Kaleidoscope: Learnable Masks for Heterogeneous Multi-agent Reinforcement Learning

Xinran Li **Ling Pan** **Jun Zhang**

Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
xinran.li@connect.ust.hk, lingpan@ust.hk, eejzhang@ust.hk

Abstract

In multi-agent reinforcement learning (MARL), parameter sharing is commonly employed to enhance sample efficiency. However, the popular approach of full parameter sharing often leads to homogeneous policies among agents, potentially limiting the performance benefits that could be derived from policy diversity. To address this critical limitation, we introduce *Kaleidoscope*, a novel adaptive partial parameter sharing scheme that fosters policy heterogeneity while still maintaining high sample efficiency. Specifically, Kaleidoscope maintains one set of common parameters alongside multiple sets of distinct, learnable masks for different agents, dictating the sharing of parameters. It promotes diversity among policy networks by encouraging discrepancy among these masks, without sacrificing the efficiencies of parameter sharing. This design allows Kaleidoscope to dynamically balance high sample efficiency with a broad policy representational capacity, effectively bridging the gap between full parameter sharing and non-parameter sharing across various environments. We further extend Kaleidoscope to critic ensembles in the context of actor-critic algorithms, which could help improve value estimations. Our empirical evaluations across extensive environments, including multi-agent particle environment, multi-agent MuJoCo and StarCraft multi-agent challenge v2, demonstrate the superior performance of Kaleidoscope compared with existing parameter sharing approaches, showcasing its potential for performance enhancement in MARL. The code is publicly available at `https://github.com/LXXXXR/Kaleidoscope`.

1 Introduction

Cooperative multi-agent reinforcement learning (MARL) has demonstrated remarkable effectiveness in solving complex real-world decision-making problems across various domains, such as resource allocation (Ying and Dayong, 2005), package delivery (Seuken and Zilberstein, 2007), autonomous driving (Zhou et al., 2021), and robot control (Swamy et al., 2020). To mitigate the challenges posed by the non-stationary and partially observable environments typical of MARL (Yuan et al., 2023), the centralized training with decentralized execution (CTDE) paradigm (Foerster et al., 2016) has become prevalent, inspiring many influential MARL algorithms such as MADDPG (Lowe et al., 2017), COMA (Foerster et al., 2018), MATD3 (Ackermann et al., 2019), QMIX (Rashid et al., 2020), and MAPPO (Yu et al., 2022).

Under the CTDE paradigm, parameter sharing among agents is a commonly adopted practice to improve sample efficiency. However, identical network parameters across agents often lead to homogeneous policies, restricting diversity in behaviors and the overall joint policy representational capacity. This limitation can result in undesired outcomes in certain situations (Christianos et al., 2021; Fu et al., 2022; Kim and Sung, 2023), as shown in Figure 1, impeding further performance

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-0695

gains. An alternative approach is the non-parameter sharing scheme, where each agent possesses its own unique parameters. Nevertheless, while this method naturally supports heterogeneous policies, it suffers from reduced sample efficiency, leading to significant training costs. This is particularly problematic given the current trend towards increasingly large model sizes, with some scaling to trillions of parameters (Zhao et al., 2023; Achiam et al., 2023). Therefore, it is imperative to develop a parameter sharing strategy that enjoys both high sample efficiency and broad policy representational capacity, potentially achieving significantly enhanced performance. While several efforts (Christianos et al., 2021; Kim and Sung, 2023) have explored partial parameter sharing initiated at the start of training, such initializations can be challenging to design without detailed knowledge of agent-specific environmental transitions or reward functions (Christianos et al., 2021).

In this work, we build upon insights from previous studies (Christianos et al., 2021; Fu et al., 2022; Kim and Sung, 2023) and introduce *Kaleidoscope*, a novel adaptive partial parameter sharing scheme. It maintains a single set of policy parameters and employs multiple learnable masks to designate the shared parameters. Unlike earlier methods that depend on fixed initializations, Kaleidoscope dynamically learns these masks alongside MARL parameters throughout the training process. This end-to-end training approach inherently integrates environmental information, and its adaptive nature enables Kaleidoscope to dynamically adjust the level of parameter sharing based on the demands of the environment and the learning progress of the agents. The learnable masks facilitate a dynamic balance between full parameter sharing and non-parameter sharing, offering a flexible trade-off between sample efficiency and policy representational capacity through enhanced heterogeneity. Initially, we build Kaleidoscope upon agent networks, where it achieves diverse policies. Following this success, we extend it to multi-agent actor-critic algorithms to encourage heterogeneity among the central critic ensembles for further performance enhancement.

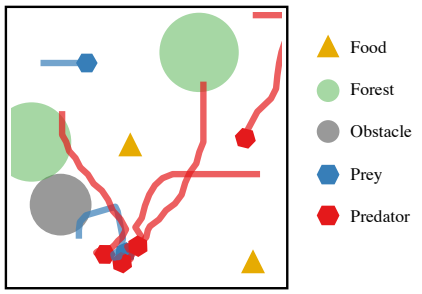


Figure 1: Full parameter sharing confines the policies to be homogeneous. In this example, all predators pursue the same prey, neglecting another prey in the game `World`. Further game details are in Appendix A.2.

> Just like a *kaleidoscope* uses the reflective properties of rotating mirrors to transform simple shapes into beautiful patterns, our proposed method leverages learnable masks to map a single set of parameters into diverse policies, thereby enhancing task performance.

We summarize our contributions as follows:

- To enable policy heterogeneity among agents for better training flexibility, we adapt the soft threshold reparameterization (STR) technique to learn distinct masks for different agent networks while only maintaining one set of common parameters, effectively balancing between full parameter sharing and non-parameter sharing mechanisms.
- To enhance policy diversity among agents, we introduce a novel regularization term that encourages the pairwise discrepancy between masks. Additionally, we design resetting mechanisms that recycle masked parameters to preserve the representational capacity of the joint networks.
- Through extensive experiments on MARL benchmarks, including multi-agent particle environment (MPE) (Lowe et al., 2017), multi-agent MuJoCo (MAMuJoCo) (Peng et al., 2021) and StarCraft multi-agent challenge v2 (SMACv2) (Ellis et al., 2024), we demonstrate the superior performance of Kaleidoscope over existing parameter sharing approaches.

2 Background

Multi-agent reinforcement learning (MARL) In MARL, a fully cooperative partially observable multi-agent task is typically formulated as a decentralized partially observable Markov decision process (dec-POMDP) (Oliehoek and Amato, 2016), represented by a tuple $\mathcal{M} = \langle \mathcal{S}, A, P, R, \Omega, O, N, \gamma \rangle$. Here, N denotes the number of agents, and $\gamma \in (0, 1]$ represents the discount factor. At each timestep t, with the environment state as $s^t \in \mathcal{S}$, agent i receives a local observation $o_i^t \in \Omega$ drawn from the observation function $O(s^t, i)$ and then follows its local policy

π_i to select an action $a_i^t \in A$. Individual actions form a joint action $\boldsymbol{a}^t \in A^N$, leading to a state transition to the next state $s^{t+1} \sim P(s^{t+1}|s^t, \boldsymbol{a}^t)$ and inducing a global reward $r^t = R(s^t, \boldsymbol{a}^t)$. The overall team objective is to learn the joint policies $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_N \rangle$ such that the expectation of discounted accumulated reward $G^t = \sum_t \gamma^t r^t$ is maximized.

To learn such policies $\boldsymbol{\pi_\theta}$, various MARL algorithms (Lowe et al., 2017; Foerster et al., 2018; Rashid et al., 2020; Yu et al., 2022) have been developed. For instance, the off-policy actor-critic algorithm MATD3 (Ackermann et al., 2019) serves as an example method. Specifically, the critic networks are updated by minimizing the temporal difference (TD) error loss

$$\mathcal{L}_c(\phi) = \mathbb{E}_{(s^t, \boldsymbol{o}^t, \boldsymbol{a}^t, r^t, s^{t+1}, \boldsymbol{o}^{t+1}) \sim \mathcal{D}} \left[\left(y^t - Q(s^t, \boldsymbol{a}^t; \phi) \right)^2 \right], \tag{1}$$

with

$$y^t = r^t + \gamma \min_{j=1,2} Q(s^{t+1}, \pi_1(o_1^{t+1}; \theta_1') + \epsilon, \ldots, \pi_N(o_N^{t+1}; \theta_N') + \epsilon; \phi_j), \tag{2}$$

where ϕ are the parameters for critics, θ are the parameters for actor policies and θ' are the parameters for target actor policies. And ϵ is the clipped Gaussian noise, given as $\text{clip}(\mathcal{N}(0, \sigma), -c, c)$.

The policy is updated by the deterministic policy gradient algorithm (Silver et al., 2014)

$$\nabla \mathcal{J}(\theta_i) = \mathbb{E}_{(s^t, \boldsymbol{o}^t, \boldsymbol{a}^t, r^t, s^{t+1}, \boldsymbol{o}^{t+1}) \sim \mathcal{D}} \left[\nabla_{\theta_i} \pi_i(o_i^t; \theta_i) \nabla_{a_i} Q(s^t, a_1, \ldots, a_N|_{a_i = \pi_i(o_i^t; \theta_i)}; \phi_1) \right]. \tag{3}$$

Soft threshold reparameterization (STR) Originally introduced in the context of model sparsification, STR (Kusupati et al., 2020) is an unstructured pruning method that achieves notable performance without requiring a predetermined sparsity level. Specifically, STR applies a transformation to the original parameters W as follows

$$\mathcal{S}_g(W, s) = \text{sign}(W) \cdot \text{ReLU}\left(|W| - g(s) \right), \tag{4}$$

where s is a learnable parameter, $\alpha = g(s)$ serves as the pruning threshold, and $\text{ReLU}(\cdot) = \max(\cdot, 0)$. The original supervised learning problem modeled by

$$\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}; \mathcal{D}) \tag{5}$$

with $\mathcal{D}$ as the data is now transferred to

$$\min_{\boldsymbol{W}, \boldsymbol{s}} \mathcal{L}(\mathcal{S}_g(\boldsymbol{W}, \boldsymbol{s}); \mathcal{D}). \tag{6}$$

Overall, this approach optimizes the learnable pruning threshold alongside the model parameters, facilitating dynamic adjustment to the sparsity level during training.

3 Learnable Masks for Heterogenous MARL

In this section, we propose using learnable masks as a low-cost method to enable network heterogeneity in MARL. The core concept, illustrated in Figure 2, is to learn a single set of shared parameters complemented by multiple masks for distinct agents, specifying which parameters to share.

Specifically, in Section 3.1, we first adapt STR into a dynamic partially parameter sharing method, unlocking the joint policy network's capability to represent diverse policies among agents. In Section 3.2, we actively foster policy heterogeneity through a novel regularization term based on the masks. Given that the masking technique could excessively sparsify the network, potentially diminishing its representational capacity, in Section 3.3, we propose a straightforward remedy to periodically reset the parameters based on the outcomes of masking, which additionally mitigates primacy bias. Finally, in Section 3.4, we explore how to further extend this approach within the critic components of actor-critic algorithms to improve value estimations in MARL and further boost performance.

For the sake of clarity, we integrate the proposed Kaleidoscope with the MATD3 (Ackermann et al., 2019) algorithm to demonstrate the concept within this section. Nevertheless, as a versatile partial parameter-sharing technique, our method can readily be adapted to other MARL algorithms. We defer its integration with other MARL frameworks to Appendix A.1.2 and will evaluate them empirically in Section 4.

https://doi.org/10.52202/079017-0695

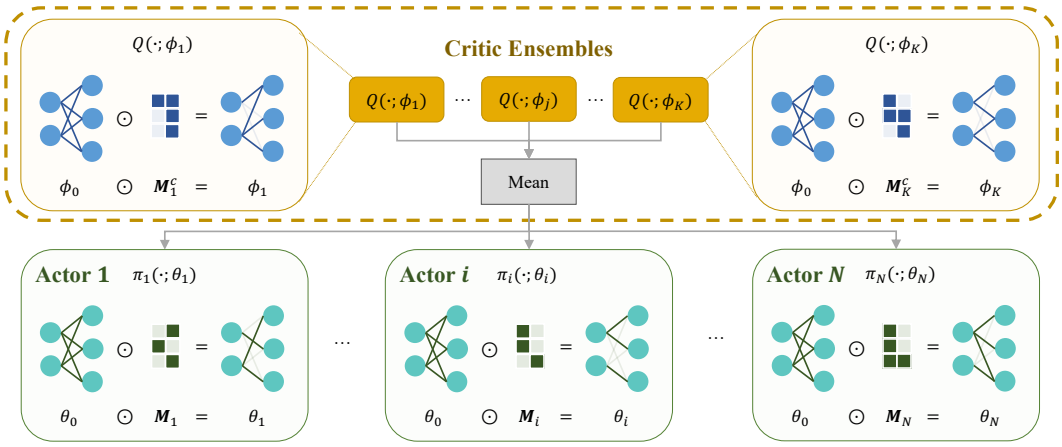


Figure 2: Overall network architecture of Kaleidoscope. It maintains one set of parameters θ_0 with N sets of masks $[\boldsymbol{M}_i]_{i=1}^N$ for actor networks, and one set of parameters ϕ_0 with K sets of masks $\left[\boldsymbol{M}_j^c\right]_{j=1}^K$ for critic ensemble networks, where N is the number of agents, K is the number of ensembles, and $\odot$ denotes the Hadamard product.

3.1 Adaptive partial parameter sharing ♦

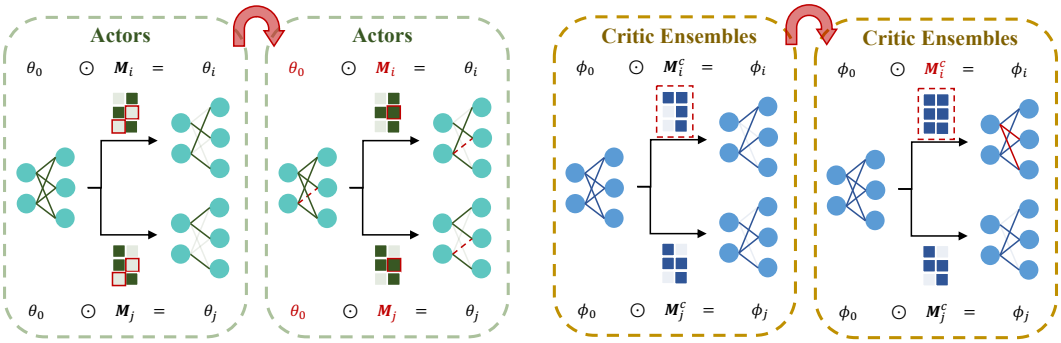
The core idea of this work is to learn distinct binary masks $\boldsymbol{M}_i$ for different agents to facilitate differentiated policies, ultimately aiming to improve MARL performance. To achieve this, we apply the STR (Kusupati et al., 2020) technique to the policy parameters with different thresholds dedicated to each agent:

$$\theta_i = \theta_0 \odot \boldsymbol{M}_i, \tag{7}$$

where θ_i parameterizes the policy for agent i, θ_0 is the set of learnable parameters shared by all agents, and $\boldsymbol{M}_i$ is the learnable mask for agent i. Specifically, assume $\theta_0 = \left[\theta_0^{(1)}, \ldots, \theta_0^{(N_a)}\right]$, $\theta_i = \left[\theta_i^{(1)}, \ldots, \theta_i^{(N_a)}\right]$ and $\boldsymbol{M}_i = \left[m_i^{(1)}, \ldots, m_i^{(N_a)}\right]$ with N_a being the total parameter count of an agent's network. In line with STR, we compute each element $m_i^{(k)}$ of $\boldsymbol{M}_i$ as $m_i^{(k)} = \mathbb{1}\left[|\theta_0^{(k)}| > \sigma(s_i^{(k)})\right]$, where $\sigma(\cdot)$ denotes the Sigmoid function.

The benefits of such a combination are summarized as follows:

- **Preservation of original MARL learning objectives:** Unlike most of the methods in pruning literature, which primarily aim to minimize the discrepancies between pruned and unpruned networks in terms of weights, loss, or activations (Hoefler et al., 2021; Menghani, 2023; Deng et al., 2020), STR maintains the original goal of minimizing task-specific loss, aligning directly with our objectives to enhance MARL performance.
- **Flexibility in sparsity:** Many classical pruning methods require predefined per-layer sparsity levels (Evci et al., 2020; Ramanujan et al., 2020). Such requirements can complicate our design, with the goal not to gain extreme sparsity but rather to promote heterogeneity through masking. The STR technique is ideal in our case as it does not require predefining sparsity levels, allowing for adaptive learning of the masks.
- **Enhanced network representational capacity:** Utilizing learnable masks for adaptive partial parameter sharing enhances the network's representational capacity beyond traditional full parameter sharing. In full parameter sharing, agents' joint policies are parameterized as $\boldsymbol{\pi}^{\text{ps}}(\cdot|\theta_0) = \langle \pi_1(\cdot|\theta_0), \ldots, \pi_N(\cdot|\theta_0)\rangle$. In contrast, our proposed adaptive partial parameter sharing mechanism parameterizes the joint policies as $\boldsymbol{\pi}^{\text{Kaleidoscope}}(\cdot|\theta_0, \boldsymbol{M}) = \langle \pi_1(\cdot|\theta_0 \odot \boldsymbol{M}_1), \ldots, \pi_n(\cdot|\theta_0 \odot \boldsymbol{M}_N)\rangle$. In the extreme case where all the values in $\boldsymbol{M}_i$ are 1s, the function set represented by $\boldsymbol{\pi}^{\text{Kaleidoscope}}(\cdot|\theta_0, \boldsymbol{M})$ degrades to that of $\boldsymbol{\pi}^{\text{ps}}(\cdot|\theta_0)$. In other scenarios, it is a superset of that represented by $\boldsymbol{\pi}^{\text{ps}}(\cdot|\theta_0)$.



(a) Actors: reinitialize the weights that are masked by all agents with probability ρ.

(b) Critic ensembles: reset one set of masks at a time.

Figure 3: Illustration on resetting mechanisms.

3.2 Policy diversity regularization ♣

While independently learned masks enable agents to develop distinct policies, without a specific incentive, these policies may still converge to being homogeneous. To this end, we propose to explicitly encourage agent policy heterogeneity by introducing a diversity regularization term maximizing the weighted pairwise distance between network masks, which is defined as

$$\mathcal{J}^{\text{div}}(\boldsymbol{s}) = \sum_{i=1,\dots,n} \sum_{\substack{j=1,\dots,n \\ j\neq i}} \|\theta_0 \odot (\boldsymbol{M}_i - \boldsymbol{M}_j)\|_1. \tag{8}$$

This term is inherently non-differentiable due to the indicator function $\mathbb{1}[\cdot]$ inside $\boldsymbol{M}$. To overcome this difficulty, following established practices in the literature (Bengio et al., 2013; Alizadeh et al., 2018), we utilize a surrogate function for gradient approximation:

$$\frac{\partial \mathcal{J}^{\text{div}}}{\partial g(\boldsymbol{s}_i)} = -\tanh\left[\frac{\partial \mathcal{J}^{\text{div}}}{\partial \boldsymbol{M}_i}\right]. \tag{9}$$

We formally provide the overall training objective for actors in Appendix A.1.1.

3.3 Periodically reset ♠

As the training with masks proceeds, we observe an increasing sparsity in each agent's network, potentially reducing the overall network capacity. To remedy the issue, we propose a simple approach to periodically reset the parameters that are consistently masked across all $\boldsymbol{M}_i$ with a certain probability ρ, which is illustrated in Figure 3a. At intervals defined by $t \mod \text{reset\_interval} == 0$, if the parameter index k satisfies $\forall i, m_i^{(k)} == 0$, we apply the following resetting rule

$$\theta_0^{(k)}, s_1^{(k)}, \dots, s_N^{(k)} \leftarrow \begin{cases} \text{Reinitialize}[\theta_0^{(k)}, s_1^{(k)}, \dots, s_N^{(k)}] & \text{with probability } \rho \\ \theta_0^{(k)}, s_1^{(k)}, \dots, s_N^{(k)} & \text{with probability } 1-\rho \end{cases}. \tag{10}$$

This resetting mechanism recycles the weights masked as zeros by all the masks, preventing the networks from becoming overly sparse. A side benefit of this resetting mechanism is the enhancement of neural plasticity (Lyle et al., 2023; Nikishin et al., 2024), which helps alleviate the primacy bias (Nikishin et al., 2022) in reinforcement learning. Unlike methods that reinitialize entire layers resulting in abrupt performance drops (Nikishin et al., 2022), our resetting approach selectively targets weights as indicated by the learnable masks, thus avoiding significant performance disruptions, as shown in Section 4.

3.4 Critic ensembles with learnable masks

In actor-critic algorithm frameworks, we further apply Kaleidoscope to central critics as an efficient way to implement ensemble-like critics. By facilitating dynamic partial parameter sharing, Kaleidoscope enables heterogeneity among critic ensembles. Furthermore, by regularizing the diversity

https://doi.org/10.52202/079017-0695

among critic functions, we can control ensemble variances. This approach is elaborated in subsequent paragraphs.

♦ **Adaptive partial parameter sharing for critic ensembles** In the standard MATD3 algorithm (Ackermann et al., 2019), two critics with independent parameters are maintained to mitigate overestimation risks. However, using separate parameters typically results in a low update-to-data (UTD) ratio (Hiraoka et al., 2022). To address this issue, we propose to enhance the UTD ratio by employing Kaleidoscope parameter sharing among ensembles of critics. Specifically, we maintain a single set of parameters ϕ_0 and K masks $\left[\boldsymbol{M}_j^c\right]_{j=1}^K$ to distinguish the critic functions, resulting in K ensembles $[Q(\cdot;\phi_j)]_{j=1}^K$ with $\phi_j = \phi_0 \odot \boldsymbol{M}_j^c$.

To be specific, we update the critic networks by minimizing the temporal difference (TD) error loss

$$\mathcal{L}_c(\phi_j) = \mathbb{E}_{(s^t,\boldsymbol{a}^t,s_{t+1})\sim\mathcal{D}}\left[\left(y^t - Q(s^t,\boldsymbol{a}^t;\phi_j)\right)^2\right], \tag{11}$$

with

$$y^t = r^t + \gamma \min_{j=1,\dots,K} Q(s^{t+1}, \pi_1(o_1^{t+1};\theta_1') + \epsilon, \dots, \pi_n(o_N^{t+1};\theta_N') + \epsilon; \phi_j). \tag{12}$$

And the policies are updated by the mean estimation of the ensembles as

$$\nabla\mathcal{J}(\theta_i) = \mathbb{E}_{s^t\sim\mathcal{D}}\left[\nabla_{\theta_i}\pi_i(o_i^t;\theta_i)\nabla_{a_i}\frac{1}{K}\sum_{j=1}^{K}\left[Q(s^t,a_1,\dots,a_N|_{a_i=\pi_i(o_i^t;\theta_i)};\phi_j)\right]\right]. \tag{13}$$

♣ **Critic ensembles diversity regularization** As in Section 3.2, we also apply diversity regularization to critic masks to prevent critics functions from collapsing to identical ones. The diversity regularization to maximize for the critic ensembles is expressed as

$$\mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c) = \sum_{i=1,\dots,K}\sum_{\substack{j=1,\dots,K\\ j\neq i}} \|\phi_0 \odot (\boldsymbol{M}_i^c - \boldsymbol{M}_j^c)\|_1. \tag{14}$$

Intuitively, as training progresses, this term encourages divergence among the critic masks, leading to increased model estimation uncertainty. This process fosters a gradual shift from overestimation to underestimation. As discussed in prior research (Hiraoka et al., 2022; Lan et al., 2020; Chen et al., 2021; Wang et al., 2021b), overestimation can encourage exploration, beneficial in early training stages, whereas underestimation alleviates error accumulation (Fujimoto et al., 2018), which is preferred in the late training stage. We formally provide the overall training objective for critic ensembles in Appendix A.1.1.

♠ **Periodically reset** To further promote diversity among critic ensembles and counteract the reduction in network capacity caused by masking, we implement a resetting mechanism similar to that described in Section 3.3. In particular, we sequentially reinitialize the masks $\boldsymbol{M}_j^c$ following a cyclic pattern, as illustrated in Figure 3b. In this way, each critic function's mask is trained on distinct data segments, leading to different biases.

In summary, by adopting Kaleidoscope parameter sharing with learnable masks, we establish a cost-effective implementation for critic ensembles that enjoy a high UTD ratio. Through enforcing distinctiveness among the masks, we subtly control the differences among critic functions, thereby improving the value estimations in MARL.

4 Experimental Results

In this section, we integrate Kaleidoscope with the value-based MARL algorithm QMIX and the actor-critic MARL algorithm MATD3, and evaluate them across eleven scenarios in three benchmark tasks.

Table 1: Methods compared in the experiments. Here, "adaptive" indicates whether the sharing scheme evolves during training.

Methods	Paradigm	Sharing level	Adaptive	Descriptions
NoPS	No sharing	-	No	Agents have distinct parameters
FuPS	Full sharing	Networks	No	Agents share all the parameters
FuPS + ID	Full sharing	Networks	No	Agents share all the parameters with agent IDs in input
SePS	Partial sharing	Networks	No	Agents are clustered to share parameters within each cluster
MultiH	Partial sharing	Layers	No	Agents share all the parameters except for distinct action heads
SNP	Partial sharing	Neurons	No	Agents share specific neurons based on fixed, random pruning
Kaleidoscope	Partial sharing	Weights	Yes	Agents share parameters based on distinct, learnable masks

4.1 Experimental Setups

Environment descriptions We test our proposed Kaleidoscope on three benchmark tasks: MPE (Lowe et al., 2017), MaMuJoCo (Peng et al., 2021) and SMACv2 (Ellis et al., 2024). For the discrete tasks MPE and SMACv2, we integrate Kaleidoscope and baselines with QMIX (Rashid et al., 2020) and assess the performance. For the continuous task MaMuJoCo, we employ MATD3 (Ackermann et al., 2019). We use five random seeds for MPE and MaMuJoCo and three random seeds for SMACv2, reporting averaged results and displaying the 95% confidence interval with shaded areas. The chosen benchmark tasks reflect a mix of discrete and continuous action spaces and both homogeneous and heterogeneous agent types, detailed further in Appendix A.2.

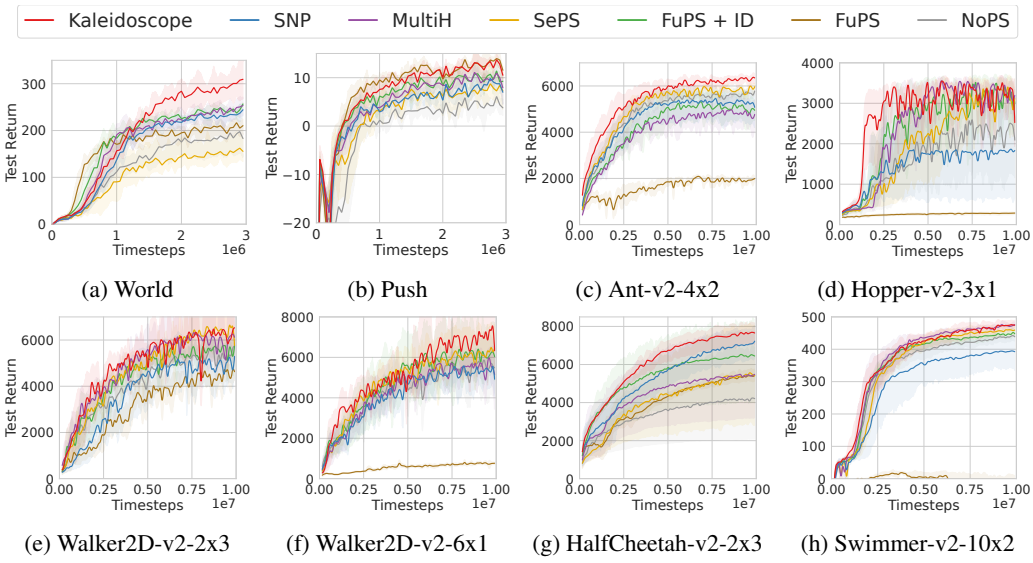
Baselines In the following, we compare our proposed Kaleidoscope with baselines (Christianos et al., 2021; Kim and Sung, 2023), as listed in Table 1. For both Kaleidoscope and the baselines, in scenarios with fixed agent types (MPE and MaMuJoCo), we assign one mask per agent. For SMACv2, where agent types vary, we assign one mask per agent type. We use official implementations of the baselines where available; otherwise, we closely follow the descriptions from their respective papers, integrating them into QMIX or MATD3. Hyperparameters and further details are provided in Appendix A.1.3.

4.2 Results

Performance We present the comparative performance of Kaleidoscope and baselines in Figure 4 and Figure 5. Overall, Kaleidoscope demonstrates superior performance, attributable to the flexibility of the learnable masks and the effectiveness of diversity regularization. Additionally, we observe that FuPS + ID generally outperforms NoPS, except for the `Ant-v2-4x2` scenario (Figure 4c). This advantage is largely due to FuPS's higher sample efficiency; a single transition data sample updates the model parameters N times in FuPS + ID, once for each agent, compared to just once in NoPS. Consequently, FuPS + ID models learn faster from the same number of transitions. Similarly, Kaleidoscope benefits from this mechanism as it shares weights among agents, allowing a single transition to update the model parameters multiple times. Furthermore, by integrating policy heterogeneity through learnable masks, Kaleidoscope enables diverse agent behaviors, as illustrated in the visualization results in Figure 8. Ultimately, Kaleidoscope effectively balances parameter sharing and diversity, outperforming both full parameter sharing and non-parameter sharing approaches.

Cost analysis Despite its superior performance, Kaleidoscope does not increase computational complexity at test time compared to the baselines. We report the test time averaged FLOPs comparison of Kaleidoscope and baselines in Table 2. We see that due to the masking technique, Kaleidoscope has lower FLOPs compared to baselines, thereby enjoying a faster inference speed when being deployed.

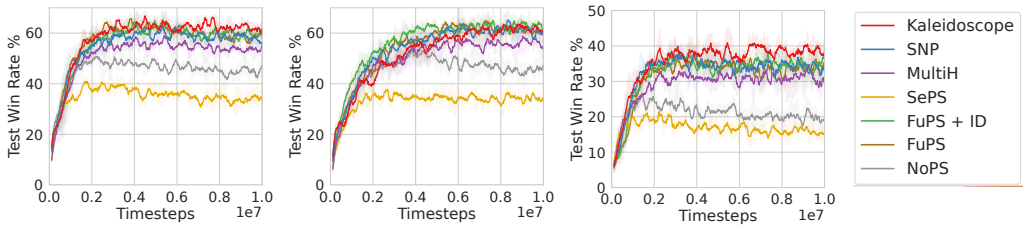
https://doi.org/10.52202/079017-0695



(a) World (b) Push (c) Ant-v2-4x2 (d) Hopper-v2-3x1

(e) Walker2D-v2-2x3 (f) Walker2D-v2-6x1 (g) HalfCheetah-v2-2x3 (h) Swimmer-v2-10x2

Figure 4: Performance comparison with baselines on MPE and MaMuJoCo benchmarks.



(a) Terran_5_vs_5 (b) Protoss_5_vs_5 (c) Zerg_5_vs_5

Figure 5: Performance comparison with baselines on SMACv2 benchmarks.

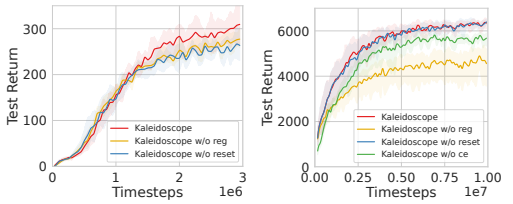
Ablation studies We conduct ablation studies to assess the impact of key components in Kaleidoscope, with results presented in Figure 6. Specifically, we compare Kaleidoscope with three ablations: 1) *Kaleidoscope w/o reg*, which lacks the regularization term in Equation (8) that encourages the masks to be distinct. 2) *Kaleidoscope w/o reset*, which does not reset parameters. 3) *Kaleidoscope w/o ce*, which does not use Kaleidoscope parameter sharing in critic ensembles and instead maintains two independent sets of parameters for critics. From the results, we observe that diversity regularization contributes the most to the performance of Kaleidoscope. Without it, masking degrades the performance due to the reduced number of parameters in each policy network. Resetting primarily aids learning in the late stages of training when needed, which aligns with the observation made by Nikishin et al. (2022). Notably, even with resetting, the performance does not experience abrupt drops thanks to the guidance provided by the masks on where to reset. When ablating the critic ensembles with Kaleidoscope parameter sharing, we observe inferior performance from the beginning of the training. This is because the critic ensembles with Kaleidoscope parameter sharing enable a higher UTD ratio of the critics, as discussed in Section 3.4.

Furthermore, we conduct experiments to study the impact of mask designs. The results are shown in Figure Figure 7. Specifically, we compare original Kaleidoscope with two alternative mask design choices: 1) *Kaleidoscope w/ neuron masks*, where adaptive masking techniques are applied to neurons rather than weights. 2) *Kaleidoscope w/ fixed masks*, where the masks are initialized at the beginning of training and kept fixed throughout the learning process. The results show that performance drops with either alternative design choice, demonstrating that Kaleidoscope's superior performance originates from the flexibility of the learnable masks on weights.

More results on hyperparameter analysis are included in Appendix B.2.

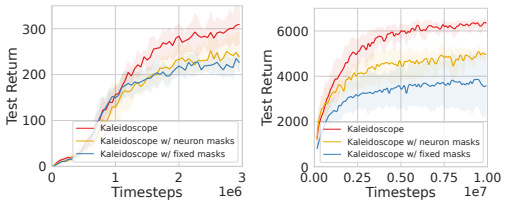
Table 2: Averaged FLOPs (with calculation methods detailed in Appendix A.3) across different methods. Results are first normalized with respect to the FuPS + ID model for each scenario and then averaged across scenarios within each environment (detailed results in Appendix B.1). The lowest costs are highlighted in **bold**.

Methods	NoPS	FuPS	FuPS +ID	SePS	MultiH	SNP	Kaleidoscope
MPE	1.0x	0.992x	1.0x	1.0x	1.0x	0.988x	**0.901x**
MaMuJoCo	1.0x	0.985x	1.0x	1.0x	1.0x	0.900x	**0.680x**
SMACv2	1.0x	0.992x	1.0x	1.0x	1.0x	0.988x	**0.890x**



(a) World (b) Ant-v2-4x2

Figure 6: Ablation studies.



(a) World (b) Ant-v2-4x2

Figure 7: Comparison on mask designs.

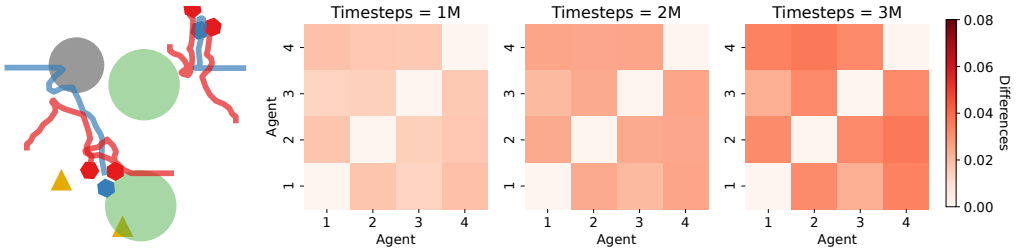
Visualization We visualize the trained policies of Kaleidoscope on `World`, as shown in Figure 8a. The agents exhibit cooperative divide-and-conquer strategies (four red agents divide into two teams and surround the preys), contrasting with the homogeneous policies depicted in Figure 1. We further examine the distinctions in the agents' masks and present the results in Figure 8b. First, we observe that by the end of the training, each agent has developed a unique mask, revealing that distinct masks facilitate diverse policies by selectively activating different segments of the neural network weights. Second, throughout the training process, we note that the differences among the agents' masks evolve dynamically. This observation confirms that Kaleidoscope effectively enables dynamic parameter sharing among the agents based on the learning progress, empowered by the adaptability of the learnable masks. More visualization results are provided in Appendix B.3.

5 Related Work

Parameter sharing First introduced by Tan (1993), parameter sharing has been widely adopted in MARL algorithms (Foerster et al., 2018; Rashid et al., 2020; Yu et al., 2022), due to its simplicity and high sample efficiency (Grammel et al., 2020). However, schemes without parameter sharing typically offer greater flexibility for policy representation. To balance sample efficiency with policy representational capacity, some research efforts aim to find effective partial parameter sharing schemes. Notably, SePS (Christianos et al., 2021) first clusters agents based on their transitions at the start of training and restricts parameter sharing within these clusters. Subsequently, SNP (Kim and Sung, 2023) enables partial parameter sharing by utilizing the lottery ticket hypothesis (Su et al., 2020) to initialize heterogeneous network structures. Concurrent to our work, AdaPS (Li et al., 2024) combines SNP and SePS by proposing a cluster-based partial parameter sharing scheme. While these methods have shown promise in certain domains, their performance potential is often limited by the static nature of the parameter sharing schemes set early in training. Our proposed Kaleidoscope distinguishes itself by dynamically learning specific parameter sharing configurations alongside the development of MARL policies, thereby offering enhanced training flexibility.

Agent heterogeneity in MARL To incorporate agent heterogeneity in MARL and enable diverse behaviors among agents, previous methods have explored concepts such as diversity and roles. Specifically, diversity-based approaches aim to enhance pairwise distinguishability among agents based on identities (Jiang and Lu, 2021), trajectories (Li et al., 2021), or credits assignment (Liu et al., 2023; Hu et al., 2023) through contrastive learning techniques. Concurrently, role-based strategies, sometimes referred to as skills (Yang et al., 2020) or subtasks (Yuan et al., 2022), employ conditional policies to differentiate agents by assigning them to various conditions. These conditions may be based on agent identities (Yang et al., 2022), local observations (Yang et al., 2020), local

https://doi.org/10.52202/079017-0695



(a) Trained policies. (b) Pairwise mask differences among agents at different training timesteps.

Figure 8: Visualization on `World`.

histories (Wang et al., 2020, 2021a; Yuan et al., 2022) or joint histories (Liu et al., 2021; Iqbal et al., 2022; Zeng et al., 2023). This line of researches mainly focus on module design and operate separately from parameter-level adjustments, making them orthogonal to our approach. Nevertheless, integrating these methods with our work could potentially enhance performance further.

Sparse networks in deep reinforcement learning (RL) Although relatively few, there are some noteworthy recent attempts to find sparse networks for deep RL. In particular, PoPS (Livne and Cohen, 2020) prunes the dense networks post-training, achieving significantly reduced execution time complexity. Additionally, (Yu et al., 2020) validate the lottery ticket hypothesis within the RL domain, producing high-performance models even under extreme pruning rates. Subsequent efforts, including DST (Sokar et al., 2022), TE-RL* (Graesser et al., 2022) and RLx2 (Tan et al., 2023) employ topology evolution (TE) techniques to further decrease the training costs. While these developments utilize sparse training techniques, which are similar to the methods we employ, their primary focus is on reducing training and execution costs in single-agent settings. In contrast, our work leverages sparse network strategies as a means to enhance parameter sharing techniques, aiming to improve MARL performance.

6 Conclusions and Future Work

In this work, we introduced *Kaleidoscope*, a novel adaptive partial parameter sharing mechanism for MARL. It leverages distinct learnable masks to facilitate network heterogeneity, applicable to both agent policies and critic ensembles. Specifically, Kaleidoscope is built on three technical components: STR-empowered learnable masks, network diversity regularization, and a periodic resetting mechanism. When applied to agent policy networks, Kaleidoscope balances sample efficiency and network representational capacities. In the context of critic ensembles, it improves value estimations. By combining our proposed Kaleidoscope with QMIX and MATD3, we have empirically demonstrated its effectiveness across various MARL benchmarks. This study shows great promises in developing adaptive partial parameter sharing mechanisms to enhance the performance of MARL. For future work, it is interesting to further extend Kaleidoscope to other domains such as offline MARL or meta-RL.

Acknowledgements

This work was supported by the Hong Kong Research Grants Council under the NSFC/RGC Collaborative Research Scheme grant CRS_HKUST603/22. And we thank the anonymous reviewers for their valuable feedback and suggestions.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

Johannes Ackermann, Volker Gabler, Takayuki Osa, and Masashi Sugiyama. 2019. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465* (2019).

Milad Alizadeh, Javier Fernández-Marqués, Nicholas D Lane, and Yarin Gal. 2018. An empirical study of binary neural networks' optimisation. In *Proceedings of the International Conference on Learning Representations*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. 2021. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *Proceedings of the 9th International Conference on Learning Representations*.

Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht. 2021. Scaling Multi-Agent Reinforcement Learning with Selective Parameter Sharing. In *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139. PMLR, 1989–1998.

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. 2020. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proc. IEEE* 108, 4 (2020), 485–532.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. 2024. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 36.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2943–2952.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 29.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. 2022. Revisiting Some Common Practices in Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 6863–6877.

Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. PMLR, 1582–1591.

Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. 2022. The state of sparse training in deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 7766–7792.

Nathaniel Grammel, Sanghyun Son, Benjamin Black, and Aakriti Agrawal. 2020. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625* (2020).

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. 2022. Dropout Q-Functions for Doubly Efficient Reinforcement Learning. In *The Tenth International Conference on Learning Representations*.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research* 22, 241 (2021), 1–124.

Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih wei Liao. 2021. Rethinking the Implementation Tricks and Monotonicity Constraint in Cooperative Multi-Agent Reinforcement Learning. (2021). arXiv:2102.03479 [cs.LG]

 https://doi.org/10.52202/079017-0695

Zican Hu, Zongzhang Zhang, Huaxiong Li, Chunlin Chen, Hongyu Ding, and Zhi Wang. 2023. Attention-Guided Contrastive Role Representations for Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2312.04819* (2023).

Shariq Iqbal, Robby Costales, and Fei Sha. 2022. ALMA: Hierarchical learning for composite multi-agent tasks. In *Advances in Neural Information Processing Systems*, Vol. 35. 7155–7166.

Jiechuan Jiang and Zongqing Lu. 2021. The emergence of individuality. In *Proceedings of the International Conference on Machine Learning*. PMLR, 4992–5001.

Woojun Kim and Youngchul Sung. 2023. Parameter Sharing with Network Pruning for Scalable Multi-Agent Deep Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 1942–1950.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. 2020. Soft threshold weight reparameterization for learnable sparsity. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5544–5555.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. 2020. Maxmin Q-learning: Controlling the Estimation Bias of Q-learning. In *Proceedings of the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, April 26-30, 2020*.

Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. 2021. Celebrating diversity in shared multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 34. 3991–4002.

Dapeng Li, Na Lou, Bin Zhang, Zhiwei Xu, and Guoliang Fan. 2024. Adaptive parameter sharing for multi-agent reinforcement learning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6035–6039.

Bo Liu, Qiang Liu, Peter Stone, Animesh Garg, Yuke Zhu, and Anima Anandkumar. 2021. Coach-player multi-agent reinforcement learning for dynamic team composition. In *Proceedings of the International Conference on Machine Learning*. PMLR, 6860–6870.

Shunyu Liu, Yihe Zhou, Jie Song, Tongya Zheng, Kaixuan Chen, Tongtian Zhu, Zunlei Feng, and Mingli Song. 2023. Contrastive identity-aware learning for multi-agent value decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11595–11603.

Dor Livne and Kobi Cohen. 2020. Pops: Policy pruning and shrinking for deep reinforcement learning. *IEEE Journal of Selected Topics in Signal Processing* 14, 4 (2020), 789–801.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, Vol. 30.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. 2023. Understanding plasticity in neural networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 23190–23211.

Gaurav Menghani. 2023. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *Comput. Surveys* 55, 12 (2023), 1–37.

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. 2024. Deep reinforcement learning with plasticity injection. In *Advances in Neural Information Processing Systems*, Vol. 36.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. 2022. The primacy bias in deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 16828–16847.

Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs*. Springer.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. http://arxiv.org/abs/2006.07869

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. Facmac: Factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems*, Vol. 34. 12208–12221.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. 2020. What's hidden in a randomly weighted neural network?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11893–11902.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research* 21, 1 (2020), 7234–7284.

Sven Seuken and Shlomo Zilberstein. 2007. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*. 344–351.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *Proceedings of the International Conference on Machine Learning*. Pmlr, 387–395.

Ghada Sokar, Elena Mocanu, Decebal Constantin Mocanu, Mykola Pechenizkiy, and Peter Stone. 2022. Dynamic Sparse Training for Deep Reinforcement Learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. ijcai.org, 3437–3443.

Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D Lee. 2020. Sanity-checking pruning methods: Random tickets can win the jackpot. In *Advances in Neural Information Processing Systems*, Vol. 33. 20390–20401.

Gokul Swamy, Siddharth Reddy, Sergey Levine, and Anca D Dragan. 2020. Scaled autonomy: Enabling human operators to control robot fleets. In *2020 IEEE International Conference on Robotics and Automation*. IEEE, 5942–5948.

Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning*. 330–337.

Yiqin Tan, Pihe Hu, Ling Pan, Jiatai Huang, and Longbo Huang. 2023. RLx2: Training a Sparse Deep Reinforcement Learning Model from Scratch. In *Proceedings of the Eleventh International Conference on Learning Representations*.

Hang Wang, Sen Lin, and Junshan Zhang. 2021b. Adaptive Ensemble Q-learning: Minimizing Estimation Bias via Error Feedback. In *Advances in Neural Information Processing Systems*. 24778–24790.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. 2020. ROMA: multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning*. 9876–9886.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. 2021a. RODE: Learning Roles to Decompose Multi-Agent Tasks. In *Proceedings of the 9th International Conference on Learning Representations*.

Jiachen Yang, Igor Borovikov, and Hongyuan Zha. 2020. Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 1566–1574.

Mingyu Yang, Jian Zhao, Xunhan Hu, Wengang Zhou, Jiangcheng Zhu, and Houqiang Li. 2022. LDSA: Learning dynamic subtask assignment in cooperative multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 35. 1698–1710.

 https://doi.org/10.52202/079017-0695

Wang Ying and Sang Dayong. 2005. Multi-agent framework for third party logistics in E-commerce. *Expert Systems with Applications* 29, 2 (2005), 431–436.

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

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2020. Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP. In *Proceedings of the 8th International Conference on Learning Representations*.

Lei Yuan, Chenghe Wang, Jianhao Wang, Fuxiang Zhang, Feng Chen, Cong Guan, Zongzhang Zhang, Chongjie Zhang, and Yang Yu. 2022. Multi-Agent Concentrative Coordination with Decentralized Task Representation.. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. 599–605.

Lei Yuan, Ziqian Zhang, Lihe Li, Cong Guan, and Yang Yu. 2023. A Survey of Progress on Cooperative Multi-agent Reinforcement Learning in Open Environment. *arXiv preprint arXiv:2312.01058* (2023).

Xianghua Zeng, Hao Peng, and Angsheng Li. 2023. Effective and stable role-based multi-agent collaboration by structural information principles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11772–11780.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

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

Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, et al. 2021. Smarts: An open-source scalable multi-agent rl training school for autonomous driving. In *Proceedings of the Conference on Robot Learning*. PMLR, 264–285.

A Experimental details

A.1 Implementation details

A.1.1 Kaleidoscope with MATD3

Critic ensembles When incorporating Kaleidoscope into the MATD3 algorithm, the overall training loss for the critic ensembles becomes:

$$\mathcal{L}_c^{\text{all}}(\phi_0, \boldsymbol{s}^c) = \sum_{j=1,\dots,K} \mathcal{L}_c^{\text{all}}(\phi_j) = \sum_{j=1,\dots,K} \mathcal{L}_c(\phi_0, \boldsymbol{s}_j^c) - \alpha^d \cdot \mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c), \tag{15}$$

with $\mathcal{L}_c(\phi_0, \boldsymbol{s}_j^c)$ being the original MATD3 loss given in Equation (11), $\mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c)$ being the diversity regularization given in Equation (14), and α^d being a coefficient balancing the original MARL objective and the proposed diversity regularization. Note that although $\mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c)$ contains parameters ϕ_0, we stop the gradients for ϕ_0 in $\mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c)$.

In the implementation, we apply layer-wise weights to the diversity regularization term $\mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c)$, which is defined as

$$\mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c) = \sum_{l=1,\dots,L} w_l \cdot \sum_{i=1,\dots,K} \sum_{\substack{j=1,\dots,K \\ j\neq i}} \|\phi_0 \odot (\boldsymbol{M}_{i,l}^c - \boldsymbol{M}_{j,l}^c)\|_1, \tag{16}$$

where l denotes the layer index of the neuron networks, L represents the total number of layers, $\boldsymbol{M}_{i,l}^c$ is the mask for agent i at layer l, and the layer-wise weights are set as $w_l = 2^l$. The intuition behind this choice is that features closer to the output tend to be more compact (Kusupati et al., 2020); consequently, assigning larger regularization weights to these layers may have a more significant impact on the output action decisions. Our initial experiments empirically demonstrate that setting $w_l = 2^l$ improves performance compared to the case where $w_l = 1$. Based on these findings, we maintain this design choice throughout all our experiments.

In practice, we adaptively adjust α^d while maintaining a constant ratio between the MATD3 loss and the diversity loss, which is treated as a hyperparameter:

$$\alpha^d = \frac{|\sum_{j=1,\dots,K} \mathcal{L}_c(\phi_0, \boldsymbol{s}_j^c)|}{|\mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c)|} \cdot \alpha, \tag{17}$$

where α is a hyperparameter, and the gradients for $\frac{|\sum_{j=1,\dots,K} \mathcal{L}_c(\phi_0, \boldsymbol{s}_j^c)|}{|\mathcal{J}_c^{\text{div}}(\boldsymbol{s}^c)|}$ are stopped.

Actors For the actors, the training objective is to maximize the following term

$$\mathcal{J}^{\text{all}}(\theta_0, \boldsymbol{s}) = \sum_{i=1,\dots,n} \mathcal{J}^{\text{all}}(\theta_i) = \sum_{i=1,\dots,n} \mathcal{J}(\theta_0, \boldsymbol{s}_i) + \beta^d \cdot \mathcal{J}^{\text{div}}(\boldsymbol{s}), \tag{18}$$

where $\mathcal{J}(\theta_0, \boldsymbol{s}_i)$ is the original actor objective defined in Equation (3), $\mathcal{J}^{\text{div}}(\boldsymbol{s})$ is the diversity regularization given in Equation (8) with layer-wise weights as in Equation (16), β^d is the regularization coefficient. The value of β^d is determined by

$$\beta^d = \frac{|\sum_{i=1,\dots,n} \mathcal{J}(\theta_0, \boldsymbol{s}_i)|}{|\mathcal{J}^{\text{div}}(\boldsymbol{s})|} \cdot \beta, \tag{19}$$

where β is a constant hyperparameter, similar to the approach used in Equation (17) for the critic ensembles.

A.1.2 Kaleidoscope with QMIX

When incorporating Kaleidoscope into the QMIX algorithm (Rashid et al., 2020), we apply Kaleidoscope parameter sharing only to the local Q networks. Consequently, the training loss is defined as:

$$\mathcal{L}^{\text{all}}(\theta_0, \boldsymbol{s}) = \mathcal{L}(\theta_0, \boldsymbol{s}) - \beta^d \cdot \mathcal{J}^{\text{div}}(\boldsymbol{s}), \tag{20}$$

where

$$\mathcal{L}(\theta_0, \boldsymbol{s}) = \mathbb{E}_{(s^t, \boldsymbol{o}^t, \boldsymbol{a}^t, r^t, s^{t+1}, \boldsymbol{o}^{t+1}) \sim \mathcal{D}} \left[\left(y^{tot} - Q_{tot}(s^t, \boldsymbol{o}^t, \boldsymbol{a}^t; \theta_0, \boldsymbol{s}) \right)^2 \right], \tag{21}$$

with $y^{tot} = r + \gamma \max_{\boldsymbol{a}} Q_{tot}(s^{t+1}, \boldsymbol{o}^{t+1}, \boldsymbol{a}; \theta^-)$ and θ^- representing the parameters of a target network as in DQN.

https://doi.org/10.52202/079017-0695

A.1.3 Network architecture and hyperparameters

Codebase Our implementation of Kaleidoscope and baseline algorithms are based on the following codebase:

- HARL (Zhong et al., 2024) (MATD3 implementation): `https://github.com/PKU-MARL/HARL`
- EPyMARL (Papoudakis et al., 2021) (QMIX implementation for MPE): `https://github.com/uoe-agents/epymarl`
- PyMARL2 (Hu et al., 2021) (QMIX implementation for SMACv2): `https://github.com/benellis3/pymarl2`
- SePS (Christianos et al., 2021): `https://github.com/uoe-agents/seps`

The code for Kaleidoscope is publicly available at `https://github.com/LXXXXR/Kaleidoscope`.

Network architecture In line with prior works (Zhong et al., 2024; Papoudakis et al., 2021; Hu et al., 2021), we employ deep neural networks consisting of multilayer perceptrons (MLPs) with rectified linear unit (ReLU) activation functions and gated recurrent units (GRUs) to parameterize the actor and critic networks. Moreover, when the masking technique is applied to critic ensembles, we incorporate layer normalization between the MLP layers and ReLU activations (Hiraoka et al., 2022). In Kaleidoscope, the masking technique is applied to the MLP layers, and the resetting mechanisms described in Sections 3.3 and 3.4 are applied to the last three layers of the respective neural networks, following Nikishin et al. (2022).

Hyperparameters To ensure a fair comparison, we implement our method and all the baselines using the same codebase with the same set of hyperparameters, with the exception of method-specific ones. The common hyperparameters are listed in Tables 3 to 5. The Kaleidoscope-specific hyperparameters are provided in Table 6.

Table 3: Common hyperparameters used for MATD3 in the MaMuJoCo domain.

Hyperparameter	Value
Number of layers	3
Hidden sizes	256
Discount factor γ	0.99
Rollout threads	10
Critic lr	1×10^{-3}
Actor lr	5×10^{-4}
Exploration noise	0.1
Batch size	1000
Replay buffer size	1×10^{6}
Number of environment steps	10×10^{6}
n_step[1]	$(5, 10, 20)$

[1] Here we adopt the per-scenario finetuned value for this hyperparameter as provided by HARL.

A.2 Environmental details

Codebase The environments used in this work are listed below with descriptions in Table 7.

- MaMuJoCo (Peng et al., 2021): `https://github.com/schroederdewitt/multiagent_mujoco`
- MPE (Lowe et al., 2017; Papoudakis et al., 2021): `https://github.com/semitable/multiagent-particle-envs`
- SMACv2 (Ellis et al., 2024): `https://github.com/oxwhirl/smacv2`

Table 4: Common hyperparameters used for QMIX in the MPE domain.

Hyperparameter	Value
Number of layers	5
Hidden sizes	64
Discount factor γ	0.99
Lr	5×10^{-4}
Initial ϵ	1.0
Final ϵ	0.05
Batch size	32
Replay buffer size	5000
Number of environment steps	3×10^6
Double Q	True

Table 5: Common hyperparameters used for QMIX in the SMACv2 domain.

Hyperparameter	Value
Number of layers	5
Hidden sizes	64
Discount factor γ	0.99
Lr	1×10^{-3}
Initial ϵ	1.0
Final ϵ	0.05
Batch size	128
Replay buffer size	5000
Number of environment steps	10×10^6
Double Q	False

MPE We extend the scenario settings provided in the original codebase to increase the complexity and challenge of the tasks. In `World`, we set the number of predators (agents) to 4, the number of prey to 2, the number of obstacles to 1, and the number of forests to 2. The objective of the game is for the predators to approach the prey while avoiding collisions with obstacles. The prey is attracted to the food and can hide from the predators in the forests. In `Push`, we set the number of agents to 5, the number of adversaries to 2, and the number of landmarks to 2. The goal of the game is for the agents to push the adversaries away from the landmarks. In both scenarios, we pretrain the adversary (prey) policies using the MADDPG algorithm Lowe et al. (2017) and use these pretrained policies to test the performance of different algorithms

A.3 FLOPs calculation

To calculate the number of floating-point operations (FLOPs) for a single forward pass of a sparse model, we sum the total number of multiplications and additions layer by layer, following the approach in Evci et al. (2020). For a fully-connected layer, the FLOPs are computed as follows:

$$\text{FLOPs} = 2 \times (1 - \text{Sparsity}) \times \text{In} \times \text{Out}. \tag{22}$$

For a GRU cell, the FLOPs are computed as:

$$\text{FLOPs} = 2 \times (3 \times \text{Hidden}^2 + 3 \times \text{In} \times \text{Hidden} + 13 \times \text{Hidden}). \tag{23}$$

A.4 Experimental Infrastructure

The experiments on the SMACv2 benchmark were conducted using NVIDIA GeForce RTX 3090 GPUs, while the experiments on other benchmarks were performed using NVIDIA GeForce RTX 3080 GPUs. Each experimental run required less than 2 days to complete.

https://doi.org/10.52202/079017-0695

Table 6: Hyperparameters used for Kaleidoscope.

Hyperparameter	Environment	Value
Actor diversity coefficient β	MaMuJoCo	0.1
	MPE	0.5
	SMACv2	5.0
Actors reset probability ρ	MaMuJoCo	0.5
	MPE	0.1
	SMACv2	0.2
Actor reset interval	MaMuJoCo	1×10^6
	MPE	200×10^3
	SMACv2	1×10^6
Number of critic ensembles K	MaMuJoCo	5
Critic ensembles diversity coefficient α	MaMuJoCo	0.1
Critic reset interval	MaMuJoCo	800×10^3

Table 7: Environments details.

Environment	Action space	Agent types	Scenarios	Number of agents
MaMuJoCo	Continuous	Heterogeneous, fixed	Ant-v2-4x2	4
			Hopper-v2-3x1	3
			Walker2D-v2-2x3	2
			Walker2D-v2-6x1	6
			HalfCheetah-v2-2x3	2
			Swimmer-v2-10x2	10
MPE	Discrete	Homogeneous, fixed	World	4
			Push	5
SMACv2	Discrete	Heterogeneous, dynamic	Terran_5_vs_5	5
			Protoss_5_vs_5	5
			Zerg_5_vs_5	5

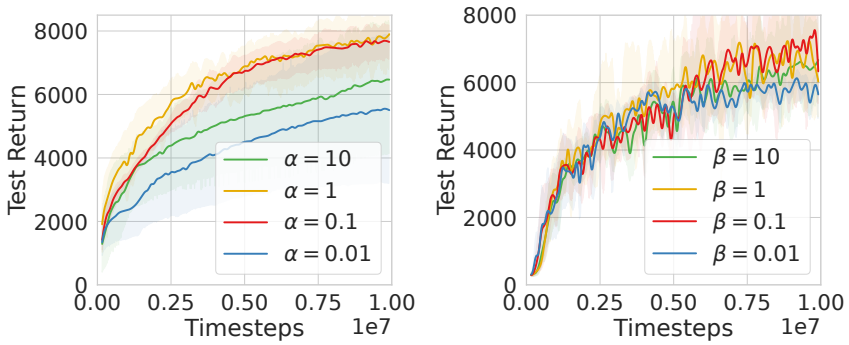
B More results and discussion

B.1 Detailed Costs

We provide per-scenario FLOPs across different methods in Table 8 as a supplement for Table 2.

Table 8: Averaged FLOPs for different methods. Results are normalized w.r.t. the FuPS + ID model. The lowest costs are highlighted in **bold**.

Scenarios	NoPS	FuPS	FuPS +ID	SePS	MultiH	SNP	Kaleidoscope
World	1.0x	0.993x	1.0x	1.0x	1.0x	0.988x	**0.897x**
Push	1.0x	0.991x	1.0x	1.0x	1.0x	0.988x	**0.904x**
Ant-v2-4x2	1.0x	0.990x	1.0x	1.0x	1.0x	0.900x	**0.640x**
Hopper-v2-3x1	1.0x	0.989x	1.0x	1.0x	1.0x	0.900x	**0.721x**
Walker2D-v2-2x3	1.0x	0.992x	1.0x	1.0x	1.0x	0.900x	**0.731x**
Walker2D-v2-6x1	1.0x	0.979x	1.0x	1.0x	1.0x	0.900x	**0.763x**
HalfCheetah-v2-2x3	1.0x	0.993x	1.0x	1.0x	1.0x	0.900x	**0.614x**
Swimmer-v2-10x2	1.0x	0.968x	1.0x	1.0x	1.0x	0.900x	**0.611x**
Terran_5_vs_5	1.0x	0.992x	1.0x	1.0x	1.0x	0.988x	**0.890x**
Protoss_5_vs_5	1.0x	0.992x	1.0x	1.0x	1.0x	0.988x	**0.895x**
Zerg_5_vs_5	1.0x	0.992x	1.0x	1.0x	1.0x	0.988x	**0.885x**



(a) Critic ensembles diversity coefficient α. (b) Actors diversify coefficient β.

Figure 9: Hyperparameter analysis.

B.2 Hyperparameter Analysis

We conduct further analysis on the hyperparameters α and β, and present the results in Figure 9. The hyperparameter α controls the variance of the critic ensembles. As shown in Figure 9a, we observe that an excessively small α results in degraded performance because it reduces the critic ensembles to a single critic network, causing the value estimation to suffer from severe overestimation. Conversely, an excessively large α also deteriorates performance, possibly due to increased estimation bias. For β, as illustrated in Figure 9b, an overly small β leads to degraded performance because it reduces the Kaleidoscope parameter sharing to full parameter sharing, confining the policies to be identical. An overly large β also negatively impacts performance, as it may cause the training objective to deviate too much from minimizing the original MARL loss.

In general, we recommend setting both hyperparameters between 0.1 and 1. However, the optimal hyperparameter values may vary across different scenarios. For fair comparisons, we maintain the same set of hyperparameters across all scenarios in our experiments. Nevertheless, further tuning of these hyperparameters has the potential to enhance performance.

B.3 Further Visualization Results

To better understand how learnable masks in Kaleidoscope affect the performance through policies, we visualize the pairwise mask differences among agents and the agent trajectories at different training stages in Figure 10. As training progresses, the test return increases and diversity loss decreases, indicating better performance and greater diversity among agent policies. Correspondingly, mask differences among agents increase, and the agent trajectory distribution becomes more diverse.

B.4 Limitations

Here we discuss some limitations of Kaleidoscope.

First, as suggested by results in Appendix B.2, the optimal hyperparameters vary from scenario to scenario. Therefore, using the same hyperparameters across all scenarios may not yield the best performance for Kaleidoscope. Developing an automatic scheme that utilizes environmental information to determine these hyperparameters would be beneficial.

Second, as the environments used in this work contain no more than 10 agents, we assign a distinct mask for each agent. However, when the problem scales to hundreds of agents, this vanilla implementation may fail. In such cases, a possible approach is to cluster N agents into K ($K < N$) groups and train K masks with Kaleidoscope. This would reduce computational costs and achieve a better trade-off between sample efficiency and diversity. Within the same group, agents share all parameters, while agents from different groups share only partial parameters. Techniques for clustering agents based on experience, as proposed by Christianos et al. (2021), could be useful.

https://doi.org/10.52202/079017-0695

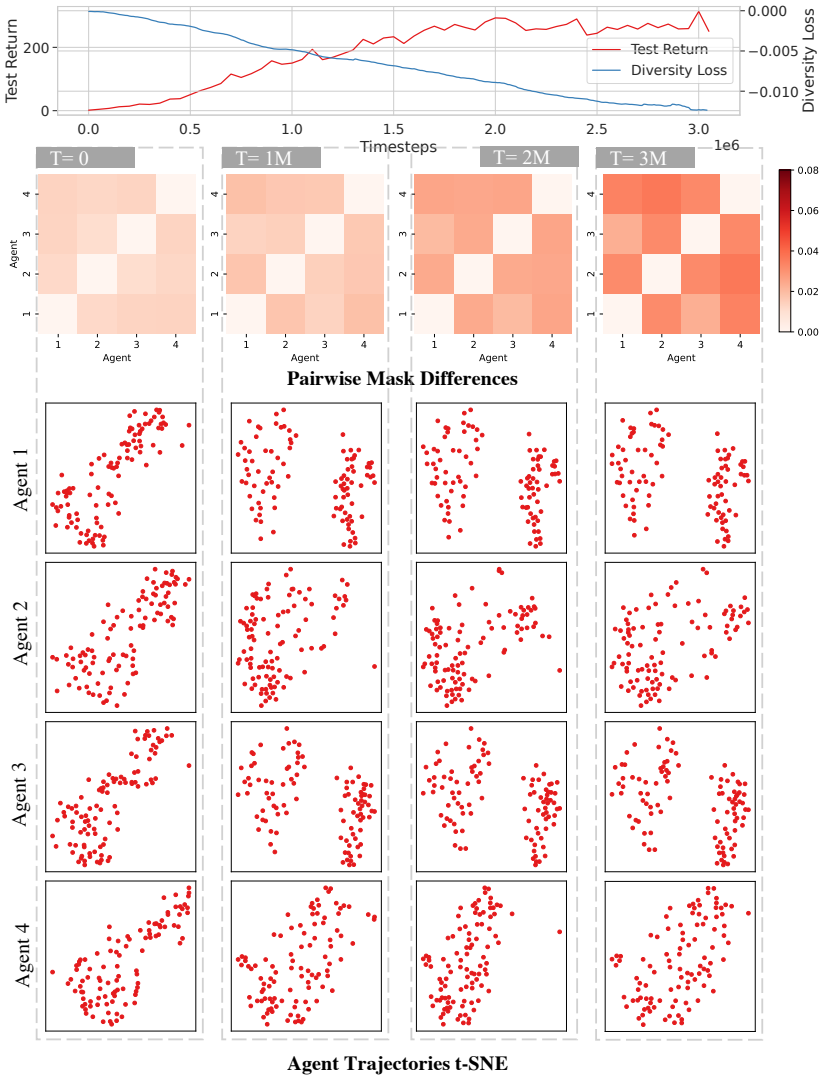


Figure 10: Further visualization on `World`.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions by proposing a novel parameter sharing technique in MARL (Section 3) and claiming its superior performance compared to existing methods, which is supported by the experimental results (Section 4).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.

- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see Appendix B.4.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

https://doi.org/10.52202/079017-0695

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The method and implementation details are provided in Section 3 and Appendix A.1, and the experiment settings and environment details are described in Section 4 and Appendix A.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is available at `https://github.com/LXXXXR/Kaleidoscope`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experiment settings are provided in Section 4 with details elaborated in Appendices A.1 and A.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main results are reported with 95% confidence error bars in Figures 4 to 6.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

https://doi.org/10.52202/079017-0695

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Please see Appendix A.4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This paper presents work whose goal is to advance the field of Reinforcement Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The experiments in this work are conducted in simulated game environments, thereby presenting minimal risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The papers corresponding to the environments used are cited in Section 4, and the codebases utilized are listed in Appendices A.1.3 and A.2.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

https://doi.org/10.52202/079017-0695

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.