GASPOL: DUAL GATING SUB-NETWORKS FOR CONTINUAL MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous coordination under dynamic environments remains a challenge for a cooperative multi-agent system. The existing study on Continual Multi-agent Reinforcement Learning (MARL) focuses on task discrimination that requires trajectory buffers from previous tasks, which could be inapplicable in real-world applications. In addition, the existing study requires feedback or numerous trajectory data to select the best available action layer during the inference phase. Addressing the mentioned issues, this paper proposes a study on rehearsal-free continual multi-agent reinforcement learning and proposes a new method named Dual Gating Sub-networks Policy (GASPOL). GASPOL introduces a dual expert system, i.e., global experts and local experts, as a part of the trained policy network. Global experts preserve general knowledge (stability) of all learned tasks, while local experts obtain task-specific knowledge (plasticity). GASPOL leverages a new actor advantage-guided regularization mechanism for more flexible regularization at each training iteration. Our evaluation on MARL environments, i.e., SMAC, Pettingzoo MPE2, and MAMuJoCo shows that the proposed method significantly achieves better stability-plasticity than the existing state-of-the-art methods. Our extended analysis shows that the proposed method can maintain its performance under various numbers of experts and neurons.

1 INTRODUCTION

Cooperative Multi-Agent Reinforcement Learning (MARL) attracts tremendous interest in the world of artificial intelligence (AI). It has delivered promising solutions for various real-world applications such as games (Samvelyan et al., 2019), robotics (Christianos et al., 2020), and automation (Stephenson & Schaub, 2024). Many aspects are focused on the latest MARL studies, including learning algorithms (Yu et al., 2022; Zhong et al., 2024), architectures (Hu et al., 2021; Wen et al., 2022), and handling dynamic environments (Liu et al., 2025). Despite its fast-growing advancement, continuous coordination remains an open challenge, as the agents suffer from catastrophic forgetting when dealing with a sequence of tasks. The policy network suffers from catastrophic forgetting during the fine-tuning process for the current task. (Nekoei et al., 2021; Tomilin et al., 2025; Hu et al., 2025). In addition, the policy misalignment between the previous and current tasks hampers the policy adaptation. The transferred weights of the previously learned policy networks slow down policy learning on the current task and lead to sub-optimality, rather than boosting it. We termed this phenomenon negative policy transfer. Figure 1 (a) shows a visualization of catastrophic forgetting, where the performance of the policy network on task-1 and task-2 is decreasing upon fine-tuning to the new tasks. Figure 1 (b) shows negative policy transfer, where the performance of the fine-tuned policy model from the previous task is lower than the performance of a model that is trained from scratch. It represents policy misalignment between two tasks despite having a similar objective, i.e., robot control in the MAMuJoCo environment.

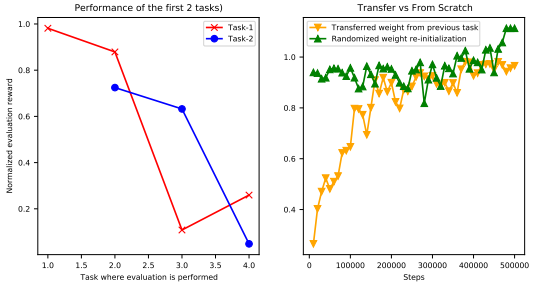

Figure 1: Visualization of (a) Catastrophic forgetting (b) Negative policy transfer in continual MARL setting on multi-agent MuJoCo (MAMuJoCo) games.

Few studies on continual multi-agent reinforcement learning (CMARL) have tried to tackle these challenges by various strategies such as, regularization, task contextualization, and automated head

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

layer selection (Kessler et al., 2022; Yuan et al., 2024). These customized strategies overcome catastrophic policy forgetting for multi-agent systems better than native continual learning methods (CL) methods such as (Kirkpatrick et al., 2017). Despite their promising performance, the methods have several drawbacks related to applicability. First, the methods require environment rewards and update part of their networks during the evaluation phase. Such a mechanism is impractical, since the environment feedback may not be available in the inference phase, and decision-making needs to be done in a real-time manner without delayed steps for the network update. Second, the methods save previous task trajectories to be rehearsed along with the current task trajectories. Aside from applicability issues for a real-world use case due to constrained memory space and data openness policy, the assumption may harm the policy model, as different tasks have different ranges of reward values. Third, the existing methods have not been evaluated yet in problems with continuous spaces, that is more challenging than discrete spaces.

Addressing these gaps, we propose a new rehearsal-free continual multi-agent reinforcement learning method named Dual Gating Sub-networks Policy (GASPOL). We designed an interconnected dual-expert system, i.e., global experts and local experts as sub-networks of the trained policy model. Global experts preserve general knowledge (stability) of all learned tasks, while local experts obtain task-specific knowledge (plasticity). GASPOL introduces a new actor advantage-guided regularization mechanism to scale the regularization loss based on the current and old policies, rather than using the raw weight importance. This mechanism is designed to achieve less sensitive updates, accommodating better stability on the previous task policy, while trying to achieve a new policy for the currently learned task. Our contributions are: (1) We highlighted catastrophic forgetting and negative transfer in the continual multi-agent reinforcement learning problem and propose a novel rehearsal-free method. (2) We designed an interconnected global-local expert system, a new task representation, and introduced a soft regularization guided by actor advantage magnitude in training steps. (3) Our comprehensive analysis in a rehearsal-free CMARL setting shows the effectiveness of our method in challenging CMARL problems with discrete and continuous action spaces.

2 Related Works

(a) Multi Agent Reinforcement Learning: Several cooperative Multi-Agent Reinforcement Learning (MARL) algorithms based on policy-gradient methods have been proposed and evaluated across various multi-agent environments, including Multi-Agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al., 2017b) and Multi-Agent Proximal Policy Optimization (MAPPO) (Yu et al., 2022). These methods commonly adopt the Centralized Training with Decentralized Execution (CTDE) paradigm, where joint observations and actions are available only during training phase to construct a centralized critic. By incorporating global information, the critic can capture inter-agent dependencies, facilitate coordination, and mitigate the non-stationarity caused by simultaneous policy updates. To address heterogeneous multi-agent settings, methods such as Heterogeneous-Agent Trust Region Policy Optimization (HATRPO) (Kuba et al., 2022), Heterogeneous-Agent Proximal Policy Optimization (HAPPO) (Zhong et al., 2024), and Heterogeneous-Agent Soft Actor-Critic (HASAC) (Liu et al., 2024) have been developed. These approaches allow agents to maintain individual policies that can be specialized according to their distinct roles, capabilities, and observation or action spaces, while centralized critics can still be employed during training to facilitate global coordination. Another study, focused on developing a universal transformer based policy (Hu et al., 2021) to handle transferability between tasks in SMAC problem.

(b) Continual Multi-Agent Reinforcement Learning: COntinual RL Without Conflict (OWL) (Kessler et al., 2022) applied a regularization method to a Q-MIX (Rashid et al., 2018) policy and utilized a multi-armed bandit to select the best head for the current task. Similarly, Multi-agent Continual Coordination via Progressive Task Contextualization (MACPRO) utilized an attention-based architecture, regularization, and selected the best classification head based on a task contextualization mechanism. Another study was focused on developing a framework and benchmarking for continual MARL environments (Tomilin et al., 2026). The mentioned studies reported that native continual learning methods such as EWC Kirkpatrick et al. (2017) and MAS (Aljundi et al., 2018) can not fully handle catastrophic policy forgetting for the multi-agent systems. The concept in rehearsal-based methods such as knowledge distillation , experience replay, as in ICARL (Rebuffi et al., 2017), GDUMB (Prabhu et al., 2020), and XDER (Boschini et al., 2022), and growing sub-network mechanism as in BiC (Wu et al., 2019) and LUCIR (Hou et al., 2019) is relevant to rehearsal-based methods such as MACPRO. However, distillation can not be conducted via logits as CMARL learns policy, not class discrimination.

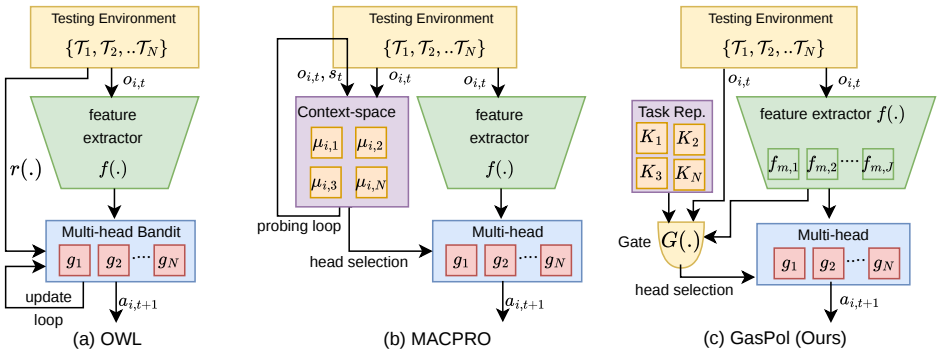

Figure 2: Visualization of the existing CMARL methods, i.e., OWL (a) and MACPRO (b), and the crucial difference with our proposed method (GASPOL) (c).

3 PRELIMINARY

(a) Problem Formulation: In MARL, a task $\mathcal{T}$ is formulated as a decoupled partially observable Markov decision process (Dec-POMDP) tuple $\langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, \mathcal{O}, r, \gamma \rangle$, where $\mathcal{I} = \{1, ..., i\}, \mathcal{S}, \mathcal{A}, (\mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2) \times ... \times \mathcal{A}^i), \Omega$ are the set of agent, states, joint actions, and local observations, respectively. Symbol $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ represents the transition probability function, $\mathcal{O} : \mathcal{S} \times \mathcal{I} \rightarrow \Omega$ represents corresponding observation,$\nabla : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ represents reward function, and $\gamma \in [0, 1)$ represents the discount factor. At each time step, agent i, gathers its local observation o^i, produces an action $a^i \in \mathcal{A}^i$. The objective of each agent is to maximize the cumulative discounted reward $\mathbb{E}[\Sigma_{t=0}^{\infty} \gamma^t r(s_t, a)]$ by learning an optimal joint policy. Continual multi-agent reinforcement learning (CMARL) is defined as the problem of learning a sequence of Dec-POMDP tasks $\{\mathcal{T}_n\}_{n=1}^{N}$, where each task $\mathcal{T}_n$ is formulated as the aforementioned Dec-POMDP tuple. After finishing learning a task $\mathcal{T}_n$, the multi-agent system need to recognize all learned tasks, i.e., $\mathcal{T}_1$,$\mathcal{T}_2$,...$\mathcal{T}_{n-1}$,$\mathcal{T}_N$, where the symbol N represents the number of consecutive tasks. The objective of CMARL is to maximize the average of cumulative discounted reward across all tasks, i.e., $\frac{1}{N}\Sigma_{n=1}^{N} R_n$, where $R_n = \mathbb{E}[\Sigma_{t=0}^{\infty} \gamma^t r(s_t, a)]$

(b) Existing CMARL Methods and their Drawbacks: Figure 2 (a) and (b) show inference process for the existing CMARL methods, i.e., MACPRO (Yuan et al., 2024) and OWL(Kessler et al., 2022) respectively. The figure shows, that OWL requires environment reward in the inference phase. OWL trains each head g_i separately during the training phase of each task, while the Bandit head selector is tuned in the evaluation phase. This type of mechanism is unrealistic, as during the evaluation phase, environment reward is available only for performance measurement, not for tuning the trained policy network. The second sub-figure shows how MACPRO works in the inference phase. Instead of environment reward, each agent in MACPRO performs a probing mechanism, gathering P trajectories to select the closest task representation associated with an action layer, where P is the number of probing episodes, and the trajectories include global trajectory (state) s_t, and local trajectory (observation) $o_{i,t}$. Although not utilizing the environment's reward as feedback, the probing mechanism takes P steps of states and all agent observations before selecting the action head g_i. This mechanism is less realistic, as the action needs to be taken immediately once the testing environment is started. In a multi-agent system with the CTDE paradigm, the state s_t is practically available only in the training phase. In the evaluation phase, each agent gets its own observation $o_{i,t}$, which is a part of the state s_t. The local observation $o_{i,t}$ may be unsharable; thus, the assumption that each agent get state s information during evaluation is not always applicable. In addition, MACPRO saves learned trajectories, i.e., states $s_0, s_1, s_2, ..., s_T$ and agent observations $o_0^i, o_1^i, ..., o_T^i$ into buffer memory $\mathcal{M}$, where $t \in [1..T]$ denotes the environment step from all tasks. This memory utilization is not always applicable since the data openness policy may prohibit saving trajectories. Besides, different ranges of reward values between tasks can harm the policy model. Reflecting to these drawbacks, we design a rehearsal-free CMARL method where the head can be selected immediately during the evaluation phase via a gating mechanism, as shown in Figure 2 (c).

4 PROPOSED METHOD

We designed a novel Dual Gating Sub-networks policy (GASPOL) that incorporates an interconnected dual-expert system, i.e., global experts and local experts as sub-networks for preserving general knowledge (stability) of all learned tasks and achieving task-specific knowledge (plasticity),

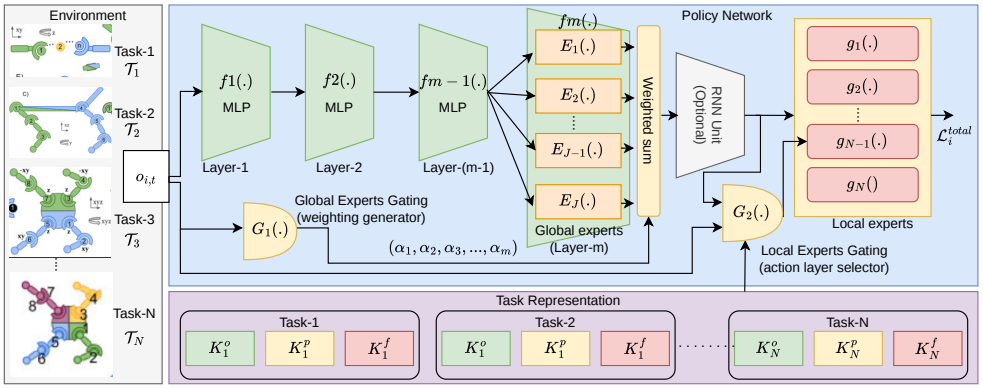

Figure 3: Visualization of the proposed Dual Gating Sub-networks Policy (GASPOL). GASPOL expands one of the sub-networks as a mixture of global experts, gated by an MLP gating $G_1(.)$, and expands the action layer as a set of local experts, gated by a similarity-based gating function $G_2(.)$

respectively. GASPOL introduces a new actor advantage-guided regularization mechanism to scale the regularization loss based on the current and old policies for less sensitive updates. We design our proposed method on top of a heterogeneous actor-critic structure, where each agent trains its local (actor) network, while sharing the centralized critic network. We adopt a growing sub-network mechanism as in existing CMARL and CL methods, by incrementally growing the action layer. We train our proposed solution in a rehearsal-free manner.

4.1 Network Architecture and Trainable Parameters

(a) Global-Local Experts: As shown in Figure 3, we design dual gating sub-networks structure with a global expert set and a local expert set. Suppose that a policy network of an agent π_θ is decomposed as a sequence of m sub-networks and an action layer as $\pi_\theta = g_\psi(fm_{\theta m}(...(f2_{\theta 2}(f1_{\theta 1}(.)))))$ where $f1(.), f2(.), ...fm(.)$ are the layers of the policy network, parameterized by $\theta 1, \theta 2, ...\theta m$, respectively, and $g(.)$ is parameterized by ψ is action layer of the network. We expand $fm(.)$ and $g(.)$ as gated global and local experts, respectively, with gating networks $G_1(.)$ and $G_2(.)$. The global expert can now be represented as $fm(.) = E^G(.) = [E_1^G(.), E_2^G(.), ...E_J^G(.)]$, where J is the number of global experts. Global experts are designed to learn all task representations; thus, the sub-network will be trained on all tasks. For that purpose, we design the experts combined by a simple mixture of experts (Jordan & Jacobs, 1994), with an MLP-based gating network, where the output of the layer is computed by equation 1.

$$fm(.) = \sum_{j=1}^{J} \alpha_j . E_j^G(.), \text{ where, } (\alpha_1 . \alpha_2, ...\alpha_J) = G_1(o_{i,t}) = MLP(o_{i,t}) \tag{1}$$

The local expert set is designed as a set of action layers (heads), where each local expert represents the respective task. During execution of the forward function, only one of the local experts is selected to get the action logits for a given observation input. Formally, the action layer $g(.)$ is now expanded into $[g_1(.), g_2(.), ...g_N(.)]$. Since the set is built gradually along with the respective arrival, we design the gating network for local experts $G_2(.)$ as a switch router based on input-reference similarity. The local gating network is a routing function with 3 inputs, i.e., agent observation $o_{i,t}$, a feature formed from the observation $f_{i,t} = f_m(...(f_2(f_1(o_{i,t}))))$, and a set of task representations $\{K_n\}_{n=1}^N$, where N, is the number of learned tasks.

(b) Task Representation and Local Expert Routing: A task representation K_n for task n is a tuple of trainable parameters $K_n = (K_n^o, K_n^p, K_n^f)$, where, K_n^o, K_p^p and K_n^f are dedicated to learn the agent observation space $\{o_{i,t}\}_{t=0}^\infty$, projection space of the observation $\{p(o_{i,t})\}_{t=0}^\infty$, and feature space $\{f_{i,t}\}_{t=0}^\infty$, respectively. $p(o_{i,t})$ denotes the projected observation vector by a non-trainable projection function $p(.)$. The tuple of task representations is designed to achieve better task discrimination rather than using only a single trainable vector for key matching as in previous studies (Wang et al., 2022b;a). The local gating network $G_2(.)$ selects the action layer $g_{\hat{n}}$ by performing Eq. 2

$$\begin{aligned} \hat{n} = G_2(o_{i,t}, f_{i,t}, \{K_n\}_{n=1}^N) &= \text{argmax}\{(s_n^o \times s_n^f) + \beta^p s_n^p\} \text{ for } 1 \le n \le N, \text{ where} \\ s^o &= Softmax([cos(K_1^o, o_{i,t}), cos(K_2^o, o_{i,t}), ..., cos(K_N^o, o_{i,t})]) \\ s^f &= Softmax([cos(K_1^f, f_{i,t}), cos(K_2^f, f_{i,t}), ..., cos(K_N^f, f_{i,t})]) \\ s^p &= [sim(K_1^p, p(o_{i,t})), sim(K_2^p, p(o_{i,t})), ..., sim(K_N^p, p(o_{i,t}))] \end{aligned} \tag{2}$$

where $cos()$ denotes cosine similarity function, and $sim(.)$ denotes Euclidean based similarity, i.e. $sim(a, b) = P_{max} - ||a - b||_2$, where P_{max} is pre-defined maximum value, and β is a temperature constant. The projection similarity s_n^p in Equation 2 plays a safeguard role when two different tasks produce the same $(s_n^o \times s_n^f)$ value. After selecting the local expert, the agent produces an action $a_{i,t}$ by performing $a_{i,t+1} = g_{\hat{n}}(f_{i,t})$.

4.2 Learning Mechanism

(a) Learning Policy: We build GASPOL on top of an on-policy MARL algorithm named Heterogeneous-Agent Proximal Policy Optimisation (HAPPO) (Zhong et al., 2024) due to several considerations. First, as a family of on-policy algorithms, the method trains the same policy network both for exploration and exploitation, resulting a faster training process than off-policy algorithms such as QMIX (Rashid et al., 2018). HAPPO offers a realistic heterogeneity assumption where different roles require different policy models, and the agents do not share their locally trained policy model. To train the policy model π_{θ_i}, we adopt the HAPPO algorithm where heterogeneous agents train their local actor network based on the ordered update, leveraging allied agents' experience Zhong et al. (2024). It uses individual advantage functions and decentralized policies while maintaining centralized critics. In HAPPO, the centralized value function is computed in an agent-specific manner, guiding each heterogeneous agent differently from others, defined as $V_i^{\text{ctr}}(s) \approx \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{i,t} \mid s_0 = s\right]$. where i denotes the agent index and $r_{i,t}$ is the reward specific to agent i. The policy learning in HAPPO is updated by the following loss function:

$$\mathcal{L}_i^{\text{pol}} = \mathbb{E}_t \left[\min \left(r_{i,t}(\theta_i) \hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta_i), 1-\epsilon, 1+\epsilon) \hat{A}_{i,t} \right) \right], \tag{3}$$

where $r_{i,t}(\theta_i) = \frac{\pi_{\theta_i}(a_{i,t}|o_{i,t})}{\pi_{\theta_i}^{\text{old}}(a_{i,t}|o_{i,t})}$ and $\hat{A}_{i,t}$ are the advantage functions of agent i. It follows the multi-agent decomposition lemma, and it has a sequential update mechanism denoted as in equation 4, where k is the number of agents, in the sequence update of HAPPO.

$$\hat{A}_{i_{1:k},t}(o_t, a_{i_{1:k},t}) = \sum_{j=1}^{k} \hat{A}_{i_j}\left(o_t, a_{i_{1:j-1},t}, a_{i_j}\right). \tag{4}$$

(b) Learning Task Representations: Distinct task representations are an important component in GASPOL, since it determines the suitable local expert $g_n(,)$ to generate actions. On each iteration of task n training, the task representation tuple K_n^o, K_n^p and K_n^f are adjusted to be closer to $o_{i,t}$, $p(o_{i,t})$ and $f_{i,t}$, respectively. Thus, the task-associated parameters represent the observation, projected observation, and feature spaces of task n. To achieve such objectives, we train K_n^o and K_n^f parameters using negative cosine similarity loss. As intended to represent the task-n feature space $\{f_{i,t}\}_{t=0}^{\infty}$, where $f_{i,t} = f_m(...(f_2(f_1(o_{i,t}))))$ is computed by the trainable sub-networks $(\theta 1, \theta 2, ...\theta m)$, K_n^f has more leverage to separate task representation than K_n^o. Thus, we add an orthogonality (dot product) loss function to separate K_n^o and $K_{n'}^o$, $n' \in [1, 2, .., n-1]$. Therefore, learning mechanism for updating K_n^o and K_f^o are presented in equation 5.

$$\mathcal{L}_i^{K^o} = -cos(o_{i,t}, K_n^o), \mathcal{L}_i^{K^f} = -cos(f_{i,t}, K_n^f) + dot(K_n^f, K_{n'}^f) \tag{5}$$

Different from $cos(K_n^f, f_{i,t})])$ and $cos(K_n^f, f_{i,t})])$ that are bounded in a fixed range, i.e., $[-1, 1]$, as a safeguard component, $sim(K_n^p, p(o_{i,t}))$ has unrestricted value range. Thus, we update the value of K_n^p with a Euclidean distance-based loss toward $p(o_{i,t})$. However, to achieve a smoother projection and similarity s^p measurement, we perturbed the observation data before the projection, and the update process for K_n^p is guided by the loss $mathcal L_i^{K^p}$ as presented in equation 6.

$$\mathcal{L}_i^{K^p} = ||p(o_{i,t} + \epsilon) - K_n^f||_2, \text{where}, \epsilon \sim \mathcal{N}(\mu, \sigma^2) \tag{6}$$

(c) Soft Routing Strategy: We utilize a soft routing strategy during the training phase on task n, where the selected head $g_{\hat{n}}(.)$ is selected by the gating mechanism as in equation 2, even though $K_n = (K_n^o, K_n^p, K_n^f)$ has not converged yet, and the task-ID is visible. Thus, it may select the wrong head in the initial steps of the training phase on task n. Therefore, to guide the local expert gating $G_2()$ in selecting the correct head, we add a small penalty amount $\mathcal{L}^{pen}$ if the model selects the wrong head during the training phase. This mechanism differs from the head training in OWL, where the method forcefully trains the head of the current task $g_{n(.)}$

(d) Advantage-Guided Regularization: We introduce a new advantage-guided regularization method, i.e., scaled weight importance by a function of advantage. Suppose that θ_i is the parameter

of agent-i policy model π_i, and θ_i^F is the Fisher weight importance of θ_i, which is computed cumulatively on each task. Applying normal regularization such as $||\theta_i - \theta_i^F||^2$ on each iteration will add a significant loss magnitude. To accommodate a policy network that is very sensitive, we propose a more flexible regularizer that is formulated in Equation 7.

$$\mathcal{L}_i^{reg} = l(\hat{A}_i)||\theta_i - \theta_i^F||^2 \tag{7}$$

where $l()$ is the scaling function based on the actor advantage value. In this study, we implement $l()$ as the difference between logits of the current policy model and logits of the old policy model, i.e., $l() = log(\pi_{\theta_{i,t}}) - log(\pi_{\theta_{i,t}}^{old})$. In that way, the model will be regularized in a more proportional way, based on the current-to-old policy difference.

(e) Final Objective and Loss Function: Accommodating the aforementioned learning objectives, we integrate them into the total loss function as in Equation 8. The user-defined loss coefficients is added $\{\lambda_1, \lambda_2, \lambda_3\}$ for a more flexible loss weighting.

$$\mathcal{L}_i^{total} = \mathcal{L}_i^{pol} + \lambda_1 \mathcal{L}_i^{K_n^o} + \lambda_2 \mathcal{L}_i^{K_n^f} + \lambda_3 \mathcal{L}_i^{reg} + \mathcal{L}^{pen} \tag{8}$$

5 Theoretical Analysis

We state the following theorem to prove the effectiveness of our proposed method.

Theorem 1: *A sequence $(\pi)_{t=0}^{\infty} = (\Sigma_{j=1}^{J} \alpha_j E_j)_{t=0}^{\infty}$ of joint policies by a gated policy network, updated by HAPPO algorithm, has the monotonic improvement property, i.e., $W(\pi_{t+1}) \geq W(\pi_t)$, for all $t \in \mathbb{N}$, and W denotes the joint policy function.*

Theorem 2: *Let G and G' denote the local expert gating functions parameterized by $(\{K_n^o\}, \{K_n^f\})$ and $\{K_n^f\}$ respectively, and $cos(,)$ denote cosine similarity function, and L denote local expert selection error. If K_n^o and K_n^f are trained well enough so that $cos(o_{i,t}, K_n^o) \approx cos(o_{i,t}, K_n^f) \geq C$, and $cos(o_{i,t}, K_{n'}^o) \approx cos((f_{i,t}, K_{n'}^f) \approx cos((o_{i,t}, K_{n'}^f) \leq 2/3C$ for any $o_{i,t} \in \mathcal{O}$ then it guarantees $L(G, o_{i,t}, f_{i,t}) \leq L(G', o_{i,t}, f_{i,t})$.*

while Theorem 1 theoretically guarantees of policy improvement during the training phase, Theorem 2 shows that if the learnable parameters are well-trained then it guarantees our local expert router, as presented in equation 2 is more accurate than a router that is based on $\{K_f^o\}$ only. Please see Appendix 5 for the detailed analysis.

6 Experimental Result and Analysis

6.1 Experimental Setting

(a) Problem: We evaluate our method in three popular continual MARL problems i.e. SMAC Marine, and SMAC SZ(Samvelyan et al., 2019), Pettingzoo MPE2 (Lowe et al., 2017a) and Multi agent MuJoCo(Peng et al., 2021). SMAC games have a discrete action space, while the rest have continuous action spaces. The problems are set into 4 sequential games with a rehearsal-free constraint. The SMAC Marine and SMAC SZ utilize StarCraft II games with marines and stalker-zealots maps, respectively. For SMAC Marine, the task consists of 3m, 5m, 7m, and 8m games, while for SMAC SZ, the task consists of 1s2z, 2s2z, 2s3z, and 2s4z, games. MAMuJoCo carries robotic control tasks with different configurations, i.e., Swimmer-v2(2x1), HalfCheetah-v2(2x3), Ant-v2(2x4), and Ant-v2(4x2). Pettingzoo MPE2 carries 4 different tasks for multi-agent particle environments, i.e., simple_reference_v3, simple_speaker_listener_v4, simple_line_v1, and simple_formation_v1.

(b) Baselines and Performance Metrics: We compare our proposed method with existing continual MARL methods such as OWL (Kessler et al., 2022), MACPRO (Yuan et al., 2024), combination of HAPPO Zhong et al. (2024) and existing rehearsal-free CL method such as EWC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017), and MAS (Aljundi et al., 2018). We also compared the proposed method with ORACLE and ORACLE-Full that are task-informed versions of the growing head and growing full network, respectively. For MACPRO, we run it only for SMAC games, since it does not provide implementations for MPE2 and MAMuJoCo. After finishing a task $\mathcal{T}^n$, we measure the average success rate (SR), which covers all learned tasks $(\mathcal{T}^1, \mathcal{T}^2, ..\mathcal{T}^n)$, final average success rate (FSR) that is the same as SR on the last task; the final forgetting measure (FFM), that represent the final performance after learning all tasks; and the cumulative success rate (CSR), that represent cumulative historical performance on each task. For SMAC games, success rate is measured by winning rate, while for other problems, it is measured by the normalized reward.

Table 1: Overall performance of consolidated methods on 4 problems, i.e., SMAC Marine, SMAC SZ, Pettingzoo MPE2, and MAMuJoCo

Method	SMAC Marine			SMAC SZ		
	FSR	CSR	FFM	FSR	CSR	FFM
ORACLE-Full	97.5 ± 2.17	97.22 ± 1.14	2.78 ± 0.96	69.17 ± 0.72	72.43 ± 0.66	2.22 ± 1.67
ORACLE	56.25 ± 15.16	74.27 ± 11.73	53.89 ± 19.25	27.08 ± 12.27	45.31 ± 8.84	49.44 ± 6.74
FT	67.50 ± 6.50	81.67 ± 1.98	41.11 ± 6.74	37.50 ± 2.17	70.21 ± 0.54	76.67 ± 2.89
EWC	60.75 ± 15.17	79.15 ± 4.06	42.69 ± 17.75	63.60 ± 8.36	75.40 ± 8.63	33.92 ± 10.96
SI	63.42 ± 13.82	72.07 ± 13.86	45.41 ± 17.38	35.96 ± 10.64	60.01 ± 6.26	67.84 ± 11.41
MAS	70.02 ± 3.35	83.47 ± 6.63	31.75 ± 7.05	43.86 ± 5.32	71.86 ± 5.58	60.26 ± 1.75
OWL	65.83 ± 18.76	82.50 ± 8.20	42.78 ± 25.98	63.89 ± 6.74	49.17 ± 7.64	2.50 ± 5.00
MACPRO	47.81 ± 0.01	74.97 ± 0.05	1.05 ± 1.09	55.47 ± 0.00	68.17 ± 0.00	4.59 ± 0.00
GASPOL	95.18 ± 0.76	96.67 ± 0.23	5.26 ± 2.03	75.44 ± 9.96	84.87 ± 8.50	20.47 ± 13.63
	Pettingzoo MPE2			MAMuJoCo		
	FSR	CSR	FFM	FSR	CSR	FFM
ORACLE-Full	80.27 ± 3.18	82.59 ± 2.64	22.12 ± 1.7	81.93 ± 3.15	90.96 ± 3.98	-4.25 ± 3.50
ORACLE	67.19 ± 1.66	76.34 ± 4.08	28.21 ± 8.85	48.05 ± 11.44	64.6 ± 6.12	14.25 ± 6.71
FT	56.73 ± 4.00	72.21 ± 2.11	26.46 ± 3.84	44.72 ± 8.87	60.49 ± 6.67	41.01 ± 12.43
EWC	61.07 ± 3.85	72.10 ± 2.77	27.79 ± 11.07	40.68 ± 11.57	63.00 ± 3.11	36.02 ± 12.59
SI	59.76 ± 6.94	73.85 ± 3.47	22.34 ± 6.03	38.38 ± 2.55	54.6 ± 9.36	52.68 ± 2.38
MAS	61.07+3.85	72.10+2.77	27.79+11.07	46.90 ± 9.71	64.08 ± 3.26	43.42 ± 4.00
OWL	50.92 ± 2.71	71.21 ± 1.58	23.76 ± 3.35	53.01 ± 10.29	67.01 ± 4.31	15.73 ± 16.00
GASPOL	73.10 ± 1.00	77.68 ± 1.41	24.01 ± 2.93	66.51 ± 4.25	80.99 ± 0.44	15.44 ± 3.34

Table 2: Historical average success rate of learned tasks of consolidated methods on 4 problems, i.e., SMAC Marine, SMAC SZ, Pettingzoo MPE2, and MAMuJoCo

Method	SMAC Marine				SMAC SZ			
	Task-1	Task-2	Task-3	Task-4	Task-1	Task-2	Task-3	Task-4
ORACLE-Full	96.67 ± 2.89	99.17 ± 1.44	95.56 ± 0.96	97.50 ± 2.17	100 ± 0.00	51.67 ± 1.44	68.89 ± 1.92	69.17 ± 0.72
ORACLE	96.67 ± 2.89	77.50 ± 8.66	66.67 ± 20.21	56.25 ± 15.16	100 ± 0.00	22.5 ± 17.32	31.67 ± 5.77	27.08 ± 12.27
FT	96.67 ± 2.89	89.17 ± 10.10	73.33 ± 8.66	67.50 ± 6.50	100 ± 0.00	86.67 ± 7.22	56.67 ± 11.55	37.50 ± 2.17
EWC	94.74 ± 5.26	90.35 ± 6.62	70.76 ± 6.64	60.75 ± 15.17	98.25 ± 3.04	77.19 ± 14.49	62.57 ± 16.67	63.60 ± 8.36
SI	94.74 ± 5.26	74.56 ± 22.99	55.56 ± 19.25	63.42 ± 13.82	98.25 ± 3.04	59.65 ± 11.87	46.20 ± 2.03	35.96 ± 10.64
MAS	94.74 ± 5.26	95.61 ± 1.52	73.51 ± 16.71	70.02 ± 3.35	98.25 ± 3.04	82.76 ± 8.14	62.57 ± 20.93	43.86 ± 5.32
OWL	98.33 ± 2.89	85.83 ± 12.58	80.00 ± 13.64	65.83 ± 18.76	98.33 ± 2.89	95.00 ± 8.66	65.83 ± 27.42	63.89 ± 6.74
MACPRO	98.12 ± 0.00	90.40 ± 0.20	63.54 ± 0.02	47.81 ± 0.01	99.38 ± 0.00	52.19 ± 0.00	65.63 ± 0.00	55.47 ± 0.00
GASPOL	98.25 ± 3.04	99.12 ± 1.52	94.15 ± 2.03	95.18 ± 0.76	94.74 ± 9.12	88.6.00 ± 8.46	80.70 ± 14.99	75.44 ± 9.96
	Pettingzoo MPE2				MAMuJoCo			
	Task-1	Task-2	Task-3	Task-4	Task-1	Task-2	Task-3	Task-4
ORACLE-Full	90.00 ± 5.60	80.76 ± 2.53	79.32 ± 3.09	80.27 ± 3.18	92.84 ± 6.32	96.07 ± 2.64	92.98 ± 4.86	81.93 ± 3.15
ORACLE	90.00 ± 5.60	77.49 ± 8.04	70.65 ± 3.75	67.19 ± 1.66	92.84 ± 6.32	63.21 ± 10.45	54.31 ± 3.92	48.05 ± 11.44
FT	89.97 ± 5.61	79.09 ± 5.00	63.04 ± 2.47	56.73 ± 4.00	92.84 ± 6.32	53.25 ± 30.49	51.14 ± 11.88	44.72 ± 8.87
EWC	90.00 ± 5.60	74.38 ± 1.57	62.96 ± 8.26	61.07 ± 3.85	92.84 ± 6.32	64.47 ± 6.18	54.01 ± 1.73	40.68 ± 11.57
SI	92.65 ± 6.24	75.53 ± 3.70	67.44 ± 6.68	59.76 ± 6.94	92.84 ± 6.32	34.85 ± 31.82	52.34 ± 9.75	38.38 ± 2.55
MAS	90.00 ± 5.60	74.38+1.57	62.96 ± 8.26	61.07+3.85	97.05 ± 3.91	64.86 ± 11.91	47.53 ± 4.69	46.90 ± 9.71
OWL	90.00 ± 5.60	75.88 ± 2.01	68.02 ± 1.38	50.92 ± 2.71	93.12 ± 5.38	58.05 ± 5.82	63.86 ± 11.83	53.01 ± 10.29
GASPOL	89.38 ± 2.53	75.57 ± 2.62	72.69 ± 1.86	73.10 ± 1.00	86.65 ± 4.62	87.23 ± 1.78	83.56 ± 5.4	66.51 ± 4.25

(c) Implementation Details: We implemented GASPOL in Pytorch utilizing the 3-layered MLP+RNN backbone following the common backbone for MARL. We utilize the Adam optimizer, set a 200-episode length, 10 rollouts, 500K steps of training, and the same learning rate decay for all methods. The initial learning rate is set to 0.005 both the actor and critic networks. All the methods are run under the same environment, i.e., a single workstation with a 64-core AMD EPYC 7742 CPU and 48 GB RAM. Please see Appendix for the detailed metrics and settings.

6.2 Result and Analysis

(a) Overall Performance Table 1 shows the summarized numerical results of consolidated algorithms in 4 MARL problems. The table shows that the proposed method performs excellently, only second to ORACLE-Full that grows a full backbone and is able to access the task ID. GASPOL achieves a remarkable performance compared to the existing methods with more than 20% FSR and 10% CSR in SMAC Marine. GASPOL significantly outperforms ORACLE in this game, even though ORACLE can access the task ID during the evaluation phase. From forgetting measurements, GASPOL achieves relatively low forgetting in the SMAC Marine game, slightly higher than MACPRO and ORACLE-Full. Despite achieving the lowest forgetting, MACPRO achieves relatively low FSR and CSR. It shows that it can not achieve plasticity and tends to maintain stability. The regularization methods can not handle catastrophic forgetting in this task, as shown by the forgetting measurements of the methods. In the SMAC SZ games, the proposed method, once again,

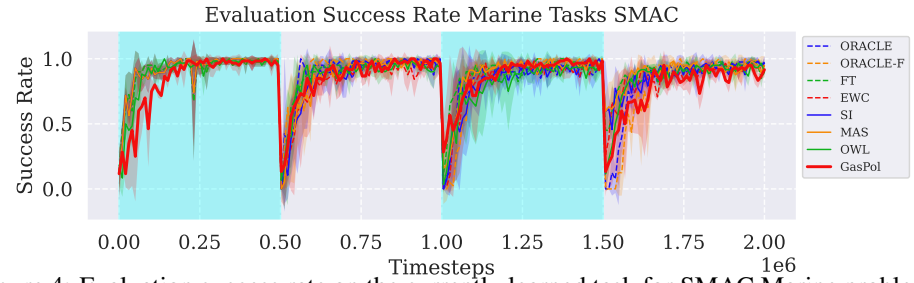

Figure 4: Evaluation success rate on the currently learned task for SMAC Marine problem.

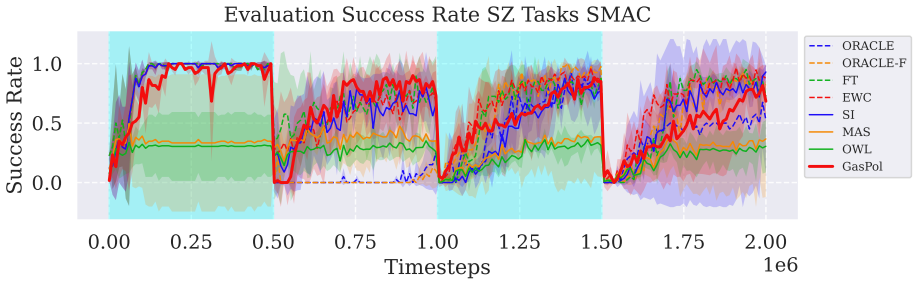

Figure 5: WEvaluation success rate on the currently learned task for SMAC SZ problem.

achieves excellent performance, and even significantly outperforms ORACLE-Full on CSR metrics. From the FSR perspective, GASPOL outperforms the existing methods with a convincing margin, i.e., more than 10%. From forgetting measurement metrics, GASPOL experiences a lower forgetting rate than the regularization methods, but it is still higher than MACPRO and ORACLE-Full. Similarly, MACPRO achieves a relatively low FSR and CSR while successfully keeping its low forgetting rate. Despite having the advantage of task ID information, ORACLE performs poor in this problem. In the Pettingzoo MPE2 problem, GASPOL achieves an excellent performance that is second only to ORACLE-Full. Except for ORACLE, it outperforms the other methods with a significant margin, i.e., more than 12% and 3% for FSR and CSR, respectively. In term of FSR, it still has a better performance than ORACLE with around 6% margin. Similarly, in MAMuJoCo, GASPOL achieve the highest performance except for ORACLE-Full with a convincing gap, i.e., more than 13% and 13.5% for FSR and CSR, respectively. In addition, GASPOL achieves relatively low forgetting in this problem, slightly higher than ORACLE and lower than the other methods.

(b) Historical Performance: Table 2 shows the detailed capability of the proposed method in maintaining its stability while achieving plasticity. For instance, in SMAC marine games, GASPOL does not suffer from a performance drop from task-1 to task-2. GASPOL experiences a slight performance drop after finishing task-3, but on the last task, its performance increasing despite learning a more complex game than before. It shows that the idea of general experts play an important role in continual MARL settings. Similarly, in the SMAC SZ game, the proposed method experiences the smallest performance drop from task-1 to task-4, while the existing methods suffer from more than 30% performance drop. It shows that despite the task sequence being less similar compared to the previous game, the proposed method still can mitigate forgetting and maintain a better stability-plasticity balance. In Pettingzoo MPE2, GASPOL manages to handle the stability-plasticity trade-off, as shown by its smallest performance drop between the first and the last task. It experiences a 16% performance drop, while the other Non-ORACLE methods suffer from more than a 28% drop. In MAMuJoCo, GASPOL does not experience a performance drop on the second task and has a small amount, i.e., 3% in the third task. Despite experiencing a higher drop, GASPOL manages to achieve the highest success rate in the last task, except for ORACLE-Full. These results show the effectiveness of the proposed method in tackling catastrophic forgetting for CMARL problems with discrete and continuous action spaces.

(c) Adaptability Analysis: Figures 4 and 5 show the performance of the consolidated methods in on the currently learned task. The figure shows performance plots of four sequential games within each problem. The figure shows that, in both problems, all methods start from zero upon arrival of a new task. In SMAC marine games, all the methods are able to achieve their best performance on the currently learned task. However, looking back to the Table 1 and 2, where the methods

suffer from performance drops in the latest task, it shows that the models cannot maintain their previously learned policy while adapting to the current task. In the second game, OWL, MAS, SI, and ORACLE can not adapt to the new task properly. Their low performance on the currently learned task shows that the method can not consolidate their knowledge on two or more different environment in a rehearsal-free manner. Looking at GASPOL performance within the plots, we see that despite having a slower start to learn a new task, it can achieve a relatively high reward value at the end of the training steps. Combined with the previous analysis from Table 1 and 2, where the proposed method successfully achieves the highest performance in the last task, and an a cumulative view, the proposed method's slow adaptation is reasonable.

Table 3: Ablation study on SMAC Marine problem

Component	Loss	FSR	CSR	FFM
FT	$\mathcal{L}_i^{pol}$	40.00	69.58	55.28
$FT + \{E_j\} + G_1$	$\mathcal{L}_i^{pol} + \mathcal{L}_i^{K^o}$	57.50	76.67	50.83
$FT + \{E_j\} + G_1 + \{g_n\} + K_n^o + G_2$	$\mathcal{L}_i^{pol} + \mathcal{L}_i^{K^o}$	72.50	80.63	22.28
$FT + \{E_j\} + G_1 + \{g_n\} + K_n^o + G_2 + K_n^f + K_n^p$	$\mathcal{L}_i^{pol} + \mathcal{L}_i^{K^o} + \mathcal{L}_i^{K^f} + \mathcal{L}_i^{K^p}$	75.81	83.15	19.37
$FT + \{E_j\} + G_1 + \{g_n\} + K_n^o + G_2 + K_n^f + K_n^p$	$\mathcal{L}_i^{pol} + \mathcal{L}_i^{K^o} + \mathcal{L}_i^{K^f} + \mathcal{L}_i^{K^p} + \|\|\theta - \theta^F\|\|^2$	86.25	95.52	3.60
$FT + \{E_j\} + G_1 + \{g_n\} + K_n^o + G_2 + K_n^f + K_n^p + l(\hat{A})$	$\mathcal{L}_i^{pol} + \mathcal{L}_i^{K^o} + \mathcal{L}_i^{K^f} + \mathcal{L}_i^{K^p} + \mathcal{L}_i^{reg}$	92.50	96.25	0.28
$FT + \{E_j\} + G_1 + \{g_n\} + K_n^o + G_2 + K_n^f + K_n^p + l(\hat{A})$	$\mathcal{L}^{total}, \lambda_1, \lambda_2, \lambda_3$	97.50	98.33	-1.39

(d) Ablation Study: Table 3 shows the impacts of the components in pur proposed method in improving the model performance. The global expert $\{E_j\}$ and its gating network $G_1(.)$ improve the base fine-tuning method by 17% and 7% for FSR and CSR, respectively. Adding local experts $\{g_n\}$ as a dedicated action layer for each task, along with the second router $G_2(.)$ and trainable observation key K_n^o improves the performance by around 15% for both FSR and CSR. It also reduces the forgetting significantly. Adding a feature-based learnable key K_n^f and safeguard key K_n^p increase the method's success rate by around 3% for both FSR and CSR. Adding a Fisher-based regularizer $||\theta - \theta^F||^2$ significantly improves model performance and significantly reduces its forgetting. However, there is still a gap between FSR and CSR. Scaling the regularizer with the advantages-guided scaling function $l(\tilde{A})$ improves the methods' FSR and CSR. Adjusting the loss coefficient with $\lambda_1, \lambda_2, \lambda_3$ and adding the penalty $\mathcal{L}^{pen}$ into total loss $\mathcal{L}^{total}$ improve the method performance with 5% and 3% improvement for FSR and CSR respectively, and reduces its forgetting.

(e) Sensitivity Analysis: Table 4 shows the sensitivity analysis of our proposed method to the number of global experts and the number of neurons in the model's hidden layer. The table shows that our model is less sensitive to the number of experts than to the number of neurons. Both from FSR and CSR, the model performance swings around 3% due to variation in #Experts, while it can drop by more than 8% due to #Neurons. This event is aligned with the finding in a previous MARL study (Zhong et al., 2024) that stated each MARL game has different best parameters for #Neurons. As shown in Table 4, we found that the best #Experts is the same as the number of tasks (4). This finding is worth reinvestigating in the future.

Table 4: Sensitivity analysis of the proposed method to number of global experts and the number of neurons in hidden layer.

Sensitivity to #Experts				Sensitivity to #Neurons			
#Experts	FSR	CSR	FFM	#Neuron	FSR	CSR	FFM
2	94.74	97.23	1.86	32	82.90	90.24	12.28
3	94.87	95.59	3.33	64	98.65	96.60	3.51
4	97.50	98.33	-1.39	96	83.03	91.81	19.25
5	96.05	95.06	8.77	128	97.50	98.33	-1.39
6	94.75	92.43	9.51	256	82.89	90.24	9.12

7 Conclusion and Future Work

This study proposes a new method named Dual Gating Sub-networks Policy (GASPOL) for rehearsal-free continual multi-agent reinforcement learning. GASPOL introduces a dual expert system, i.e., global experts and local experts as a part of the trained policy network. Global preserve general knowledge (stability) of all learned tasks, while local experts obtain a task-specific knowledge (plasticity). GASPOL leverages a new actor advantage-guided regularization mechanism for a more flexible regularization at each training iteration. Our evaluation on MARL environments, i.e., SMAC Marine, SMAC SZ, Pettingzoo MPE2, and MAMuJoCo, shows that the proposed method significantly achieves better stability-plasticity than the existing state-of-the-art methods. Our results demonstrate the ability to adapt to continual task changes in both discrete and continuous actions. The results show that learning stability of continual MARL algorithms in the presence of randomness can still be strengthened in the future. Although it has been evaluated under several seeds, it remains sensitive to a randomization factor such as the initial agent's position. Moreover, it is intriguing to further investigate CMARL algorithm across different environments and different action spaces.

Author Contributions

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

AI use statement

In this work, we used AI tools only for grammar error checking and typo fixing. We have not used generative AI tools for generating ideas, code, and writing in this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Reproducibility statement

We provide comprehensive materials to reproduce our proposed GASPOL and raw experimental results. The main paper describes the problem formulation, architecture, and training and evaluation methodology, while the appendix provides the complete mathematical formulation, assumptions, and proofs underlying the theoretical analysis. The implementation details, including the model architecture, fine-tuning configurations, hyperparameters, and experimental settings, are provided in the appendices. We further provide GASPOL source code and the baseline methods (except for MACPRO), in https://anonymous.4open.science/r/ICLR2027_ID42421/. All experimental comparisons are conducted using the same computational environment, thus the reported results to be reproduced.

References

Rahaf Aljundi, F. Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5497–5512, 2022.

Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10707–10717. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 831–839, 2019.

Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. Updet: Universal multi-agent rl via policy decoupling with transformers. In *International Conference on Learning Representations*, 2021.

Siyi Hu, Mohamad A Hady, Jianglin Qiao, Jimmy Cao, Mahardhika Pratama, and Ryszard Kowalczyk. Adaptability in multi-agent reinforcement learning: A framework and unified review. *arXiv preprint arXiv:2507.10142*, 2025.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Same state, different task: Continual reinforcement learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7143–7151, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

JG Kuba, R Chen, M Wen, Y Wen, F Sun, J Wang, and Y Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *ICLR 2022-10th International Conference on Learning Representations*, pp. 1046. The International Conference on Learning Representations (ICLR), 2022.

Guangyi Liu, Suzan Iloglu, Michael Caldara, Joseph W Durham, and Michael M. Zavlanos. Distributionally robust multi-agent reinforcement learning for dynamic chute mapping. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=DqlN1yFyxN.

Jiarong Liu, Yifan Zhong, Siyi Hu, Haobo Fu, Qiang Fu, Xiaojun Chang, and Yaodong Yang. Maximum entropy heterogeneous-agent reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 57333–57374, 2024.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017a.

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

Muhammad Anwar Masum, Mahardhika Pratama, Savitha Ramasamy, Lin Liu, Habibullah Habibullah, and Ryszard Kowalczyk. Vision and language synergy for rehearsal free continual learning. In *International Conference on Learning Representations*, volume 2025, pp. 57222–57252, 2025.

Hadi Nekoei, Akilesh Badrinaaraayanan, Aaron Courville, and Sarath Chandar. Continuous coordination as a realistic scenario for lifelong learning. In *International Conference on Machine Learning*, pp. 8016–8024. PMLR, 2021.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. FACMAC: Factored multi-agent centralised policy gradients. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=wZYWwJvkneF.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 524–540. Springer, 2020.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pp. 4295–4304. Pmlr, 2018.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

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

Mark A Stephenson and Hanspeter Schaub. Bsk-rl: Modular, high-fidelity reinforcement learning environments for spacecraft tasking. In *75th International Astronautical Congress, Milan, Italy, IAF*, pp. 1186–1197, 2024.

Tristan Tomilin, Luka van den Boogaard, Samuel Garcin, Constantin Ruhdorfer, Bram Grooten, Fabrice Kusters, Yali Du, Andreas Bulling, Mykola Pechenizkiy, and Meng Fang. Meal: a benchmark for continual multi-agent reinforcement learning. *arXiv preprint arXiv:2506.14990*, 2025.

Tristan Tomilin, Luka van den Boogaard, Samuel Garcin, Constantin Ruhdorfer, Bram Grooten, Yali Du, Andreas Bulling, Mykola Pechenizkiy, and Meng Fang. MEAL: A benchmark for continual multi-agent reinforcement learning, 2026. URL https://openreview.net/forum?id=I3W8PynQU0.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pp. 631–648. Springer, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022b.

Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 35:16509–16521, 2022.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Lei Yuan, Lihe Li, Ziqian Zhang, Fuxiang Zhang, Cong Guan, and Yang Yu. Multiagent continual coordination via progressive task contextualization. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):6326–6340, 2024.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. Pmlr, 2017.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024.

A Theoretical Analysis

This section presents theoretical analysis proving Theorem 1 and Theorem 2.

Theorem 1: *A sequence* $(\pi)_{t=0}^{\infty} = (\Sigma_{j=1}^{J}\alpha_j E_j)_{t=0}^{\infty}$ *of joint policies by a gated policy network, updated by HAPPO algorithm, has the monotonic improvement property, i.e.,* $W(\pi_{t+1}) \geq W(\pi_t)$ *equal* $W((\Sigma_{j=1}^{J}\alpha_j E_j)_{t+1}) \geq W(\Sigma_{j=1}^{J}\alpha_j E_j)_t)$ *, for all* $t \in \mathbb{N}$*, and* W *denotes the joint policy function.*

Theorem 2: *Let* G *and* G' *denote the local expert gating functions parameterized by* $(\{K_n^o\}, \{K_n^f\})$ *and* $\{K_n^f\}$ *respectively, and* $cos(,)$ *denote cosine similarity function, and* L *denote local expert selection error. If* K_n^o *and* K_n^f *are trained well enough so that* $cos(o_{i,t}, K_n^o) \approx cos(o_{i,t}, K_n^f) \geq C$*, and* $cos(o_{i,t}, K_{n'}^o) \approx cos((f_{i,t}, K_{n'}^f) \approx cos((o_{i,t}, K_{n'}^f) \leq 2/3C$ *for any* $o_{i,t} \in \mathcal{O}$ *then it guarantees* $L(G, o_{i,t}, f_{i,t}) \leq L(G', o_{i,t}, f_{i,t})$.

A.1 Proof of Theorem 1

This proof refer to Theorem 7 and Theorem 16 in HAPPO (Zhong et al., 2024). Let π_0 be any joint policy, for each $t \geq 0$, the joint policy π_{t+1} is obtained by HAPPO update algorithm for agent $m = 1, ...I$ as follows:

$$\pi_{i_m,t+1} = \arg\max_{\pi_{i_m}} [\mathcal{L}_{i_{1:m},t+1}(\pi_{i_{1:m-1},t+1}, \pi_{i_m}) - CD_{KL}^{max}(\pi_{i_m,t}, \pi_{i_m})] \tag{9}$$

where $C = (4\gamma \max_{s,a}|A_\pi(s,a)|)/(1-\gamma)^2$. By Therem 16 in HAPPO, we get

$$W(\pi_{t+1}) \geq \mathcal{L}_{\pi_t}(\pi_{t+1}) - CD_{KL}^{max}(\pi_t, \pi_{t+1}) \tag{10}$$

Applying Lemma 18 lower bound in HAPPO, we get

$$W(\pi_{t+1}) \geq \mathcal{L}_{\pi_t}(\pi_{t+1}) - \sum_{m=1}^{I} CD_{KL}^{max}(\pi_{i_m,t}, \pi_{i_m,t+1}) \tag{11}$$

$$W(\pi_{t+1}) = W(\pi_t) + \sum_{m=1}^{I} (\mathcal{L}_{i_{1:m},\pi_t}(\pi_{i_{1:m-1},t}, \pi_{i_m,t+1}) - CD_{KL}^{max}(\pi_{i_m,t}, \pi_{i_m,t+1})) \tag{12}$$

For each m, $\pi_{i_m,t+1}$ is the argmax, therefore, the expression above becomes

$$W(\pi_{t+1}) \geq W(\pi_t) + \sum_{m=1}^{I} (\mathcal{L}_{i_{1:m},\pi_t}(\pi_{i_{1:m-1},t}, \pi_{i_m,t+1}) - CD_{KL}^{max}(\pi_{i_m,t}, \pi_{i_m,t+1})) \tag{13}$$

Following the Denition 5 in HAPPO where is:

$$\sum_{m=1}^{I} (\mathcal{L}_{i_{1:m},\pi_t}(\pi_{i_{1:m-1},t}, \pi_{i_m,t+1}) - CD_{KL}^{max}(\pi_{i_m,t}, \pi_{i_m,t+1})) = \sum_{m=1}^{I} 0 \tag{14}$$

Thus The inequation above is derived into

$$W(\pi_{t+1}) \geq W(\pi_t) + \sum_{m=1}^{I} 0 = W(\pi_t) \tag{15}$$

$$W(\pi_{t+1}) \geq W(\pi_t) \tag{16}$$

Since we deploy a gated network as policy network model, i.e., $\pi = (\Sigma_{j=1}^{J} \alpha_j E_j)$, thus we can state $W((\Sigma_{j=1}^{J} \alpha_j E_j)_{t+1}) \geq W(\Sigma_{j=1}^{J} \alpha_j E_j)_t)$

A.2 Proof of Theorem 2

This derivation refers to (Masum et al., 2025). Let G and G' denote the local expert gating function parameterized by $(\{K_n^o\}, \{K_n^f\})$ and $\{K_n^f\}$ respectively therefore, G is utilize both K_n^o, and K_n^f,as shown in equation 17 while F' utilize K_n^f only, as shown in equation 18.

$$G(o_{i,t}) = \hat{n} = \underset{n\in[1..N]}{\arg\max}\{cos(o_{i,t}, K_n^o)cos(f_{i,t}, K_n^f)\} \tag{17}$$

$$G'(o_{i,t}) = \hat{n} = \underset{n\in[1..N],}{\arg\max}\{(f_{i,t}.K_n^f)/(max(||f_{i,t}||_2.||K_n^f||_2, \epsilon)\} = \underset{n\in[1..N],}{\arg\max}\{cos(f_{i,t}, K_n^f)\} \tag{18}$$

Let $cos(.)$ denote cosine similarity function, and $L(.)$ denotes head routing error, i.e., the difference between the intended and selected task representation. Therefore, for any $oi, t \in \mathcal{O}$ then we have

$$L(G(oi,t)) = cos(o_{i,t}, K_n^o)cos(f_{i,t}, K_n^f) - \frac{1}{(N-1)N} \sum_{n'\neq n} \sum_{n\in[1..N]} cos(o_{i,t}, K_{n'}^o)cos(f_{i,t}, K_{n'}^f) - \frac{1}{N} \sum_{n\in[1..N]} cos(o_{i,t}, K_n^o)cos(f_{i,t}, K_{n'}^f) \tag{19}$$

$$L(G'(o_{i,t})) = cos(f_{i,t}, K_n^f) - \frac{1}{N-1} \sum_{n'\neq n} cos(f_{i,t}, K_{n'}^f) \tag{20}$$

Thus minimizing $L(G(.))$ and $LG'(.)$ is the same as maximizing G and G' respectively, as presented in the previous equations. Suppose that K_n^f and K_n^o are trained well enough so that for any $oi, t \in \mathcal{O}$, it satisfy $cos(f_{i,t}, K_n^f) \approx C_1$, and $cos(f_{i,t}, K_{n'}^f) \approx C_a = C_1 - \epsilon_1$, $cos(o_{i,t}, K_n^o) \approx S_2$ and

$cos(o_{i,t}, K^o_{n' \neq n}) \approx cos(o, K^o_{n'}) \approx C_b = C_2 - \epsilon_2$, where C_1, C_2, C_a, C_b, ϵ_1, and ϵ_2 have positive values. Then equations 19 can be derived into

$$L(F(.)) = C_1 C_2 - \frac{1}{(N-1)|N|} \sum_{n' \neq n} \sum_{n \in [1..N]} C_a C_b \tag{21}$$

$$L(G(o_{i,t})) = C_1 C_2 - C_a C_b - C_1 C_b \tag{22}$$

and 20 can be derived into

$$L(G'(o_{i,t})) = C_1 - \frac{1}{N-1} \sum_{n' \neq n} C_a \tag{23}$$

$$L(G'(o_{i,t})) = C_1 - C_a \tag{24}$$

Now, assuming that $C_1 \approx C_2 \approx C$ and $\epsilon_1 \approx \epsilon_2 \approx \epsilon$, then we have

$$L(G(o_{i,t})) = C^2 - (C-\epsilon)(C-\epsilon) - C(C-\epsilon) \tag{25}$$

$$L(G(o_{i,t})) = C^2 - (C^2 - 2C\epsilon + \epsilon^2) - C(C-\epsilon) \tag{26}$$

$$L(G(o_{i,t})) = C^2 - C^2 + 2C\epsilon - \epsilon^2 - C^2 + C\epsilon \tag{27}$$

$$L(G(o_{i,2})) = 3C\epsilon - C^2 - \epsilon^2 \tag{28}$$

and we have,

$$L(G'(o_{i,t})) = C_1 - Ca = C - (C-\epsilon) = \epsilon \tag{29}$$

Suppose that we want to find a condition where $L(G(o_{i,t})) \leq L(G'(o_{i,t}))$, then we have

$$3C\epsilon - C^2 - \epsilon^2 \leq \epsilon \tag{30}$$

Assumming that $\epsilon < C$. Then we can choose $\epsilon = 1/3C$, Thus the equation above becomes

$$3C(1/3)C - C^2 - \epsilon^2 \leq \epsilon \tag{31}$$

$$-\epsilon^2 \leq \epsilon \tag{32}$$

that always true for any $\epsilon > 0$. Note that $\epsilon = (1/3)C$ means $C_a = C - \epsilon = C - (1/3)C = (2/3)C$. Choosing $\epsilon < (1/3)C$ also satisfies the inequation above. The derivation above proves Theorem 2:i.e. let G and G' denote the local expert gating functions parameterized by $(\{K^f_n\}, \{K^o_n\})$ and $\{K^f_n\}$ respectively, and $cos(,)$ denote cosine similarity function. If K^f_n and K^o_n are trained well enough so that $cos(o_{i,t}, K^o_n) \approx cos(f_{i,t}, K^f_n) \geq C$, and $cos(o_{i,t}, K^o_{n'}) \approx cos(f_{i,t}, K^f_{n'}) \leq 2/3c$ for any $o_{i,t} \in \mathcal{O}$ then it guarantees $L(G(.)) \leq L(G))$ that means the local expert router $G_2(.)$ will perform better with two reference instead of one reference vector

B Pseudo-Code of GASPOL Algorithm

In this section, we present the procedures of GASPOL training and inference that are shown in algorithms 1 and 2, respectively.

C Extended Literature Review

Several cooperative Multi-Agent Reinforcement Learning (MARL) algorithms based on policy-gradient methods have been proposed and evaluated across various multi-agent environments, including Multi-Agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al., 2017b) and Multi-Agent Proximal Policy Optimization (MAPPO) (Yu et al., 2022). These methods commonly adopt the Centralized Training with Decentralized Execution (CTDE) paradigm, where joint observations and actions are available only during training phase to construct a centralized critic. By incorporating global information, the critic can capture inter-agent dependencies, facilitate coordination, and mitigate the non-stationarity caused by simultaneous policy updates. During the execution

Algorithm 1 GASPOL Training

Input: A sequence of tasks $\mathcal{T}^1$, $\mathcal{T}^2$,...,$\mathcal{T}^N$, a policy model π_θ, stepsize α, batchsize B, number of agent $I = |\mathcal{I}|$, episodes K, step per episode T
Initialize: Actor networks $\{\theta_{i,0}, \forall i \in \mathcal{I}\}$, global V-value (critic) network V^{ctr}, $G_{1,i}$, and buffer $\mathcal{B}$, gating network $G_{2,i}(.)$ is included in $\theta_i(.)$
for $n = 1 : N$ **do**
  Initiate $K^o_{i,n}$, $K^f_{i,n}$ and $K^p_{i,n}$
  If $(n > 1)$ add new head g_n into θ_i
  for $k = 0, 1, ..., K-1$ **do** do
    Collect a set of trajectories by running the joint policy $\pi_{\theta_k} = (\pi_{\theta_{1,k}}, \pi_{\theta_{2,k}}, ..., \pi_{\theta_{I,k}})$
    Push transitions $\{(s_t, o_{i,t}, a_{i,t}, r_t, s_{t+1}, o_{i,t+1}\}$
    Generate $f_{i,t}$, and $a_{i,t+1}$ by executing Eq 1 and 2
    Compute advantage function $\tilde{A}(s, a)$ based on V-value network V^{ctr}
    Draw a random permutation of agents $i_{1:I}$
    Set $M_{i1}(s, a) = \tilde{A}(s, a)$
    for a **do**gent $i_m = i_1, ..., i_I$ do
      Compute $\mathcal{L}^{pol}_i$ as in eq. 3
      Compute $\mathcal{L}^{K^o}_i$ and $\mathcal{L}^{K^f}_i$ as in eq. 5
      Compute $\mathcal{L}^{K^p}_i$ as in eq. 6
      Compute $\mathcal{L}^{reg}_i$ as in eq. 7
      Compute $\mathcal{L}^{pen}_i$ if $(n' \neq n)$
      Compute $\mathcal{L}^{total}_i$ as in eq. 8
      Update parameters $\theta^t = \{\{G^t_i\}^k_{i=1}, K^t, \{L^t_c\}^{|C^t|}_{c=1}\}$ based on $\mathcal{L}_{total}$
      Update actor i_m with $\theta_{i_m,k+1}$, following PPO-CLIP objective
      Compute $M_{i_{1:m+1}}(s, a)$ following HAPPO advantage order as in eq. 4
    end for
    Update critic network V^{ctr} by $agrmin \frac{1}{BT} \Sigma^B_{b=1} \Sigma^T_{t=0} (V(s_t) - \tilde{R}_t)^2$
  end for
end for
Output: Optimum parameters $\theta_i, i \in [1..I]$

Algorithm 2 GASPOL Inference

Input: A sequence of tasks $\mathcal{T}^1$, $\mathcal{T}^2$,...,$\mathcal{T}^N$, a policy model π_θ, T number of evaluation step on each task
for $n = 1 : N$ **do**
  for $t = 1 : T$ **do**
    for a **do**gent $i_m = i_1, ..., i_I$ do
      Get observation from environment $o_{i,t}$
      Generate $f_{i,t}$, and $a_{i,t+1}$ by executing Eq 1 and 2
    end for
  end for
end for

phase, each agent independently selects actions using its local observation and decentralized actor, without requiring global information.

However, conventional MARL methods generally assume homogeneous agents with similar capabilities and policy structures. To address heterogeneous multi-agent settings, methods such as Heterogeneous-Agent Trust Region Policy Optimization (HATRPO) (Kuba et al., 2022), Heterogeneous-Agent Proximal Policy Optimization (HAPPO) (Zhong et al., 2024), and Heterogeneous-Agent Soft Actor-Critic (HASAC) (Liu et al., 2024) have been developed. These approaches allow agents to maintain individual policies that can be specialized according to their distinct roles, capabilities, and observation or action spaces, while centralized critics can still be employed during training to facilitate global coordination. In particular, HATRPO and HAPPO optimize agent policies sequentially rather than simultaneously, allowing each update to account for the evolving policies of other agents and thereby improving training stability. HATRPO further constrains policy updates using a trust-region objective, while HAPPO employs a clipped surrogate objective to limit excessively large policy changes. These characteristics make heterogeneous-agent policy optimization suitable for multi-agent resource allocation problems involving agents with different capabilities and operational roles.

Despite these advances, the aforementioned methods are predominantly evaluated in single-task environments, where agents are optimized under a fixed task distribution and typically remain within the same task setting. An universal transformer based policy (Hu et al., 2021) has been proposed to handle transferability between tasks in SMAC. However, the evaluation performed only in target or new task. This assumption limits their ability to address scenarios in which agents must adapt to a continual sequence of changing tasks without loosing the capability in the previous tasks. Therefore, further investigation has been extended into continual MARL, where agents incrementally learn and adapt to multiple tasks over time while retaining previously acquired knowledge. Such a setting introduces additional challenges, including catastrophic forgetting, knowledge transfer, and adaptation to task changes.

COntinual RL Without Conflict (OWL) (Kessler et al., 2022) applied a regularization method to a Q-MIX (Rashid et al., 2018) policy and utilized a multi-armed bandit to select the best head for the current task. Similarly, Multi-agent Continual Coordination via Progressive Task Contextualization (MACPRO) utilized an attention-based architecture, regularization, and selected the best classification head based on a task contextualization mechanism. Both OWL and MACPRO require trajectory data from previous tasks to train the actor model. In addition, both methods gather many observations from the environment to adjust their head selector during the evaluation phase. Such a mechanism may not be suitable for a real-world environment where the agents need to take actions based on every single observation. Another study was focused on developing a framework and benchmarking for continual MARL environments (Tomilin et al., 2026). The mentioned studies reported that native continual learning methods such as EWC Kirkpatrick et al. (2017) and MAS (Aljundi et al., 2018) can not fully handle catastrophic policy forgetting for the multi-agent systems. The concept in rehearsal-based methods such as knowledge distillation and experience replay, as in ICARL (Rebuffi et al., 2017), GDUMB (Prabhu et al., 2020), and XDER (Boschini et al., 2022) is relevant to rehearsal-based methods such as MACPRO. In addition, the growing sub-network mechanism as in BiC (Wu et al., 2019) and LUCIR (Hou et al., 2019) is presented in OWL and MACPRO, to learn a new policy from the currently learned task.

D Extended Experimental Settings:

D.1 Performance Metrics:

After finishing a task $\mathcal{T}^n$, we measure the average success rate (SR), which covers all learned tasks ($\mathcal{T}^1, \mathcal{T}^2, ..\mathcal{T}^n$), final average success rate (FSR) that is the same as SR on the last task (task N), final forgetting measure (FFM) that represent the final performance after learning all tasks, and cumulative success rate rate (CSR) that represent cumulative historical performance on each task. Success rate is measured as winning rate in SMAC games, and normalized rewards for Pettingzoo MPE2 and MAMuJoCo. Final average success rate (FSR), cumulative average rate(CSR), and final forgetting measure (FFM) defined in the equations below.

$$FSR = SR_N \tag{33}$$

```
StochasticPolicyGASPOL(
  (base): MLPBase(
    (feature_norm): LayerNorm((384,), eps=1e-05, elementwise_affine=True)
    (mlp): MLPLayer(
      (fc): Sequential(
        (0): Linear(in_features=384, out_features=128, bias=True)
        (1): ReLU()
        (2): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
        (3): Linear(in_features=128, out_features=128, bias=True)
        (4): ReLU()
        (5): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
      )
    )
  )
  (s_experts): ModuleList(
    (0-3): 4 x MLPLayer(
      (fc): Sequential(
        (0): Linear(in_features=128, out_features=128, bias=True)
        (1): ReLU()
        (2): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
      )
    )
  )
  (rnn): RNNLayer(
    (rnn): GRU(128, 128)
    (norm): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
  )
  (act): ModuleList(
    (0-3): 4 x ACTLayer(
      (action_out): DiagGaussian(
        (fc_mean): Linear(in_features=128, out_features=16, bias=True)
      )
    )
  )
  (gating_network): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=384, out_features=4, bias=True)
    (2): Softmax(dim=1)
  )
  (obs_key): ParameterList(
      (0): Parameter containing: [torch.float32 of size 384]
      (1): Parameter containing: [torch.float32 of size 384]
      (2): Parameter containing: [torch.float32 of size 384]
      (3): Parameter containing: [torch.float32 of size 384]
  )
  (feat_key): ParameterList(
      (0): Parameter containing: [torch.float32 of size 128]
      (1): Parameter containing: [torch.float32 of size 128]
      (2): Parameter containing: [torch.float32 of size 128]
      (3): Parameter containing: [torch.float32 of size 128]
  )
  (rfeat_key): ParameterList(
      (0): Parameter containing: [torch.float32 of size 384]
      (1): Parameter containing: [torch.float32 of size 384]
      (2): Parameter containing: [torch.float32 of size 384]
      (3): Parameter containing: [torch.float32 of size 384]
  )
)
```

Figure 6: The structure of Our Policy Model.

$$CSR = \frac{1}{N}\Sigma_{n=1}^{N} SR_n \tag{34}$$

$$FFM = \frac{1}{N-1}\Sigma_{i=1}^{N-1} max_{t\in 1,\ldots,T-1}(SR_{i,n} - SR_{i,N}) \tag{35}$$

D.2 GASPOL Policy Model

Please see figure 6 for the detail architecture of our policy network.

Table 5: Hyperparameter Setting

Category	Parameter	Value	Description
device	cuda	False	Whether to enable CUDA (GPU acceleration)
	cuda_deterministic	True	Enforce deterministic CUDA operations
	torch_threads	4	Number of threads assigned to PyTorch
train	n_rollout_threads	10	Parallel environments for data collection
	num_env_steps	500000	Total environment execution steps
	episode_length	200	Steps per environment per trajectory
	log_interval	5	Logging frequency in episodes
	eval_interval	5	Evaluation frequency in episodes
	use_valuenorm	True	Enable value normalization
	use_linear_lr_decay	False	Enable linear decay for learning rates
	use_proper_time_limits	True	Account for truncation at episode boundaries
	model_dir	$\sim$(null)	Pre-trained model directory path
eval	use_eval	True	Enable evaluation phases
	n_eval_rollout_threads	2	Parallel evaluation environments
	eval_episodes	20	Episodes executed per evaluation run
render	use_render	False	Enable environment rendering
	render_episodes	10	Episodes to render when enabled
model	hidden_sizes	[128, 128]	Dimensions of MLP hidden layers
	activation_func	relu	Activation function (sigmoid, tanh, relu, leaky_relu, selu)
	use_feature_normalization	True	Standardize observation inputs
	initialization_method	orthogonal_	Network parameter initialization strategy
	gain	0.01	Gain factor for network output layer
	use_naive_recurrent_policy	False	Unchunked RNN policy execution
	use_recurrent_policy	True	Chunked RNN policy execution
	recurrent_n	1	Number of recurrent layers
	data_chunk_length	10	Sequence chunk length for BPTT
	lr	0.0005	Learning rate for the actor network
	critic_lr	0.0005	Learning rate for the critic network
	opti_eps	1E-05	Epsilon hyperparameter for Adam optimizer
	weight_decay	0	Weight decay penalty for Adam optimizer
	std_x_coef	1	Diagonal Gaussian distribution X parameter
	std_y_coef	0.5	Diagonal Gaussian distribution Y parameter
	max_obs_length	384	Maximum allowable observation length
	max_act_length	16	Maximum allowable action length
	key_length	1	Key length parameter
algo	ppo_epoch	5	Epochs per actor optimization pass
	critic_epoch	5	Epochs per critic optimization pass
	use_clipped_value_loss	True	Enable value loss clipping
	clip_param	0.2	PPO surrogate clipping threshold (ϵ)
	actor_num_mini_batch	1	Mini-batches per actor update epoch
	critic_num_mini_batch	1	Mini-batches per critic update epoch
	entropy_coef	0.01	Weight of the entropy loss component
	value_loss_coef	1	Weight of the value loss component
	use_max_grad_norm	True	Enable gradient norm clipping
	max_grad_norm	10	Maximum gradient norm threshold
	use_gae	True	Use Generalized Advantage Estimation
	gamma	0.99	Discount factor (γ)
	gae_lambda	0.95	GAE trade-off parameter (λ)
	use_huber_loss	True	Use Huber loss for critic objective
	use_policy_active_masks	True	Apply active masking to policies
	huber_delta	10	Huber loss threshold (δ)
	action_aggregation	prod	Multi-dimensional action aggregation (prod or mean)
	share_param	False	Share policy parameters across actors
	fixed_order	False	Enforce fixed update order across actors
	obs_sim_loss	1	Auxiliary loss coefficient for observation similarity
	feat_sim_loss	1	Auxiliary loss coefficient for feature similarity

D.3 Hyper Parameters:

Please see Table 5 for the details of hyperparameter setting.

Table 6: Raw Numerical Result on MAMuJoCo problem

Evaluated Task	Task where To do eval	ORACLE-Full AVG	ORACLE-Full STDEV	ORACLE AVG	ORACLE STDEV	FT STDEV	FT AVG
0	0	-27.60	2.02	-27.60	2.02	-27.61	2.02
0	1	-35.92	0.76	-36.20	3.03	-33.83	1.64
1	1	-14.04	1.70	-16.22	3.45	-17.52	2.44
0	2	-31.25	0.96	-33.82	1.75	-35.81	2.56
1	2	-22.42	2.46	-28.72	4.19	-31.43	1.38
2	2	-29.40	1.14	-30.89	2.42	-37.51	0.31
0	3	-34.83	0.47	-44.03	2.46	-40.68	6.67
1	3	-19.86	3.02	-26.16	1.97	-32.39	4.03
2	3	-25.18	1.29	-37.88	3.92	-40.98	1.13
3	3	-30.08	1.20	-25.40	1.74	-40.39	1.41
Evaluated Task	**Task where To do eval**	**EWC AVG**	**EWC STDEV**	**SI AVG**	**SI STDEV**	**MAS STDEV**	**MAS AVG**
0	0	-27.60	2.02	-26.64	2.25	-27.60	2.02
0	1	-35.96	2.33	-34.82	2.28	-35.98	2.47
1	1	-17.70	2.47	-19.17	0.69	-18.83	2.10
0	2	-32.33	2.37	-32.32	1.76	-34.44	4.46
1	2	-30.57	3.90	-29.59	5.84	-32.70	3.91
2	2	-35.55	1.28	-38.38	0.98	-37.97	2.13
0	3	-40.98	2.58	-36.60	3.10	-38.62	4.46
1	3	-43.87	6.34	-32.32	7.01	-28.53	3.96
2	3	-37.33	1.89	-40.63	1.80	-39.70	1.24
3	3	-39.31	1.40	-40.37	2.99	-40.66	2.06
Evaluated Task	**Task where To do eval**	**OWL AVG**	**OWL STDEV**	**GASPOL AVG**	**GASPOL STDEV**		
0	0	-27.60	2.02	-27.82	0.91		
0	1	-35.96	2.33	-36.65	2.56		
1	1	-17.70	2.47	-17.21	2.66		
0	2	-32.33	2.37	-33.36	1.18		
1	2	-30.57	3.90	-24.75	1.07		
2	2	-35.55	1.28	-34.54	2.91		
0	3	-40.98	2.58	-34.12	1.00		
1	3	-43.87	6.34	-20.94	1.35		
2	3	-37.33	1.89	-35.14	1.09		
3	3	-39.31	1.40	-36.34	1.01		

E RAW NUMERICAL RESULT

This section presents raw numerical results on MPE2 and MAMuJoCo problems, as hown by Tables 6 and 7.

Table 7: Raw Numerical Result on MAMuJoCo problem

Evaluated Task	Task where To do eval	ORACLE-Full AVG	ORACLE-Full STDEV	ORACLE AVG	ORACLE STDEV	FT STDEV	FT AVG
0	0	38.70	4.68	38.70	4.68	16.02	19.64
0	1	38.37	3.90	23.42	2.57	9.96	11.67
1	1	4538.01	0.00	2421.64	1094.96	1172.20	1212.67
0	2	38.19	4.60	22.10	3.45	10.05	10.45
1	2	4518.81	0.00	996.93	574.95	523.96	500.42
2	2	935.97	104.57	737.53	111.10	317.73	363.57
0	3	38.87	3.99	7.24	20.85	10.69	8.94
1	3	4461.20	0.00	596.88	633.90	410.26	355.78
2	3	936.59	101.98	789.91	112.48	334.79	394.18
3	3	894.88	24.61	968.51	115.12	369.42	520.80
Evaluated Task	**Task where To do eval**	**EWC AVG**	**EWC STDEV**	**SI AVG**	**SI STDEV**	**MAS STDEV**	**MAS AVG**
0	0	38.70	4.68	38.70	4.68	41.82	2.89
0	1	30.54	7.51	-26.12	25.79	15.72	1.22
1	1	2091.58	241.71	2897.97	1367.94	3059.07	1031.52
0	2	23.62	0.84	22.74	20.28	11.26	6.14
1	2	56.38	130.81	-56.78	25.33	-19.50	64.23
2	2	922.44	44.40	907.48	22.41	910.49	46.81
0	3	-3.00	27.14	-19.42	6.36	-6.96	12.40
1	3	71.02	248.56	-53.17	42.14	-16.13	39.87
2	3	855.14	38.90	907.60	28.18	868.50	17.10
3	3	777.77	53.52	974.37	94.07	1341.36	416.78
Evaluated Task	**Task where To do eval**	**OWL AVG**	**OWL STDEV**	**GASPOL AVG**	**GASPOL STDEV**		
0	0	36.70	0.33	34.12	3.42		
0	1	35.12	3.58	30.51	5.48		
1	1	1540.14	848.66	4210.44	180.55		
0	2	36.76	2.37	35.75	3.11		
1	2	1302.29	835.46	3493.61	920.79		
2	2	962.72	223.64	904.57	52.09		
0	3	34.14	2.27	29.86	10.02		
1	3	826.34	656.55	2594.32	588.83		
2	3	910.02	134.87	859.38	82.56		
3	3	853.92	156.27	855.38	155.11		