

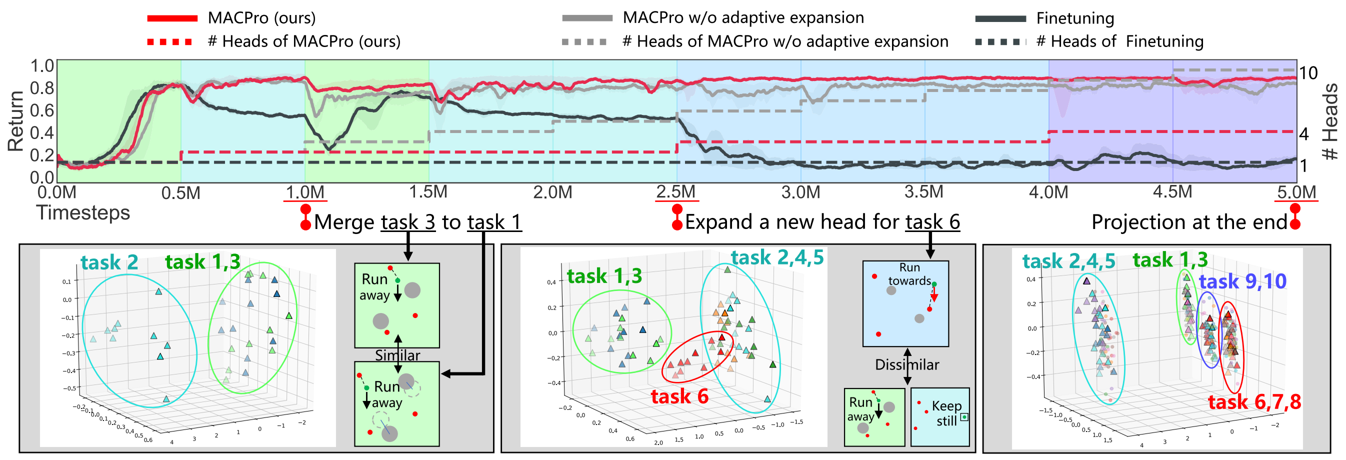


Fig. 8. Task contextualization analysis. Similar tasks have the same background color, e.g., task 1 and task 3 correspond to the green background. When encountering a new task, we sample latent variables generated by $g_\theta(\tau)$ and apply dimensionality reduction to them by principal component analysis (PCA) [66], denoted as △.

TABLE III
INTEGRATIVE ABILITIES

Envs	Method	VDN	QMIX	QPLEX
LBF	W/ MACPro	**0.92 ± 0.02**	**0.90 ± 0.09**	**0.97 ± 0.03**
	W/o MACPro	0.21 ± 0.01	0.20 ± 0.00	0.21 ± 0.01
PP	W/ MACPro	**0.62 ± 0.06**	**0.80 ± 0.02**	**0.63 ± 0.05**
	W/o MACPro	0.29 ± 0.05	0.27 ± 0.04	0.30 ± 0.03

change of ϕ_i's parameters may induce severe forgetting. On the other hand, if it is too large, agents remember the old task at the expense of not learning the new task. We, thus, find each hyperparameter via grid search. As shown in Fig. 9(a), we can find that $\alpha_{\text{reg}} = 500$ is the best choice in this benchmark. Furthermore, another adjustable hyperparameter α_{cont_g} influences the training of global trajectory encoder g_θ in multiagent task contextualization learning. Fig. 9(b) shows that $\alpha_{\text{cont}_g} = 0.1$ performs the best. In decentralized task approximation, α_{cont_l} balances the learning of local trajectory encoder $f_{\theta_i'}$. We find that in Fig. 9(c), $\alpha_{\text{cont}_l} = 0.1$ performs the best. During decentralized execution, agents first probe P episodes before evaluation to derive the task contextualization and select the optimal head. The more episodes the agents can probe, the more information about the evaluating task the agents can gain. However, setting P to a very large value is not practical. We find in Fig. 9(d) that $P = 20$ is enough for accurate task approximation.

VI. Final Remarks

Observing the great significance and practicability of continual learning, this work takes a further step toward continual coordination in cooperative MARL. We first formulate this problem, where agents are centralized trained with access to global information; then, an efficient task contextualization learning module is designed to obtain efficient task representation, and an adaptive dynamic network expansion technique is applied. We, finally, design a local continual coordination mechanism to approximate the global optimal task head selection. Extensive experiments demonstrate the effectiveness of our approach. To the best of our knowledge, the proposed MACPro is the first multiagent continual algorithm capable of multiagent scenarios. While our work contributes valuable insights into continual coordination, it is crucial to recognize the limitations, including the cost of few-shot exploration and extra data storage, and the need for heuristically designed environments or tasks. Future work on more reasonable and efficient ways, such as ZSC, generative replay, and environment automatic generation, extending MACPro to more complex scenarios, such as continual coordination with communication or robust continual coordination, and applying MACPro to real-world applications, would be of great value.

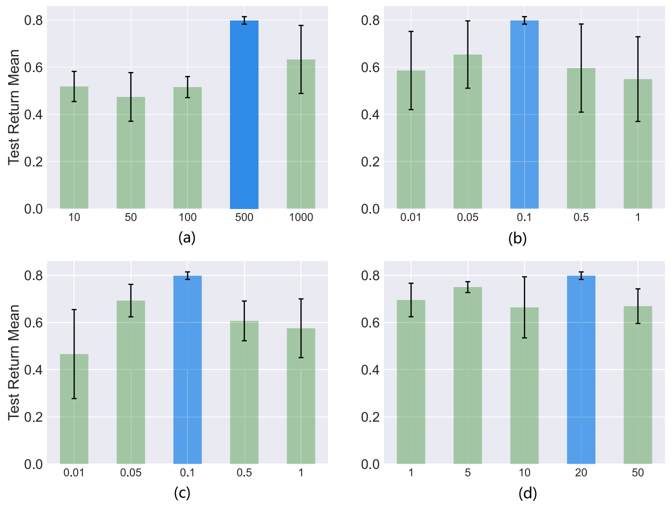


Fig. 9. Test results of parameter sensitivity studies. (a) Sensitivity of α_{reg}. (b) Sensitivity of α_{cont_g}. (c) Sensitivity of α_{cont_l}. (d) # probe episodes P.

Appendix A
Details About Baselines and Benchmarks

A. Baselines

Finetuning is a simple method based on a single feature extraction model and policy head to learn a sequence of tasks,

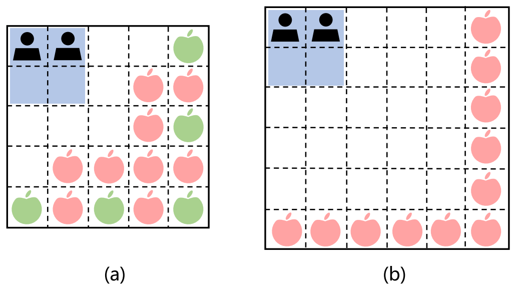


Fig. 10. Benchmark LBF used in this article. (a) 5 × 5 grid. (b) 6 × 6 grid.

ignoring the changes in tasks and directly tuning the learned policy on the current task. However, if the current task is different from the previous ones, the parameters of the policy network would change dramatically to acquire good performance on the current task, thus inducing the phenomenon of catastrophic forgetting.

EWC [46] is one of the regularization-based approaches to address the catastrophic forgetting problem. Concretely, it tries to maintain expertise on old tasks by selectively slowing down learning on the weights that are important for them. The loss function for learning the current task M is

$$\mathcal{L}(\theta) = \mathcal{L}_M(\theta) + \frac{\lambda}{2}\sum_j F_j(\theta_j - \theta_{M-1,j})^2 \tag{12}$$

where $\mathcal{L}_M(\theta)$ is the loss for task M only and F_i is the i^{th} diagonal element of the Fisher information matrix F. θ_{M-1} is the saved snapshot of θ after training task $M-1$, and j labels each parameter. λ is an adjustable coefficient to control the tradeoff between the current task and previous ones. In this article, we set the parameters of the agent's Q network as θ and calculate the Fisher information matrix F with temporal difference error.

Unlike EWC that constrains the change of network parameters when learning a new task, **Coreset** [48], one of the replay-based methods, prevents catastrophic forgetting by choosing and storing a significantly smaller subset of data of the previous tasks. When learning the current task, the stored data are also utilized for training the policy, which is expected to remember the previous tasks. In this article, we set the replay buffer to uniformly store trajectories of all seen tasks, including the current one. A small batch of trajectories of one randomly chosen task is sampled from the buffer to train the agents' network.

OWL [23] is a recent approach that learns a multihead architecture and achieves high learning efficiency when the tasks in a sequence have conflicting goals. Specifically, it learns a factorized policy with a shared feature extractor but separate heads, each specializing in only one task. With a similar architecture to our method MACPro, we can apply it to learn task sequences in a continual manner. During testing, OWL uses bandit algorithms to find the policy that achieves the highest test task reward. However, this strategy could bring performance degradation since agents choose action uniformly at the beginning of the episodes.

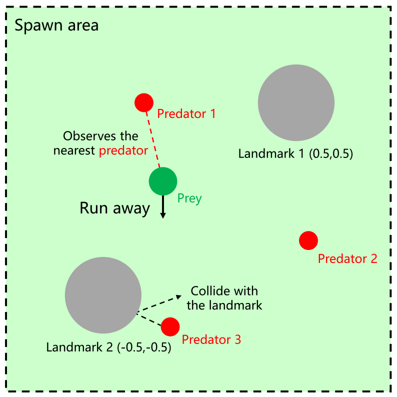


Fig. 11. Benchmark PP used in this article, where the prey's policy is to run in the opposite direction of the nearest predator.

B. Benchmarks

We select four multiagent environments for the evaluation benchmarks. LBF [27] is a cooperative grid world game (see Fig. 10). Here, the positions of two agents and one food are represented by discrete states, and agents are randomly spawned at cells $(0, 0), (0, 1), (1, 0), (1, 1)$. Each agent observes the relative position of other agents and the food, moves a single cell in one of the four directions (up, left, down, right), and gains reward 1 if and only if both agents navigate to the food and be at a distance of one cell from the food. In continual learning ability comparison, we design five tasks in a 5 × 5 grid, with the food at cell $(0, 4), (2, 4), (4, 4), (4, 2), (4, 0)$ [green food in Fig. 10(a)], respectively. To test the generalization ability of different methods, we further design 20 tasks in both 5 × 5 and 6 × 6 grid, and the food position is changed as well [red food in Fig. 10(a) and (b)].

PP [11] is another popular benchmark where three agents (predators) need to chase the adversary agent (prey) and collide it to win the game (see Fig. 11). Here, agents and landmarks are represented by circles of different sizes, and colliding means circles' intersection. Positions of the two fixed landmarks and positions and speed of the predators and the prey are encoded into continuous states. The predators and the prey can accelerate in one of the four directions (up, left, down, right). In different tasks, the position of landmarks, the predators and the prey's acceleration, maximum speed, and spawn areas, and the fixed heuristic policies that the prey uses are different. Specifically, the prey: 1) runs in the opposite direction of the nearest predator; 2) stays still at a position far away from the predators; 3) runs toward the nearest predator; and 4) runs in a random direction with great speed. Predators gain reward 1 if n of them collide with the prey at the same time [$n = 1$ in cases 1), 2), and 4) and $n = 2$ in case 3)]. In the generalization test, for one original task, we create one corresponding additional task by adding one constant ξ to the original value of different task parameters, including landmark's size, x-coordinate, y-coordinate, predator's and prey's size, acceleration, and maximum speed. We set $\xi = \pm 0.01, \pm 0.02, 0.03$ on the four original tasks to derive 20 addition tasks to test the generalization ability.

Fig. 12. Benchmark Marines and SZ used in this article. (a) Task 5m_vs_6m in the Marines series. (b) Task 2s3z in the SZ series.

The other benchmarks are two task series, named **Marines** and **SZ** (see Fig. 12), from SMAC [28], involving various numbers of Marines and Stalkers/Zealots in two camps, respectively. The goal of the multiagent algorithm is to control one of the camps to defeat the other. Agents receive a positive reward signal by causing damage to enemies, killing enemies, and winning the battle. On the contrary, agents receive a negative reward signal when they receive damage from enemies, get killed, and lose the battle. Each agent observes information about the map within a circular area around it and takes actions, including moving and firing when it is alive. In continual learning ability comparison, the Marines series consists of: 1) 5m_vs_6m; 2) 13m; 3) 4m; and 4) 8m_vs_9m; the SZ series consists of: 1) 2s1z_vs_3z; 2) 2s3z; 3) 3s5z; and 4) 2s2z_vs_4s, where *m* stands for *marine*, which can attack an enemy unit from a long distance at a time, *s* stands for the *stalker*, which attacks like a marine and has a self-regenerate shield, and *z* stands for *zealot*, which also has a self-regenerate shield but can only attack an enemy unit from a short distance. For the generalization test, we first decrease the default sight range and shoot range by 1 to create four additional tasks for both Marines and SZ. Then, we design scenarios {3m, 5m, 6m, 7m, 8m, 9m, 10m, 11m, 12m, 4m_vs_5m, 6m_vs_7m, 7m_vs_8m, 9m_vs_10m, 10m_vs_11m, 11m_vs_12m, 12m _vs_13m} for Marines and scenarios {1s1z, 1s2z, 1s3z, 2s1z, 2s2z, 2s4z, 3s1z, 3s2z, 3s3z, 3s4z, 4s2z, 4s3z, 4s4z, 2s2z_vs_4z, 3s3z_vs_6s, 4s4z_vs_8z} for SZ. If *vs* is in the task name, it indicates that the two camps are asymmetric, e.g., in 5m_vs_6m, there are five marines in our camp and six enemy marines. Otherwise, it indicates that the two camps are symmetric, e.g., in 2s3z, there are two stalkers and three zealots in both camps.

C. Value Function Factorization MARL Methods

As we investigate the integrative abilities of MACPro in this article, here, we introduce the value-based methods used in this article, including VDN [13], QMIX [14], and QPLEX [67]. The difference among the three methods lies in the mixing networks, with increasing representational complexity. Our proposed framework MACPro follows the CTDE paradigm used in value-based MARL methods.

These three methods all follow the individual-global-max (IGM) [68] principle, which asserts the consistency between joint and local greedy action selections by the joint value function $Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a})$ and individual value functions

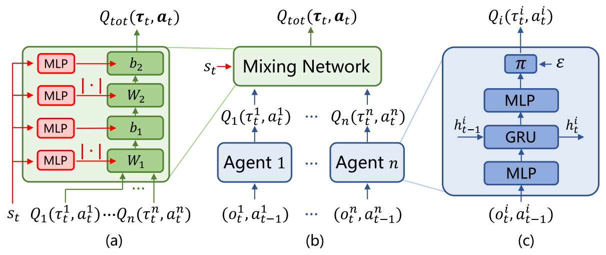


Fig. 13. Overall structure of QMIX. (a) Detailed structure of the mixing network, whose weights and biases are generated from a hypernet (red), which takes the global state as the input. (b) QMIX is composed of a mixing network and several agent networks (overall structure). (c) Detailed structure of the individual agent network.

$[Q_i(\tau^i, a^i)]_{i=1}^n$

$$\forall \boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}, \arg\max_{\boldsymbol{a} \in \boldsymbol{\mathcal{A}}} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \left(\arg\max_{a^1 \in \mathcal{A}} Q_1(\tau^1, a^1), \ldots, \arg\max_{a^n \in \mathcal{A}} Q_n(\tau^n, a^n)\right). \quad (13)$$

VDN [13] factorizes the global value function $Q_{\text{tot}}^{\text{VDN}}(\boldsymbol{\tau}, \boldsymbol{a})$ as the sum of all agents' local value functions $[Q_i(\tau^i, a^i)]_{i=1}^n$

$$Q_{\text{tot}}^{\text{VDN}}(\boldsymbol{\tau}, \boldsymbol{a}) = \sum_{i=1}^{n} Q_i(\tau^i, a^i). \quad (14)$$

QMIX [14] (see Fig. 13) extends VDN by factorizing the global value function $Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})$ as a monotonic combination of the agents' local value functions $[Q_i(\tau^i, a^i)]_{i=1}^n$

$$\forall i \in \mathcal{N}, \frac{\partial Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q_i(\tau^i, a^i)} > 0. \quad (15)$$

We mainly implement MACPro on QMIX for its proven performance in various papers. VDN and QMIX are two sufficient conditions of the IGM principle to factorize the global value function, but the conditions that they propose are not necessary. To achieve a complete IGM function class, **QPLEX** [67] uses duplex dueling network architecture by decomposing the global value function as

$$\begin{aligned} Q_{\text{tot}}^{\text{QPLEX}}(\boldsymbol{\tau}, \boldsymbol{a}) &= V_{\text{tot}}(\boldsymbol{\tau}) + A_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) \\ &= \sum_{i=1}^{n} Q_i(\boldsymbol{\tau}, a^i) + \sum_{i=1}^{n} (\lambda^i(\boldsymbol{\tau}, \boldsymbol{a}) - 1) A_i(\boldsymbol{\tau}, a^i) \end{aligned} \quad (16)$$

where $\lambda^i(\boldsymbol{\tau}, \boldsymbol{a})$ is the weight depending on the joint history and action, and $A_i(\boldsymbol{\tau}, a^i)$ is the advantage function conditioning on the history information of each agent. QPLEX aims to find the monotonic property between the individual Q function and the individual advantage function.

Appendix B
Architecture, Infrastructure, and Hyperparameters' Choices of MACPro

A. Network Architecture

In benchmarks, LBF and PP, the number of agents, the dimension of state, observation, and action remain unchanged in different tasks. Specifically, for (1) agent networks, we apply the technique of parameter sharing and design

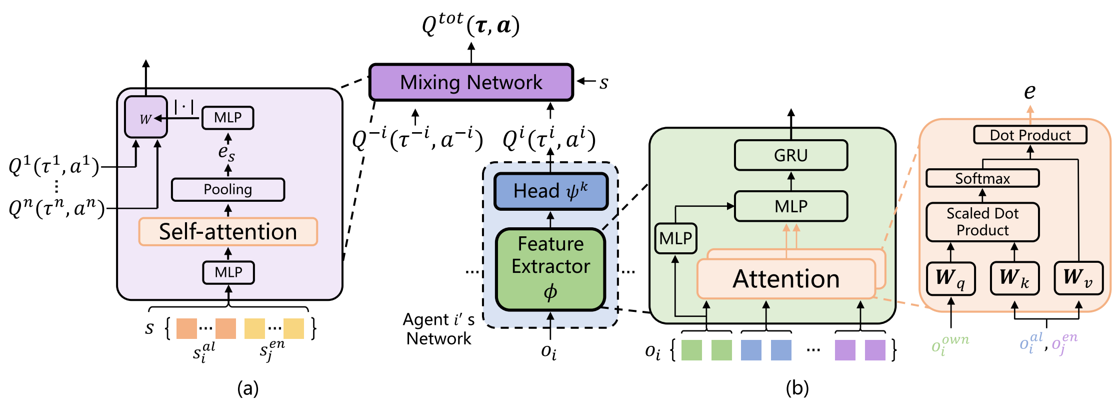


Fig. 14. Network architecture used in Marines and SZ. (a) Mixing network. (b) Individual Q network.

the feature extractor ϕ as a five-layer multi-layer perceptron (MLP) and a gated recurrent unit (GRU) [69]. The hidden dimension is 128 for the MLP and 64 for the GRU. Then, each separated head is a linear layer, which takes the output of the feature extractor as input and outputs the Q value of all actions. For (2) task contextualization learning, we design a global trajectory encoder g_θ and a context-aware forward model h. g_θ consists of a transformer encoder, an MLP, and a POE module. The six-layer transformer encoder takes trajectory $\tau = (s_1, \ldots, s_T)$ as input and outputs T 32-D embeddings. Then, the three-layer MLP transforms these embeddings into means and standard deviations of T Gaussians. Finally, the POE module acquires the joint representation of the trajectory, which is also a Gaussian distribution $\mathcal{N}(\mu_\theta(\tau), \sigma_\theta^2(\tau))$. The context-aware forward model h is a three-layer MLP that takes as input the concatenation of the current state, local observations, actions, and task contextualization sampled from the joint task distribution and outputs the next state, next local observations, and reward. The hidden dimension is 64, and the reconstruction loss is calculated by mean squared error. For (3) decentralized task approximation, the local trajectory encoders $f_{\theta_i'}(i = 1, \ldots, n)$ have the same structure as the global trajectory encoder g_θ.

In benchmark Marines and SZ, a new difficulty arises since the number of agents, the dimension of state, observation, and action could vary from task to task, making the networks used in LBF and PP fail to work. Inspired by the unshaped network [70] and the popularly used population-invariant network (PIN) technique in MARL [43], we design a different feature extractor, head, and monotonic mixing network [14] that learns the global Q value as a combination of local Q values. For the feature extractor (see Fig. 14), we decompose the observation o_i into different parts, including agent i's own information o_i^{own}, ally information o_i^{al}, and enemy information o_i^{en}. Then, we feed them into attention networks to derive a *fixed-dimension* embedding e

$$\begin{aligned}
q &= \texttt{MLP}_q\left(o_i^{\text{own}}\right)\\
\mathbf{K}_{\text{al}} &= \texttt{MLP}_{K_{\text{al}}}\left(\left[o_i^{\text{al}_1}, \ldots, o_i^{\text{al}_j}, \ldots\right]\right)\\
\mathbf{V}_{\text{al}} &= \texttt{MLP}_{V_{\text{al}}}\left(\left[o_i^{\text{al}_1}, \ldots, o_i^{\text{al}_j}, \ldots\right]\right)\\
e_{\text{al}} &= \texttt{softmax}\left(q\mathbf{K}_{\text{al}}{}^{\text{T}}/\sqrt{d_k}\right)\mathbf{V}_{\text{al}}\\
\mathbf{K}_{\text{en}} &= \texttt{MLP}_{K_{\text{en}}}\left(\left[o_i^{\text{en}_1}, \ldots, o_i^{\text{en}_j}, \ldots\right]\right)\\
\mathbf{V}_{\text{en}} &= \texttt{MLP}_{V_{\text{en}}}\left(\left[o_i^{\text{en}_1}, \ldots, o_i^{\text{en}_j}, \ldots\right]\right)\\
e_{\text{en}} &= \texttt{softmax}\left(q\mathbf{K}_{\text{en}}{}^{\text{T}}/\sqrt{d_k}\right)\mathbf{V}_{\text{en}}\\
e &= \left[\texttt{MLP}\left(o_i^{\text{own}}\right), e_{\text{al}}, e_{\text{en}}\right] \qquad (17)
\end{aligned}$$

where $[\cdot, \cdot]$ is the vector concatenation operation, d_k is the dimension of the query vector, and bold symbols are matrices. Embedding e is then fed into an MLP and a GRU to derive the output of the feature extractor ϕ_i. Finally, the output is fed into the policy head, a three-layer MLP, to derive the Q value. Furthermore, the dimension of states could also vary in Marines and SZ. Like the way we deal with observations, state s is decomposed into ally information s_i^{al} and enemy information s_j^{en}. Then, their embeddings are fed into an attention network to derive a *fixed-dimension* embedding e_s. Finally, we feed e_s into the original mixing network whose structure is used in benchmarks LBF and PP.

Besides the networks mentioned above, the global trajectory encoder g_θ, forward model h, and local trajectory encoders $f_{\theta_i'}$ are also involved with this issue. For g_θ and $f_{\theta_i'}$, we first apply the same technique to derive the fixed-dimension embeddings of states and observations and then feed them into the transformer encoders. For the forward model h, we treat each agent's action as a part of its own observation and feed their concatenation into the attention network to derive an embedding, which will be fed into h with the embedding of state and task contextualization. Then, h outputs a fixed-dimension embedding. We decode it into the next state, local observations, and reward with task-specific MLP decoders to calculate the reconstruction loss $\mathcal{L}_{\text{model}}$.

B. Overall Flow of MACPro

To illustrate the process of training, the overall training flow of MACPro is shown in Algorithm 1. Here, lines 3–14 express the process of dynamic agent network expansion, where we use task contextualization to decide whether we should initialize a new head for the current task. To initialize a new head (line 12), we propose two strategies. One strategy is to copy the parameters of the head learned from the last task. The other is to construct an entirely new head by resetting the parameters randomly. We use the first strategy in LBF, PP, and Marines and the second one in SZ. Both strategies work well in the experiments. Then, we started training on the new task. In line 15, we can choose to reset the ϵ-greedy schedule

Algorithm 1 MACPro: Training

Input: Task sequence $\mathcal{Y} = \{\text{task } 1, \ldots, \text{task } M\}$
Initialize: trajectory encoder g_θ, forward model h, individual trajectory encoders $f_{\theta'_{i:n}}$, agents' feature extractors $\phi_{1:n}$

```
1:  for $m = 1, \ldots, M$ do
2:      Set up task $m$
3:      if $m = 1$ then
4:          $\psi^1_{1:n} \leftarrow$ Initialized new head
5:      else
6:          // Dynamic Network Expansion
7:          Calculate $l, l'$ according to Equation 5
8:          Find $k_* = \arg\min_{1\le k\le K} l'_k$
9:          if $l'_{k_*} \le \lambda_{\text{new}} l_{k_*}$ then
10:             Merge task $m$ to the task(s) that $\psi^{k_*}_{1:n}$ tasks charge, $\psi^m_{1:n} \leftarrow \psi^{k_*}_{1:n}$
11:         else
12:             $\psi^m_{1:n} \leftarrow$ Initialized new head
13:         end if
14:     end if
15:     (Optional) Reset the $\epsilon$-greedy schedule
16:     for $t = 1, \ldots, T_{\text{task } m}$ do
17:         Collect trajectories with $\{\phi_{1:n}, \psi^m_{1:n}\}$, store in buffers $\mathcal{D}, \mathcal{D}'$
18:         Update $\{\phi_{1:n}, \psi^m_{1:n}\}$ according to $\mathcal{L}_{\text{RL}}$
19:         if $t$ mod $\kappa_1 = 0$ then
20:             // $\kappa_1$ is the interval of updating the encoders
21:             // Task Contextualization Learning
22:             Train $g_\theta, h$ according to $\mathcal{L}_{\text{context}}$
23:             Train $f_{\theta'_{1:n}}$ according to $\mathcal{L}_{\text{approx}}$
24:         end if
25:         if $t$ mod $\kappa_2 = 0$ then
26:             // $\kappa_2$ is the interval of saving the learning head
27:             Save head $\psi^m_{1:n}$
28:         end if
29:     end for
30:     Evaluate task $1, \ldots, m$ according to Alg. 2
31:     Empty the experience replay buffer $\mathcal{D} = \emptyset$
32: end for
```

to enhance exploration (adopted in SZ) or not (adopted in LBF, PP, and Marines), where the schedule is to decay ϵ from 1 to 0.05 in 50k timesteps. Next, we iteratively update the parameters of each component in lines 16–27, where we also save the current head. Finally, we test all seen tasks, empty the replay buffer for the current task, and switch to the next task.

Besides, the execution flow of MACPro is shown in Algorithm 2. In the execution phase, for each testing task, agents first roll out P episodes to probe the environment and derive the context (lines 3–7). With the gathered local information, each agent independently selects an optimal head to perform this task (lines 8 and 9). The whole framework is trained end to end with collected episodic data on NVIDIA GeForce RTX 2080 Ti and 3090 GPUs with a time cost of about 5 h in benchmark LBF and PP, and 18 h in benchmark Marines and SZ.

Algorithm 2 MACPro: Execution

Input: Task sequence $\{\text{task } 1, \ldots, \text{task } M\}$, feature extractors $\phi_{1:n}$, heads $\{\psi^k_{1:n}\}_{k=1}^K$
Parameter: Number of probing episodes P

```
1:  for $m = 1, \ldots, M$ do
2:      Set up task $m$
3:      for $p = 1, \ldots, P$ do
4:          Randomly choose an integer $k$ from $\{1, \ldots, K\}$
5:          Agents collect one trajectory $\boldsymbol{\tau}_p$ with $\{\phi_{1:n}, \psi^k_{1:n}\}$
6:          Each agent $i$ calculates the mean value $\mu_{\theta'_i}(\tau^i_p)$ of the trajectory representation $f_{\theta'_i}(\tau^i_p)$
7:      end for
8:      Each agent $i$ selects the optimal head $\psi_i^{k^{\star i}}$, where $k^{\star i} = \underset{1\le k\le K}{\text{argmin}} \min_{1\le p\le P} \|\mu_{\theta'_i}(\tau^i_p) - \frac{1}{bs}\sum_{j=1}^{bs} \mu^j_k\|_2$.
9:      Agents test with $\{\phi_i, \psi_i^{k^{\star i}}\}_{i=1}^n$
10: end for
```

TABLE IV
HYPERPARAMETERS IN EXPERIMENT

Hyperparameter	Value
P (Number of probing episodes)	20
Learning rate for updating networks	0.0005
Number of heads in transformer encoders	3
Number of layers in transformer encoders	6
bs (Batch size of the sampled trajectories)	32
λ_{new} (Threshold for merging similar tasks)	1.5
Hidden Dim (Dimension of hidden layers)	64
Attn Dim (Dimension of Key in attention)	8
Entity Dim (Dimension of Value in attention)	64
Z Dim (Dimension of the encoded Gaussians)	32
κ_1 (Interval (steps) of updating both encoders)	1000
κ_2 (Interval (steps) of saving the learning head)	10000
α_{cont_l} (Coefficient of local contrastive loss $\mathcal{L}_{\text{cont}_l}$)	0.1
α_{cont_g} (Coefficient of global contrastive loss $\mathcal{L}_{\text{cont}_g}$)	0.1
α_{reg} (Coefficient of the l_2- regularization $\mathcal{L}_{\text{reg}}$ on ϕ)	500
Buffer Size of $\mathcal{D}$ (Maximum number of trajectories in $\mathcal{D}$)	5000
Buffer Size of $\mathcal{D}'$ (Maximum number of trajectories in $\mathcal{D}'$)	5000

C. Hyperparameters Choices

Our implementation of MACPro[1] is based on the PyMARL[2] [28] codebase with StarCraft 2.4.6.2.69232 and uses its default hyperparameter settings. For example, the discounted factor used to calculate the temporal difference error is set to the default value of 0.99. The selection of the additional hyperparameters introduced in our approach, e.g., the time interval of saving the heads, is listed in Table IV. We use this set of parameters for MACPro in all experiments shown in this article. For the ablation studies, we change one or two parameters to derive a variant of MACPro, investigating the

[1]https://github.com/lilh76/MACPro
[2]https://github.com/oxwhirl/pymarl

impact of different components. For example, in the variant *W/o cont*$_g$, α_{cont_g} is set to 0, while other parameters remain the same as in Table IV.

References

[1] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Appl. Intell.*, vol. 53, no. 11, pp. 13677–13722, Jun. 2023.

[2] G. Sartoretti et al., "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.

[3] J. Wang, W. Xu, Y. Gu, W. Song, and T. C. Green, "Multi-agent reinforcement learning for active voltage control on power distribution networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 3271–3284.

[4] K. Xue et al., "Multi-agent dynamic algorithm configuration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 20147–20161.

[5] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," 2019, *arXiv:1906.04737*.

[6] J. Wang, Z. Ren, B. Han, J. Ye, and C. Zhang, "Towards understanding cooperative multi-agent Q-learning with value factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 29142–29155.

[7] F. Christianos, G. Papoudakis, A. Rahman, and S. V. Albrecht, "Scaling multi-agent reinforcement learning with selective parameter sharing," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 1989–1998.

[8] C. Zhu, M. Dastani, and S. Wang, "A survey of multi-agent deep reinforcement learning with communication," *Auto. Agents Multi-Agent Syst.*, vol. 38, no. 1, p. 4, Jun. 2024.

[9] H. Hu, A. Lerer, A. Peysakhovich, and J. N. Foerster, "'Other-play' for zero-shot coordination," in *Proc. ICML*, 2020, pp. 4399–4410.

[10] J. Guo, Y. Chen, Y. Hao, Z. Yin, Y. Yu, and S. Li, "Towards comprehensive testing on the robustness of cooperative multi-agent reinforcement learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 114–121.

[11] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. NIPS*, 2017, pp. 6379–6390.

[12] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. NeurIPS*, 2022, pp. 24611–24624.

[13] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auto. Agents Multi Agent Syst.*, 2018, pp. 2085–2087.

[14] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.

[15] R. Gorsane, O. Mahjoub, R. J. de Kock, R. Dubb, S. Singh, and A. Pretorius, "Towards a standardised performance evaluation protocol for cooperative MARL," in *Proc. NeurIPS*, 2022, pp. 5510–5521.

[16] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *J. Artif. Intell. Res.*, vol. 75, pp. 1401–1476, Dec. 2022.

[17] R. Mirsky et al., "A survey of ad hoc teamwork: Definitions, methods, and open problems," 2022, *arXiv:2202.10450*.

[18] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, May 2019.

[19] C. Kaplanis, M. Shanahan, and C. Clopath, "Policy consolidation for continual reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 97, 2019, pp. 3242–3251.

[20] E. Lecarpentier, D. Abel, K. Asadi, Y. Jinnai, E. Rachelson, and M. L. Littman, "Lipschitz lifelong reinforcement learning," in *Proc. AAAI*, 2021, pp. 8270–8278.

[21] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6467–6476.

[22] L. Caccia, E. Belilovsky, M. Caccia, and J. Pineau, "Online learned continual compression with adaptive quantization modules," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1240–1250.

[23] S. Kessler, J. Parker-Holder, P. J. Ball, S. Zohren, and S. J. Roberts, "Same state, different task: Continual reinforcement learning without interference," in *Proc. AAAI*, 2022, pp. 7143–7151.

[24] J.-B. Gaya, T. Doan, L. Caccia, L. Soulier, L. Denoyer, and R. Raileanu, "Building a subspace of policies for scalable continual learning," in *Proc. ICLR*, 2023.

[25] K. Zhang, Z. Yang, and T. Baçar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning Control*. Cham, Switzerland: Springer, 2021, pp. 321–384.

[26] H. Nekoei, A. Badrinaaraayanan, A. C. Courville, and S. Chandar, "Continuous coordination as a realistic scenario for lifelong learning," in *Proc. ICML*, 2021, pp. 8016–8024.

[27] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," in *Proc. NeurIPS*, 2021.

[28] M. Samvelyan et al., "The StarCraft multi-agent challenge," in *Proc. Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2019, pp. 2186–2188.

[29] L. Busoniu, R. Babuska, and S. B. De, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Feb. 2008.

[30] H. Wang, Y. Yu, and Y. Jiang, "Fully decentralized multiagent communication via causal inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10193–10202, Dec. 2023.

[31] X. Yao, C. Wen, Y. Wang, and X. Tan, "SMIX(λ): Enhancing centralized value functions for cooperative multiagent reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 52–63, Jan. 2023.

[32] M. Wen et al., "Multi-agent reinforcement learning is a sequence modeling problem," in *Proc. Conf. Neural Inf. Process. Syst.*, 2022, pp. 16509–16521.

[33] G. Hu, Y. Zhu, D. Zhao, M. Zhao, and J. Hao, "Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3966–3978, Aug. 2023.

[34] F. Zhang, C. Jia, Y.-C. Li, L. Yuan, Y. Yu, and Z. Zhang, "Discovering generalizable multi-agent coordination skills from multi-task offline data," in *Proc. ICLR*, 2023.

[35] X. Wang, Z. Zhang, and W. Zhang, "Model-based multi-agent reinforcement learning: Recent progress and prospects," 2022, *arXiv:2203.10603*.

[36] X. Lyu, Y. Xiao, B. Daley, and C. Amato, "Contrasting centralized and decentralized critics in multi-agent reinforcement learning," in *Proc. 20th Int. Conf. Auton. Agents MultiAgent Syst. (AAMAS)*, 2021, pp. 844–852.

[37] W. Du, S. Ding, C. Zhang, and Z. Shi, "Multiagent reinforcement learning with heterogeneous graph attention network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 6851–6860, Oct. 2023.

[38] L. Chen, B. Hu, Z.-H. Guan, L. Zhao, and X. Shen, "Multiagent meta-reinforcement learning for adaptive multipath routing optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5374–5386, Oct. 2022.

[39] Y. Jin, S. Wei, J. Yuan, and X. Zhang, "Hierarchical and stable multiagent reinforcement learning for cooperative navigation control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 90–103, Jan. 2023.

[40] J. Long, D. Yu, G. Wen, L. Li, Z. Wang, and C. L. P. Chen, "Game-based backstepping design for strict-feedback nonlinear multi-agent systems based on reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 817–830, Jan. 2024.

[41] R. Zhang, Q. Zong, X. Zhang, L. Dou, and B. Tian, "Game of drones: Multi-UAV pursuit-evasion game with online motion planning by deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7900–7909, Oct. 2023.

[42] Z. Wang et al., "Expert system-based multiagent deep deterministic policy gradient for swarm robot decision making," *IEEE Trans. Cybern.*, vol. 54, no. 3, pp. 1614–1624, 2024.

[43] R. Qin et al., "Multi-agent policy transfer via task relationship modeling," in *Proc. Deep Reinforcement Learn. Workshop*, 2022.

[44] K. Xue et al., "Heterogeneous multi-agent zero-shot coordination by coevolution," 2022, *arXiv:2208.04957*.

[45] D. Kudithipudi et al., "Biological underpinnings for lifelong learning machines," *Nature Mach. Intell.*, vol. 4, no. 3, pp. 196–210, 2022.

[46] K. James et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[47] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 348–358.

[48] A. Chaudhry et al., "On tiny episodic memories in continual learning," 2019, *arXiv:1902.10486*.

[49] Y. Huang, K. Xie, H. Bharadhwaj, and F. Shkurti, "Continual model-based reinforcement learning with hypernetworks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 799–805.

[50] S. Kessler et al., "The effectiveness of world models for continual reinforcement learning," in *Proc. Conf. Lifelong Learn. Agents*, 2023, pp. 184–204.

[51] S. Lee, J. Ha, D. Zhang, and G. Kim, "A neural Dirichlet process mixture model for task-free continual learning," in *Proc. ICLR*, 2020.

[52] T. Zhang et al., "Dynamics-adaptive continual reinforcement learning via progressive contextualization," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10145851

[53] Z. Wang, C. Chen, and D. Dong, "Lifelong incremental reinforcement learning with online Bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 4003–4016, Aug. 2022.

[54] C. Gao, J. Zhu, F. Zhang, Z. Wang, and X. Li, "A novel representation learning for dynamic graphs based on graph convolutional networks," *IEEE Trans. Cybern.*, vol. 53, no. 6, pp. 3599–3612, Jun. 2023.

[55] C. Gao, Z. Yin, Z. Wang, X. Li, and X. Li, "Multilayer network community detection: A novel multi-objective evolutionary algorithm based on consensus prior information [Feature]," *IEEE Comput. Intell. Mag.*, vol. 18, no. 2, pp. 46–59, May 2023.

[56] M. Teng, C. Gao, Z. Wang, and X. Li, "A communication-based identification of critical drones in malicious drone swarm networks," *Complex Intell. Syst.*, Jan. 2024. [Online]. Available: https://link.springer.com/article/10.1007/s40747-023-01316-9#citeas

[57] M. Wołczyk, M. Zajac, R. Pascanu, L. Kucinski, and P. Miłoś, "Continual world: A robotic benchmark for continual reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 34, 2021, pp. 28496–28510.

[58] S. Powers, E. Xing, E. Kolve, R. Mottaghi, and A. Gupta, "CORA: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents," in *Proc. Conf. Lifelong Learn. Agents*, 2022, pp. 705–743.

[59] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. New York, NY, USA: Springer, 2016.

[60] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inform. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.

[61] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.

[62] L. Yuan, T. Jiang, L. Li, F. Chen, Z. Zhang, and Y. Yu, "Robust multi-agent communication via multi-view message certification," *Sci. China Inf. Sci.*, vol. 67, no. 4, pp. 102–142, 2024.

[63] F.-M. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, and Y. Yu, "A survey on model-based reinforcement learning," *Sci. China Inf. Sci.*, vol. 67, no. 2, pp. 101–121, 2024.

[64] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2005, pp. 539–546.

[65] H. Jeffreys, *The Theory of Probability*. London, U.K.: Oxford Univ. Press, 1998.

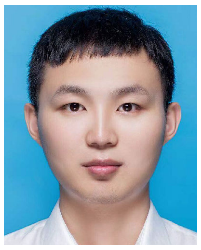
[66] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, Aug. 1987.

[67] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. ICLR*, 2021.

[68] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5887–5896.

[69] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Sep. 2014, pp. 1724–1734.

[70] J. Chai et al., "UNMAS: Multiagent reinforcement learning for unshaped cooperative scenarios," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 2093–2104, Apr. 2023.



Lei Yuan received the B.Sc. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in June 2016, and the M.Sc. degree from the Chinese Aeronautical Establishment, Xi'an, China, in June 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Nanjing University, Nanjing, China.

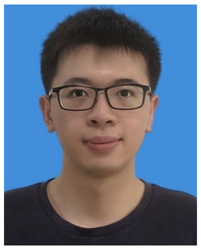
His current research interests mainly include machine learning, reinforcement learning, and multiagent reinforcement learning.

Lihe Li received the B.Sc. degree in automation from the School of Artificial Intelligence, Nanjing University, Nanjing, China, in 2023, where he is currently pursuing the M.Sc. degree.

His research interests include multiagent reinforcement learning and multiagent systems.

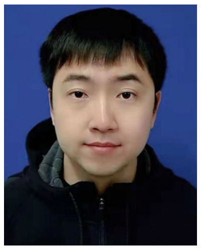
Ziqian Zhang received the B.Sc. degree in brain science and artificial intelligence from the Kuang Yaming Honors School, Nanjing University, Nanjing, China, in 2023, where he is currently pursuing the M.Sc. degree with the School of Artificial Intelligence.

His research interests include multiagent reinforcement learning and multiagent systems.



Fuxiang Zhang received the B.Sc. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in June 2021, where he is currently pursuing the M.Sc. degree with the School of Artificial Intelligence.

His research interests include multiagent reinforcement learning and multiagent systems.



Cong Guan received the B.Sc. and M.Sc. degrees from the School of Mechanical Engineering and Automation, Northeastern University, Shenyang, China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Nanjing University, Nanjing, China.

His current research interests mainly include machine learning, reinforcement learning, and multiagent reinforcement learning.

Yang Yu received the Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 2011.

He then joined the LAMDA Group, Nanjing University, as an Assistant Researcher in 2011 and an Associate Professor in 2014. He joined the School of Artificial Intelligence, Nanjing University, as a Professor, in 2019. Currently, he is working on reinforcement learning in various aspects, including optimization, representation, transfer, and so on. His research interest is in machine learning.

Prof. Yu was a recipient of the National Outstanding Doctoral Dissertation Award, the China Computer Federation Outstanding Doctoral Dissertation Award, the PAKDD'08 Best Paper Award, the GECCO'11 Best Paper (Theory Track), and the Microsoft Research Asia Collaborative Research Award. He is an Associate Editor of *Frontiers of Computer Science* and the Area Chair of Asian Conference on Machine Learning (ACML) 2017, International Joint Conference on Artificial Intelligence (IJCAI) 2018, and International Conference on Pattern Recognition (ICPR) 2018.