SAW-V2: SUPERVISED DUAL-STREAM STATE ADAPTIVE WEIGHT FOR HIGH-DIMENSIONAL MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent reinforcement learning (MARL) has advanced significantly in complex robotic systems, enabling decentralized agents to acquire coordinated behaviors for locomotion, dexterous manipulation, and cooperative decision-making under the centralized training and decentralized execution (CTDE). However, existing MARL algorithms often process heterogeneous and high-dimensional observations in a largely uniform manner, which can dilute task relevant state information, overemphasize redundant or noisy features, and make actor-critic optimization less efficient and less stable in strongly coupled multi-agent environments. To address this limitation, we propose SAW-v2, a state adaptive weighting framework that learns stable state-importance weights for actor side representations. By identifying informative observation components and suppressing less useful ones, SAW-v2 provides a lightweight plug-in module for improving policy learning. 108 groups experiment on MAMuJoCo, DexHands, SMACv2, and MPE demonstrate that SAW-v2 consistently improves representative MARL baselines across both continuous control and discrete cooperative benchmarks, achieving average 12.3% performance gains and showing stronger robustness in heterogeneous and high-dimensional coordination tasks.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has become a central framework for learning coordinated behaviors in complex systems, including robotic control, embodied cooperation, and strategic decision making Hady et al. (2025). By training multiple agents to optimize a shared or coupled objective, MARL enables decentralized policies to solve tasks that are difficult for a single agent to handle. Recent methods based on centralized training with decentralized execution (CTDE) Zhou et al. (2023a), such as MAPPO, HAPPO, and HASAC, have achieved strong performance on a range of cooperative benchmarks Yu et al. (2022); Liu et al. (2024); Schulman et al. (2017). However, modern MARL still faces several practical challenges. In heterogeneous and high-dimensional environments, agents often receive observations containing both task-critical and redundant state components. Treating all observation dimensions uniformly can waste critic capacity, slow down policy improvement, and amplify optimization noise. These issues become more severe when agents control different physical parts, operate under partial observability, or must coordinate in strongly coupled continuous-control tasks.

This paper develops **SAW-v2**, a state adaptive weighting framework for complex robotic heterogeneous MARL. As illustrated in Figure 1, SAW-v2 acts as a plug-in actor-side weighting module that learns state importance weights and injects them into actor-critic training. This is motivated by the original State Adaptive Weighting (SAW) mechanism, which improves sample utilization by assigning adaptive weights to state inputs during critic learning Wang et al. (2026). While SAW provides a useful step toward state-aware MARL, its weighting signal is still indirectly coupled with value estimation and can become unstable when the task involves high-dimensional observations, long-horizon coordination, or heterogeneous agent roles. We propose SAW-v2 to revisit this design from the perspective of stable state-importance learning. Instead of relying on a loosely coupled weighting objective, SAW-V2 introduces a more reliable importance estimation process that encourages agents to emphasize informative state components while suppressing redundant or noisy ones. This

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.

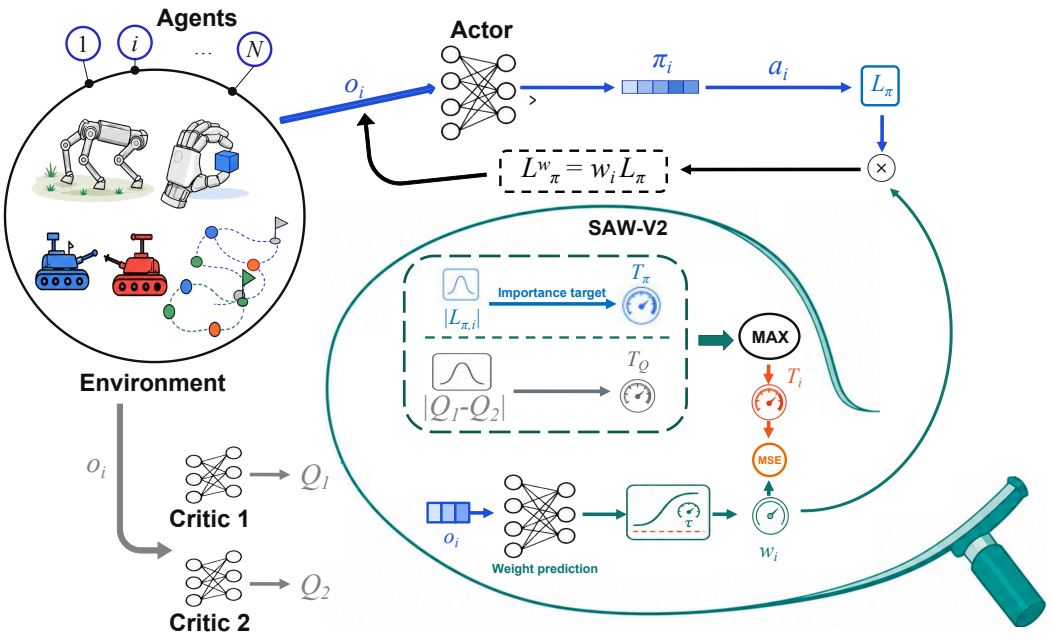


Figure 1: SAW-v2 proposes a framework that learns state adaptive weights through detached dual-stream supervision from policy difficulty and critic uncertainty, and performs loss-level weighting by applying the learned weights to per-sample actor objectives.

transition changes state weighting from an auxiliary heuristic into a more robust mechanism for improving critic learning and policy optimization in heterogeneous cooperative tasks.

We evaluate SAW-v2 on four complementary MARL benchmark families and 18 different tasks: MAMuJoCo, DexHands, SMACv2, and MPE. These environments cover continuous locomotion, high-dimensional dexterous manipulation, discrete strategic micromanagement, and lightweight particle-world coordination. Across these benchmarks, SAW-v2 consistently improves over representative baselines including MAPPO, HAPPO, and HASAC. We use the mean reward of the last 10% stage during the RL training process as the evaluation metric. SAW-v2 achieves an average improvement of 12.3% over the baseline method in 54 comparison experiments. The experiment results suggest that explicitly learning which state information to emphasize is a practical and effective direction for improving heterogeneous MARL.

2 Related Work

Cooperative multi-agent reinforcement learning (MARL) has increasingly extended from small discrete games toward physically grounded, heterogeneous, and large-scale coordination problems Albrecht et al. (2024); Zhou et al. (2023b). A dominant training paradigm is centralized training with decentralized execution (CTDE), where global information is available during training while each agent executes a decentralized policy. Earlier value-decomposition methods such as VDN, QMIX, QTRAN, and QPLEX remain useful references for discrete cooperative control Sunehag et al. (2017); Rashid et al. (2020); Son et al. (2019); Wang et al. (2021a), while later variants further improve credit assignment, policy-aware factorization, and high-order interaction modeling Wang et al. (2022); Zhou et al. (2022); Li et al. (2023); Bai et al. (2021). Recent work has increasingly emphasized policy optimization, benchmark coverage, and heterogeneous-agent learning. MAPPO shows that a carefully implemented PPO-style method can be a strong cooperative MARL baseline across diverse tasks Yu et al. (2022), and follow-up empirical studies highlight the importance of implementation details and evaluation protocols Fu et al. (2022). Other recent directions formulate cooperative MARL as sequence modeling or offline policy learning problems, including MAT, MADT, OMAR, and OMIGA Wen et al. (2022); Meng et al. (2023); Pan et al. (2022); Wang et al.

(2023), or study opponent shaping, constrained optimization, and safe multi-agent control Lu et al. (2022); Gu et al. (2021); Ying et al. (2023). Recent work has also broadened cooperative MARL beyond standard shared-reward formulations, including submodular team rewards and hierarchical critic structures for scalable coordination Chen et al. (2026); Eckel & Meeß (2026). Despite this progress, most methods primarily focus on value factorization, policy optimization, or training stability, while paying less attention to how centralized critics should adaptively identify and weight the most informative state components in heterogeneous coordination tasks.

Heterogeneous-agent MARL is especially relevant when agents have different observation spaces, action spaces, capabilities, or physical roles. Parameter sharing can improve sample efficiency in homogeneous teams, but it may suppress specialization when agents control different joints, limbs, or task components Terry et al. (2020); Bettini et al. (2023). HARL formalizes heterogeneous-agent policy optimization through multi-agent advantage decomposition and sequential policy updates, providing a principled alternative to homogeneous policy sharing Zhong et al. (2024). HAPPO and HATRPO instantiate this idea with PPO-style and trust-region updates Kuba et al. (2021), while HASAC introduces a maximum-entropy actor-critic objective for heterogeneous-agent continuous control Liu et al. (2024). Related representation and communication methods improve information aggregation among agents through attention, shared information, or theory-of-mind modeling Iqbal & Sha (2019); Nayak et al. (2023); Guan et al. (2022); Wang et al. (2021b). Beyond these algorithmic advances, recent works and benchmarks have explicitly identified heterogeneous MARL as an underdeveloped but increasingly important setting Liu et al. (2025). However, existing policy optimization methods mainly address stable updates across different agents, while representation and communication methods often focus on inter-agent message aggregation. They do not directly answer how a centralized critic should weight heterogeneous state components when different observations contribute unevenly to coordination. Our evaluation therefore includes MAPPO, HAPPO, and HASAC to cover homogeneous cooperative policy optimization, heterogeneous sequential policy optimization, and maximum-entropy heterogeneous actor-critic learning.

Recent MARL research also places increasing emphasis on reliable and diverse evaluation. BenchMARL, JaxMARL, VMAS, and Melting Pot provide modern benchmark and implementation suites for reproducible MARL evaluation Bettini et al. (2024); Rutherford et al. (2024); Bettini et al. (2022); Agapiou et al. (2022), while newer efforts target continual adaptation, open-agent systems, and explicitly heterogeneous teams Tomilin et al. (2025); Patino et al. (2025). Our experiments use four complementary benchmark families: MAMuJoCo for decentralized continuous locomotion Schroeder de Witt et al. (2020), DexHands for high-dimensional dexterous manipulation Chen et al. (2022), MPE/PettingZoo for lightweight particle-world coordination Terry et al. (2021), and SMACv2 for discrete strategic micromanagement under diverse scenarios Ellis et al. (2023).

3 METHOD

3.1 LIMITATIONS OF JOINTLY OPTIMIZED STATE WEIGHTING

Existing state-weighting methods commonly optimize the weighting network jointly with the actor or critic. Given the per-sample policy loss and its predicted state weight, the weighted policy objective is defined as

$$\mathcal{L}_\pi^w = \frac{1}{B}\sum_{i=1}^{B} w_i \ell_i^\pi \tag{1}$$

Here, B denotes the batch size, i indexes the samples, w_i is the state weight predicted by the weighting network W_ψ, ℓ_i^π is the per-sample policy loss, and $\mathcal{L}_\pi^w$ is the resulting batch averaged weighted policy objective. Because w_i remains connected to the computation graph, minimizing $\mathcal{L}_\pi^w$ updates both the policy and the weighting network. In particular, the gradient received by W_ψ is scaled by ℓ_i^π, allowing it to reduce the objective by assigning smaller weights to samples with large policy losses.

This loss driven shortcut may further cause the predicted weights to concentrate near their lower bound, leading to degenerate weight allocation. The problem can be more evident in high-dimensional tasks, where learning difficulty varies substantially across states and informative samples are often sparsely distributed.

Algorithm 1 Overall update procedure of SAW-v2

Require: Minimum weight $w_{\min}$; constrained temperature $\bar{\tau}$; stability constant ϵ; update interval K

Ensure: Updated actor parameters θ and weight parameters ψ

```
Sample a mini-batch and obtain o_b, o'_b, and mask m_b
Compute the per-sample actor loss ℓ_{π,b}
Obtain Q_{1,b} and Q_{2,b} if twin critics are available
for each valid sample b do
    Predict the initial weight z_b ← W_ψ(o_b)
    Compute the bounded weight ŵ_b ← w_min + (1 − w_min)σ(z_b/τ̄)
end for
Compute μ_ŵ as the mean of ŵ_b over valid samples
for each valid sample b do
    Normalize the weight W_b ← ŵ_b/(μ_ŵ + ε)
    Detach W_b and use it to weight ℓ_{π,b}
end for
Average the weighted losses over valid samples to obtain L_π^w
Update θ using L_π^w
if k mod K = 0 then
    Update ψ using the dual-stream supervision in Algorithm 2
end if
return L_π^w, θ, ψ
```

To address this limitation, SAW-v2 decouples state weight learning from actor-critic optimization and trains the weighting network using explicit supervision signals derived from policy difficulty and value-estimation uncertainty.

3.2 Overview of SAW-v2

As summarized in Algorithm 1, SAW-v2 augments the actor side of a multi-agent actor-critic framework with a lightweight Weight Network. The weight network maps the observation to an initial weight, which is transformed into the bounded prediction by a temperature-controlled sigmoid with the lower bound. Masked batch-mean normalization (BMN) then converts it into the final weight W_b, preserving relative state importance while keeping the average weight close to one over valid samples. The final weight is detached and applied to the corresponding per-sample actor loss $\ell_{\pi,b}$, allowing it to control the contribution of each sample while ensuring that the resulting weighted actor objective updates only the actor parameters θ.

The Weight Network is optimized separately from the actor through a supervised update performed every K training steps. SAW-v2 constructs a dual-stream target from the detached actor-loss magnitude and, when twin critics are available, their detached value disagreement. These signals are fused to supervise the bounded prediction $\widehat{w}_b$, while optional temporal consistency regularization encourages stable predictions across consecutive observations. Consequently, actor–critic signals train the Weight Network parameters ψ, whereas the detached final weights guide actor optimization without creating a shared gradient path.

3.3 State-Weight Prediction

The additional learnable component in SAW-v2 is an adaptive weight network W_ψ. As shown in Fig. 2, the module first predicts a bounded state weight and then sends that prediction to two separate paths: a supervised path for training the weight network and a detached application path for reweighting the actor loss.

For clarity, b denotes a mini-batch sample and agent indices are omitted in the following discussion. The weight network receives the local observation o_b and produces a scalar initial weight z_b:

$$z_b = W_\psi(o_b) \tag{2}$$

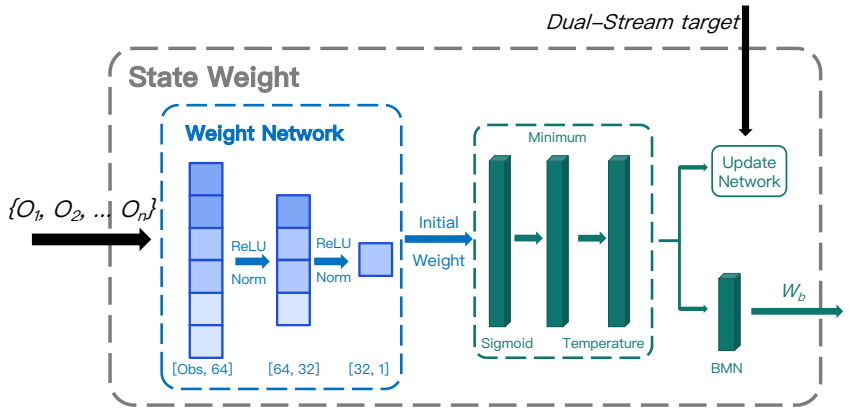


Figure 2: The weight network maps observations through two normalized ReLU hidden layers to an initial weight. A temperature-controlled sigmoid with a minimum-weight constraint bounds the prediction. The bounded weights supervise the weight network using the dual-stream target, while BMN produces the final weights W_b for reweighting the per-sample actor losses.

We use initial weight to emphasize that z_b is produced by the network. The actor loss and critic values are not network inputs, which are used later as detached supervision. This separation requires the network to learn a state-dependent importance function rather than directly copying the current loss or critic disagreement.

We implement W_ψ as a multilayer perceptron. Each hidden linear layer is followed by a ReLU activation and layer normalization, and the final linear head outputs the scalar initial weight without an activation. The network is orthogonally initialized. This compact design makes the predictor substantially smaller than the actor and critic networks while retaining enough capacity to model nonlinear state-importance patterns.

The initial weight cannot be used directly because it may be negative or large. SAW-v2 therefore applies a temperature controlled sigmoid and introduces a positive lower bound:

$$\widehat{w}_b = w_{\min} + (1 - w_{\min})\sigma\left(\frac{z_b}{\bar{\tau}}\right) \tag{3}$$

Here, $\bar{\tau}$ is the constrained temperature controlling the sharpness of the sigmoid function $\sigma(\cdot)$, $w_{\min}$ specifies the minimum allowable weight, and $\widehat{w}_b$ denotes the resulting bounded weight prediction. For numerical stability, the effective temperature is clipped by a small positive threshold. The operation is applied in the order of temperature scaling, sigmoid activation, and lower-bound mapping. It is therefore a bounded sigmoid transformation defined by SAW-v2 rather than a separate standard learning algorithm.

The bounded prediction is the branching point of the module. On one hand, $\widehat{w}_b$ is compared with the detached dual-stream target y_b, and the resulting weight objective updates W_ψ. On the other hand, $\widehat{w}_b$ is passed through BMN to produce the final weight W_b, which is subsequently detached and applied to the corresponding per-sample actor loss. Although the bounded predictions have a stable range, their batch mean can still vary over training. Such variation would change the overall magnitude of the actor gradient in addition to changing the relative contribution of samples. SAW-v2 removes this unwanted scale change through masked BMN:

$$\mu_{\widehat{w}} = \frac{\sum_j m_j \widehat{w}_j}{\max\left(\sum_j m_j, 1\right)} \tag{4}$$

$$W_b = \frac{\widehat{w}_b}{\mu_{\widehat{w}} + \epsilon} \tag{5}$$

Here, j indexes the samples in the batch, m_j is the validity mask, $\widehat{w}_j$ is the bounded weight prediction, $\mu_{\widehat{w}}$ is the mean weight over valid samples, ϵ is a small constant for numerical stability, and W_b

Algorithm 2 Dual-stream supervised update of SAW-v2

Require: Minimum weight $w_{\min}$ and learning rate η_ψ
Ensure: Updated weight-network parameters ψ
Compute the policy-difficulty target $y_\pi \leftarrow \mathcal{S}(|\mathrm{stopgrad}(\ell_\pi)|)$
if twin-critic estimates are available **then**
    Compute the critic-uncertainty target $y_Q \leftarrow \mathcal{S}(|\mathrm{stopgrad}(Q_1 - Q_2)|)$
    Combine the dual-stream targets $y \leftarrow \max(y_\pi, y_Q)$
else
    $y \leftarrow y_\pi$
end if
Predict the bounded weight $\widehat{w}$ from o
Compute $\mathcal{L}_w$ by fitting $\widehat{w}$ to y over valid entries
Add the enabled variance and temporal regularizers to $\mathcal{L}_w$
Update ψ using $\mathcal{L}_w$ with learning rate η_ψ
return ψ

is the final weight. The normalization preserves the relative ordering of the predicted weights while keeping their mean close to one over valid samples. Consequently, samples with above-average importance are emphasized, samples with below-average importance are attenuated, and the overall actor-loss scale remains stable.

During actor optimization, each normalized state weight W_b is detached and used to scale the corresponding per-sample actor loss. Consequently, the weights control the relative contributions of training samples without allowing the actor objective to update the Weight Network W_ψ. Conversely, the detached actor-loss and critic-disagreement signals supervise W_ψ without propagating gradients to the actor or critics. SAW-v2 therefore enables bidirectional information exchange while maintaining separate optimization paths for the actor parameters θ and the Weight Network parameters ψ, as summarized in Algorithm 1.

3.4 Dual-Stream Supervised State Adaptive Weighting

The detached final weights cannot train the Weight Network through the actor objective, so SAW-v2 constructs an independent supervised target from signals already available during actor-critic optimization. The two streams capture complementary properties of the training data: the policy stream measures optimization difficulty, while the critic stream measures uncertainty in value estimation. Both signals are detached before target construction, ensuring that the Weight Network update does not propagate gradients to the actor or critics. This progress is summarized as Algorithm 2 and the complete information flow is shown in Fig. 3. Both streams use the same masked scaling operation:

$$\mathcal{S}(x) = w_{\min} + (1 - w_{\min}) \frac{x}{\max(x) + \epsilon} \tag{6}$$

Here, x is a nonnegative signal, $\max(x)$ is evaluated over valid entries, ϵ prevents division by zero, and $\mathcal{S}$ maps the signal to the bounded target range. The policy stream applies this operation to the detached actor-loss magnitude:

$$y_\pi = \mathcal{S}(|\mathrm{stopgrad}(\ell_\pi)|) \tag{7}$$

Here, ℓ_π is the unreduced actor loss and y_π is the resulting policy-difficulty target. A larger loss magnitude produces a larger target, indicating that the current policy can still obtain a strong learning signal from that state. When twin critics are available, SAW-v2 additionally uses their absolute disagreement as a value-uncertainty signal:

$$y_Q = \mathcal{S}(|\mathrm{stopgrad}(Q_1 - Q_2)|) \tag{8}$$

Here, Q_1 and Q_2 are the two critic network outputs, and y_Q is the corresponding critic disagreement target. The method does not select one critic output over the other; instead, a large discrepancy between them indicates that the state has not yet been evaluated consistently. The two streams are combined by retaining the stronger target element-wise:

$$y = \max\left(y_\pi, y_Q\right) \tag{9}$$

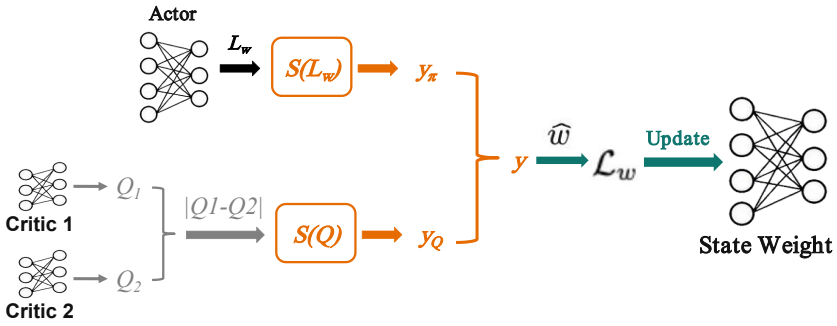


Figure 3: Dual-stream supervision for the SAW-v2 Weight Network. Actor network loss and critic disagreement are independently scaled and fused to form the target y, which supervises the bounded prediction $\widehat{w}$.

Here, y is the final supervision target. This choice method avoids the dilution that could occur if a strong signal from one stream were averaged with a weak signal from the other.

The fused target supervises the bounded prediction $\widehat{w}$ rather than the BMN output W. This keeps the prediction and its target on the same fixed scale and prevents BMN from affecting the regression objective. The main Weight Network loss is the mean squared error between $\widehat{w}$ and y over valid entries, so W_ψ learns to infer state importance from o alone. The actor loss and critic estimates provide supervision but are never concatenated with the observation or passed into the Weight Network.

3.5 Integration with MARL Algorithms

SAW-v2 is designed as a module that can be inserted into an existing MARL algorithm. The base algorithm only needs to expose its per-sample actor loss before batch reduction, after which SAW-v2 multiplies each valid loss $\ell_{\pi,b}$ by the detached final weight W_b and retains the original reduction. The validity mask of the base algorithm is reused during target scaling, BMN, and actor-loss reduction so that padded transitions and inactive agents are ignored. SAW-v2 follows the parameter sharing convention of the underlying MARL method and requires only a lightweight Weight Network forward pass together with one supervised update every K steps. Because the Weight Network is used only for loss reweighting during training, SAW-v2 introduces no additional environment interaction or test-time action-selection cost.

4 Experiments

4.1 Experimental Setup

We evaluate SAW-v2 on 18 cooperative tasks drawn from four complementary benchmark. MA-MuJoCo evaluates heterogeneous continuous control under different actuator partitions, including Ant-v2-2x4, HalfCheetah-v2-2x3, Hopper-v2-3x1, Humanoid-v2-17x1, HumanoidStandup-v2-17x1, and Walker2d. DexHands evaluates high-dimensional bimanual manipulation, including ShadowHandCatchAbreast, ShadowHandCatchOver2Underarm, ShadowHandDoorCloseInward, ShadowHandDoorOpenInward, ShadowHandDoorOpenOutward, and ShadouHandLiftUnderarm. PettingZoo MPE evaluates lightweight continuous coordination and communication, including Reference-v2, Speaker-Listener-v3, and Spread-v2. SMACv2 evaluates cooperative micromanagement with discrete actions, including Protoss_5vs5, Terran_5vs5, and Zerg_5vs5.

We use MAPPO, HAPPO, and HASAC as the three baseline algorithms. For each baseline algorithm, we compare the original implementation with the corresponding SAW-v2 implementation. All SAW-v2 variants use a two layer weight network with hidden sizes $[64, 32]$ (with ReLU and normalization), and minimum weight 0.1. Batch mean weight normalization is enabled, the weight network is updated at every optimization step, and its output is detached before weighting the actor loss. Experiment results contain 324 runs (18 tasks, 6 methods, and 3 seeds). More detailed experimental parameters are provided in the appendix.

Table 1: Performance comparison of SAW-v2 on multi-agent reinforcement learning benchmarks. Results are reported as the average episodic rewards over the final 10% of training steps. The percentages in parentheses denote relative changes over the corresponding baselines, and bold values indicate the better result within each baseline–SAW-v2 pair.

	MAPPO	+SAW-v2(ours)	HAPPO	+SAW-v2(ours)	HASAC	+SAW-v2(ours)
Ant	5750.36	**5901.94** (+2.6%)	5588.49	**5636.97** (+0.9%)	5746.51	**6295.07** (+9.6%)
HalfCheetah	6404.04	**7212.86** (+12.6%)	6316.93	**6567.48** (+4.0%)	6426.52	**6635.64** (+3.3%)
Hopper	3484.65	**3596.13** (+3.2%)	3244.92	**3652.09** (+12.5%)	2273.41	**2990.57** (+31.5%)
Humanoid	564.32	**632.61** (+12.1%)	5780.41	**6559.82** (+13.5%)	3423.36	**4428.46** (+29.4%)
HumanoidStandUp	90266.01	**113099.58** (+25.3%)	**161143.80**	158782.39 (-1.5%)	141397.87	**160958.64** (+13.8%)
Walker2d	3942.25	**5302.21** (+34.5%)	4041.29	**4345.18** (+7.5%)	5443.96	**5630.20** (+3.4%)
CatchAbreast	**0.553**	0.527 (-4.7%)	0.526	**0.852** (+62.0%)	23.003	**35.842** (+55.8%)
CatchOver2Underarm	15.39	**18.48** (+20.1%)	17.95	**19.40** (+8.1%)	19.53	**24.78** (+26.9%)
DoorCloseInward	205.72	**224.88** (+9.3%)	180.63	**229.94** (+27.3%)	439.26	**449.22** (+2.3%)
DoorOpenInward	**337.97**	317.87 (-5.9%)	263.67	**299.46** (+13.6%)	428.78	**488.65** (+14.0%)
DoorOpenOutward	231.45	**241.40** (+4.3%)	256.15	**273.06** (+6.6%)	497.96	**515.03** (+3.4%)
LiftUnderarm	151.69	**210.29** (+38.6%)	166.88	**281.06** (+68.4%)	311.58	**350.01** (+12.3%)
Reference_v2	-13.49	**-12.62** (+6.5%)	-18.40	**-18.28** (+0.7%)	-11.03	**-9.84** (+10.8%)
Speaker_Listener_v3	-13.81	**-8.61** (+37.6%)	-8.94	**-8.63** (+3.6%)	-9.29	**-9.08** (+2.3%)
Spread_v2	-76.28	**-73.22** (+4.0%)	**-62.31**	-68.79 (-10.4%)	-58.85	**-57.49** (+2.3%)
Protoss_5_vs_5	16.04	**16.33** (+1.8%)	17.22	**17.23** (+0.1%)	13.01	**13.49** (+3.8%)
Terran_5_vs_5	11.89	**13.55** (+13.9%)	15.03	**16.21** (+7.9%)	**12.03**	11.57 (-3.8%)
Zerg_5_vs_5	**12.79**	12.34 (-3.5%)	13.71	**14.17** (+3.4%)	10.67	**11.17** (+4.7%)

4.2 Evaluation Index

For each group of experiments, we analyzed the results on the same total number of steps. In order to avoid being influenced by the randomness of the data, we calculated the average reward of the last 10% of the entire training process as the evaluation metric, and the data for each group of experiments came from three different random seeds. In addition, we also calculate the standard error for 3 different random seeds.

4.3 Main Results

Table 1 provides the corresponding quantitative comparison of all the 54 groups. Each entry reports the average episode reward over the final 10% of training, and the percentage in parentheses measures the relative change from the corresponding baseline. It shows that SAW-v2 improves 48 of the 54 groups of the comparison, including 15/18 groups for MAPPO, 16/18 for HAPPO, and 17/18 for HASAC. The average relative gains for MAPPO, HAPPO, and HASAC are 11.8%, 12.7%, and 12.5%, respectively, which indicates that the effectiveness of SAW-v2 is not tied to a specific policy or optimization paradigm.

Figure 4 presents the learning curves of MAPPO, HAPPO, and HASAC with and without SAW-v2 across all 18 tasks. The dashed lines denote the original algorithms, while the solid lines denote their SAW-v2-enhanced variants, and the shaded regions represent the mean standard error. The curves show that SAW-v2 improves not only the converged performance but also the learning dynamics in many environments. For example, MAPPO+SAW-v2 learns faster and reaches higher performance than MAPPO on HalfCheetah, Walker2d, and LiftUnderarm, while HAPPO+SAW-v2 exhibits clear and sustained advantages on Humanoid, CatchAbreast, DoorCloseInward, and LiftUnderarm. Similarly, HASAC+SAW-v2 achieves substantially higher performance on Ant, Humanoid, CatchAbreast, and CatchOver2Underarm.

In these tasks, the SAW-v2-enhanced curves separate from their baselines during the intermediate stages of training and maintain their advantages until convergence, indicating that the improvements are not caused by isolated fluctuations at the end of training. The advantages are particularly evident on high-dimensional, continuous-control, and dexterous-manipulation tasks, suggesting that SAW-v2 is effective in environments involving complex coordination and credit assignment. In terms of

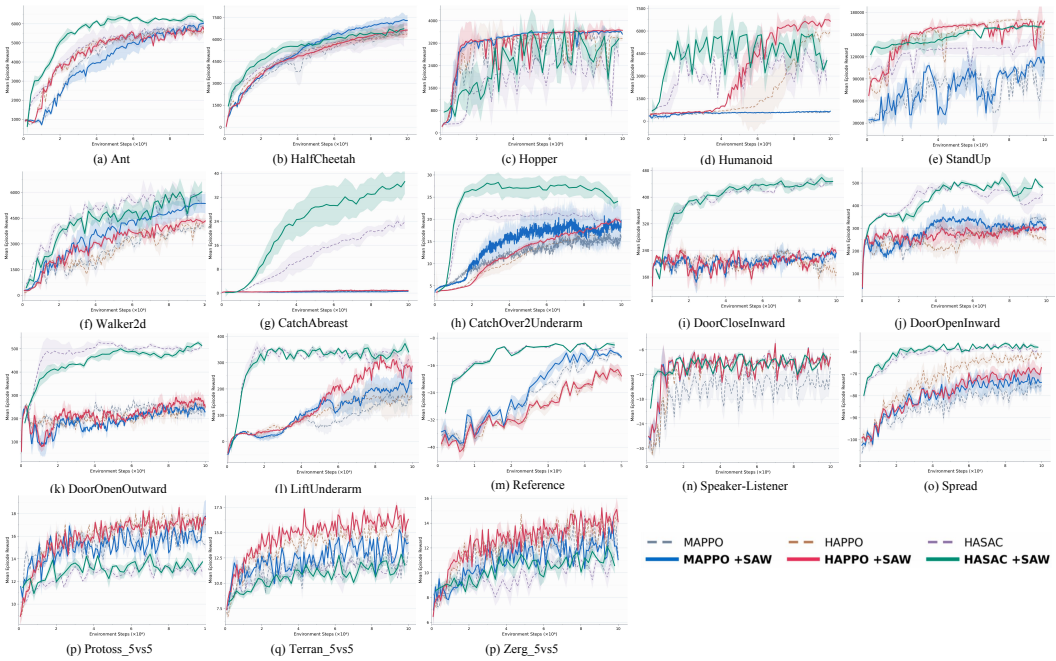


Figure 4: Learning curves across 18 MARL tasks. Dashed gray, brown, and purple lines represent the baseline MAPPO, HAPPO, and HASAC algorithms, respectively, whereas solid blue, red, and green lines represent MAPPO+SAW-v2, HAPPO+SAW-v2, and HASAC+SAW-v2, respectively. Each curve shows the mean episode reward, and the corresponding shaded region denotes the mean standard error.

simple particle motion and discrete control-based game combat tasks, the improvement is usually moderate. However, in most cases, the SAW-v2 variant can still achieve higher training curves.

Above all, the agreement between the final-stage statistics in Table 1 and the complete learning trajectories in Figure 4 provides complementary evidence that the observed gains are both consistent and sustained throughout training. The improvements across different baseline algorithms and diverse benchmarks demonstrate that SAW-v2 is a broadly applicable enhancement for MARL, particularly in high-dimensional and heterogeneous coordination tasks.

5 Conclusion

In this work, we introduced SAW-v2, a lightweight state adaptive weighting framework for MARL. It addresses the degeneracy of jointly optimized weighting mechanisms by decoupling weight learning from policy optimization. Specifically, it learns observation-dependent importance weights through detached dual-stream supervision derived from policy difficulty and critic uncertainty. Experiments across 18 tasks from MAMuJoCo, DexHands, MPE, and SMACv2 demonstrate that SAW-v2 provides consistent benefits for MAPPO, HAPPO, and HASAC. It improves 48 of the 54 group comparisons, corresponding to a win rate of 88.9% and an average relative improvement of 12.3%. The most substantial gains are observed in high-dimensional, continuous-control, and dexterous-manipulation tasks. These results support SAW-v2 as a broadly applicable plug-and-play enhancement for MARL.

Nevertheless, localized performance degradation suggests that baseline loss and critic disagreement do not always accurately reflect the learning value of a state. Future work will investigate adaptive and uncertainty calibrated fusion of the two supervision signals and evaluate this in larger scale multi-agent environments.

AI USE STATEMENT

AI tools were used to assist with language editing and LaTeX formatting during the preparation of this manuscript. They were not used to generate experimental results or to replace the authors' scientific reasoning and analysis. All AI-assisted content was carefully reviewed and verified by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide the complete formulation and update procedure of SAW-v2 in Section 3, including pseudocode for both the overall actor update and the dual-stream supervised weight update. Detailed experimental settings, hyperparameters, network architectures, and additional results are provided in the appendix. All reported results are obtained from the experimental protocol described in Section 4. Code and configuration files will be provided as anonymous supplementary material to facilitate reproduction of our results.

REFERENCES

John P Agapiou, Alexander Sasha Vezhnevets, Edgar A Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, et al. Melting pot 2.0. *arXiv preprint arXiv:2211.13746*, 2022.

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

Yunpeng Bai, Chen Gong, Bin Zhang, Guoliang Fan, Xinwen Hou, and Y Liu. Value function factorisation with hypergraph convolution for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2112.06771*, 2021.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. Vmas: A vectorized multi-agent simulator for collective robot learning. In *International Symposium on Distributed Autonomous Robotic Systems*, pp. 42–56. Springer, 2022.

Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning. *arXiv preprint arXiv:2301.07137*, 2023.

Matteo Bettini, Amanda Prorok, and Vincent Moens. BenchMARL: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024.

Wenjing Chen, Chengyuan Qian, Shuo Xing, Yi Zhou, and Victoria Crawford. Multi-agent reinforcement learning with submodular reward. *arXiv preprint arXiv:2603.06810*, 2026.

Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

David Eckel and Henri Meeß. Hierarchical lead critic based multi-agent reinforcement learning. *arXiv preprint arXiv:2602.21680*, 2026.

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

Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2206.07505*, 2022.

Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, and Yaodong Yang. Multi-agent constrained policy optimisation. *arXiv preprint arXiv:2110.02793*, 2021.

Cong Guan, Feng Chen, Lei Yuan, Chenghe Wang, Hao Yin, Zongzhang Zhang, and Yang Yu. Efficient multi-agent communication via self-supervised information aggregation. *Advances in Neural Information Processing Systems*, 35:1020–1033, 2022.

Mohamad A Hady, Siyi Hu, Mahardhika Pratama, Zehong Cao, and Ryszard Kowalczyk. Multi-agent reinforcement learning for resources allocation optimization: a survey. *Artificial Intelligence Review*, 58(11):354, 2025.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 2961–2970, 2019.

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

Chuming Li, Jie Liu, Yinmin Zhang, Yuhong Wei, Yazhe Niu, Yaodong Yang, Yu Liu, and Wanli Ouyang. Ace: Cooperative multi-agent q-learning with bidirectional action-dependency. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 8536–8544, 2023.

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.

Qi Liu, Xiaopeng Zhang, Mingshan Tan, Shuaikang Ma, Jinliang Ding, and Yanjie Li. MASH: Cooperative-heterogeneous multi-agent reinforcement learning for single humanoid robot locomotion. *arXiv preprint arXiv:2508.10423*, 2025.

Chris Lu, Timon Willi, Christian Schroeder de Witt, et al. Model-free opponent shaping. In *International Conference on Machine Learning*, volume 162, 2022.

Linghui Meng, Muning Wen, Chenyang Le, Xiyun Li, Dengpeng Xing, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, Yaodong Yang, et al. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 20(2):233–248, 2023.

Siddharth Nayak, Kenneth Choi, Wenqi Ding, Sydney Dolan, Karthik Gopalakrishnan, and Hamsa Balakrishnan. Scalable multi-agent reinforcement learning through intelligent information aggregation. In *International conference on machine learning*, pp. 25817–25833. PMLR, 2023.

Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International conference on machine learning*, pp. 17221–17237. PMLR, 2022.

Ceferino Patino, Tyler J. Billings, Alireza Saleh Abadi, Daniel Redder, Adam Eck, Prashant Doshi, and Leen-Kiat Soh. Inaugural MOASEI competition at AAMAS'2025: A technical report. *arXiv preprint arXiv:2507.05469*, 2025.

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

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian S de Witt, et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. *Advances in Neural Information Processing Systems*, 37: 50925–50951, 2024.

Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709*, 2020.

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

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

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

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

Justin K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15032–15043, 2021.

Tristan Tomilin, Luka van den Boogaard, Samuel Garcin, Bram Grooten, Meng Fang, and Mykola Pechenizkiy. MEAL: A benchmark for continual multi-agent reinforcement learning. *arXiv preprint arXiv:2506.14990*, 2025.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *9th International Conference on Learning Representations, ICLR 2021*, 2021a.

Jianhong Wang, Yuan Zhang, Yunjie Gu, and Tae-Kyun Kim. Shaq: Incorporating shapley value theory into multi-agent q-learning. *Advances in Neural Information Processing Systems*, 35: 5941–5954, 2022.

Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. *Advances in Neural Information Processing Systems*, 36:52413–52429, 2023.

Yichen Wang, Shuai Zheng, Ze Yang, and Xin Zhou. A novel multi-agent reinforcement learning approach based on state adaptive weighting and exploration path sampling. *Applied Soft Computing*, pp. 114919, 2026.

Yuanfei Wang, Fangwei Zhong, Jing Xu, and Yizhou Wang. Tom2c: Target-oriented multi-agent communication and cooperation with theory of mind. *arXiv preprint arXiv:2111.09189*, 2021b.

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

Donghao Ying, Yunkai Zhang, Yuhao Ding, Alec Koppel, and Javad Lavaei. Scalable primal-dual actor-critic method for safe multi-agent rl with general utilities. *Advances in Neural Information Processing Systems*, 36:36524–36539, 2023.

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.

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.

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

Yihe Zhou, Shunyu Liu, Yunpeng Qing, Kaixuan Chen, Tongya Zheng, Jie Song, and Mingli Song. Is centralized training with decentralized execution framework centralized enough for marl? *arXiv preprint arXiv:2305.17352*, 2023a.

Z Zhou, G Liu, and Y Tang. Multi-agent reinforcement learning: Methods, applications, visionary prospects, and challenges. *arXiv preprint arXiv:2305.10091*, 2023b.

APPENDIX

DETAILED EXPERIMENTAL SETTINGS

We evaluate SAW-v2 on 18 cooperative tasks from four benchmark families: MAMuJoCo, DexHands, the continuous-action MPE environments provided by PettingZoo, and SMACv2. MAMuJoCo evaluates heterogeneous continuous control under different actuator partitions, DexHands evaluates high-dimensional bimanual manipulation, MPE evaluates continuous coordination and communication, and SMACv2 evaluates cooperative micromanagement with discrete actions. The complete task set is summarized in Table 2.

Table 2: Evaluated benchmark tasks. Each task contains three backbone algorithms and their SAW-v2 variants, with three random seeds for every method.

Benchmark	Task

6*MAMuJoCo	`Ant-v2-2x4`
	`HalfCheetah-v2-2x3`
	`Hopper-v2-3x1`
	`Humanoid-v2-17x1`
	`HumanoidStandup-v2-17x1`
	`Walker2d-v2-2x3`

6*DexHands	`ShadowHandCatchAbreast`
	`ShadowHandCatchOver2Underarm`
	`ShadowHandDoorCloseInward`
	`ShadowHandDoorOpenInward`
	`ShadowHandDoorOpenOutward`
	`ShadowHandLiftUnderarm`

3*MPE	`simple_reference_v2-continuous`
	`simple_speaker_listener_v3-con`
	`simple_spread_v2-continuous`

3*SMACv2	`protoss_5_vs_5`
	`terran_5_vs_5`
	`zerg_5_vs_5`

All MAMuJoCo experiments set `agent_obsk=0`, such that each agent observes only the states of its own joints and body components rather than those controlled by neighboring agents, and an episode limit of 1,000 environment steps. The DexHands episode limits are task dependent: 75 steps for `ShadowHandCatchOver2Underarm`, 150 steps for `ShadowHandCatchAbreast`, 250 steps for the three door-manipulation tasks, and 500 steps for `ShadowHandLiftUnderarm`. All MPE experiments use continuous actions. The exact environment arguments that differ across individual tasks are reported together with the corresponding task results in the following sections.

BASELINE HYPERPARAMETERS

Table 3 summarizes the ranges of backbone hyperparameters used across the 18 tasks. A set of values indicates task-dependent configurations inherited from the corresponding backbone. The exact choice for each experiment is specified in its task-level subsection.

SAW-V2 HYPERPARAMETERS

SAW-v2 is applied only during training. The Weight Network receives the local observation and predicts one scalar importance weight for each valid training sample. Its normalized output is detached before multiplying the corresponding per-sample actor loss; consequently, the actor objective does not update the Weight Network directly. Batch-mean normalization is enabled in all experiments, and the Weight Network is updated at every optimization step.

The common SAW-v2 configuration is listed in Table 4. It is used by all MAPPO+SAW-v2 and HAPPO+SAW-v2 experiments and by 14 of the 18 HASAC+SAW-v2 experiments.

Table 3: Backbone hyperparameters used in the experiments. MAPPO and HAPPO use on-policy PPO-style optimization, whereas HASAC uses off-policy maximum-entropy actor–critic optimization.

Hyperparameter	MAPPO / HAPPO	HASAC
Advantage estimation	GAE with $\lambda = 0.95$	Not applicable
Discount factor γ	0.99; 0.95 for a subset of DexHands HAPPO tasks	$\{0.95, 0.99\}$
Initialization and activation	Orthogonal initialization; ReLU	Orthogonal initialization; ReLU
Value normalization	Enabled	Task dependent
Hidden-layer sizes	$[128, 128]$, $[128, 128, 128]$; $[64]$ for SMACv2 HAPPO; and $[256, 256, 256]$ for the corresponding DexHands HAPPO tasks	$[256, 256]$
Actor learning rate	$\{5\times10^{-4}, 10^{-3}\}$	$\{5\times10^{-4}, 10^{-3}\}$
Critic learning rate	$\{5\times10^{-4}, 10^{-3}\}$	$\{3\times10^{-4}, 5\times10^{-4}, 10^{-3}\}$
PPO clipping coefficient	$\{0.05, 0.1, 0.2\}$	Not applicable
Actor / critic epochs	$\{5, 10, 15\}$ / $\{5, 10, 15\}$	Not applicable
Entropy setting	Entropy coefficient $\{0, 0.01\}$	Fixed or automatically adjusted temperature, task dependent
Replay-buffer capacity	Not applicable	10^6 transitions
Batch size	On-policy mini-batches	$\{1000, 2200\}$
n-step return	Not applicable	$\{5, 10, 20\}$
Polyak coefficient	Not applicable	0.005

Table 4: Common SAW-v2 hyperparameters.

Hyperparameter	Value
Weight Network hidden sizes	$[64, 32]$
Hidden activation	ReLU
Hidden normalization	Layer normalization
Weight Network learning rate	10^{-4}
Minimum weight $w_{\min}$	0.1
Temperature $\bar{\tau}$	1.0
Update interval K	1
Batch-mean normalization	Enabled
Detach before actor weighting	Enabled

Training and Evaluation Protocol

Each method is trained with random seeds 1, 2, and 3. All recorded configurations enable CUDA, deterministic CUDA execution, and four PyTorch CPU threads. Most experiments use 20 parallel rollout threads, while the selected high-throughput DexHands HAPPO configurations use 256 rollout threads. Separate evaluation is enabled for MAMuJoCo, MPE, and SMACv2, using 20 or 40 evaluation episodes depending on the task and backbone. DexHands does not use a separate evaluator; its reported score is therefore the rollout return recorded during training. For the other benchmark families, we use the logged average evaluation return. For SMACv2, average episode reward is used as the primary metric rather than the separately recorded win rate.

Some raw runs are longer than the common analysis horizons. For example, several HASAC configurations contain 2×10^7 steps and several DexHands HAPPO configurations contain 5×10^7 steps. To provide a paired comparison under the same environment-interaction budget, almost all methods are truncated to 10^7 environment steps.

Aggregation and Reporting

For each task and method, learning curves are constructed from the reward values recorded at environment steps shared by all three seeds. We do not interpolate between logging steps. At step t, the

plotted curve is the arithmetic mean:

$$\bar{R}(t) = \frac{1}{3}\sum_{i=1}^{3} R_i(t),$$

The shaded region represents one standard error:

$$\mathrm{SE}(t) = \frac{1}{\sqrt{3}}\sqrt{\frac{1}{2}\sum_{i=1}^{3}\big(R_i(t) - \bar{R}(t)\big)^2}.$$

The scalar performance of each seed is obtained by averaging all logged rewards within the final 10% of the common analysis horizon. The reported score is the mean of the resulting three seed-level values. Relative changes are calculated against the corresponding backbone using

$$\Delta(\%) = \frac{J_{\text{SAW-v2}} - J_{\text{base}}}{|J_{\text{base}}|} \times 100\%,$$

where the absolute baseline value in the denominator preserves the correct direction of improvement for tasks whose rewards are negative.

Detailed Results

This section presents the task-level learning curves and final performance reported in the main paper. For every task, the scalar comparison refers to the average episode reward over the final 10% of the common training horizon, while the curve summarizes the mean and standard error over three random seeds.

MAMuJoCo

MAMuJoCo decomposes a single continuous-control system into multiple agents that control disjoint actuator groups. All six configurations use `agent_obsk=0`, so each agent observes only its own joints and bodies even though the team reward depends on the motion of the complete system. Across the six MAMuJoCo tasks, SAW-v2 improves 17 of the 18 backbone comparisons and achieves an average relative gain of 12.1%.

Ant-v2-2x4

The Ant task divides the quadruped's eight actuators between two agents, each of which controls four motors and must coordinate with the other agent to generate stable forward locomotion. The agents receive a shared reward determined by the forward motion and stability of the complete quadruped, while their local observations contain only the states associated with their own actuator groups.

Figure 5 shows that all three SAW-v2 variants finish above their corresponding baselines, with the clearest separation appearing in the HASAC pair. Over the final 10% of training, MAPPO increases from 5750.36 to 5901.94, HAPPO increases from 5588.49 to 5636.97, and HASAC increases from 5746.51 to 6295.07. These changes correspond to relative improvements of 2.6%, 0.9%, and 9.6%, respectively.

HalfCheetah-v2-2x3

HalfCheetah partitions the front and rear actuator groups between two agents, requiring them to synchronize propulsion and body posture to maintain fast forward motion. Each agent controls three actuators, and the shared return reflects the forward velocity and control cost of the complete body.

Figure 6 shows that MAPPO+SAW-v2 separates clearly from MAPPO during the later part of training, while the HAPPO and HASAC pairs have smaller final gaps. The final scores change from 6404.04 to 7212.86 for MAPPO, from 6316.93 to 6567.48 for HAPPO, and from 6426.52 to 6635.64 for HASAC. The corresponding relative gains are 12.6%, 4.0%, and 3.3%.

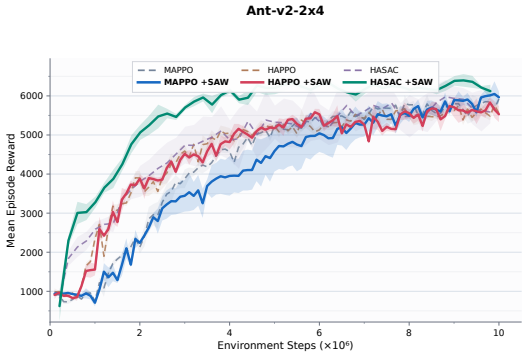


Figure 5: Learning curves of the six methods on `Ant-v2-2x4`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

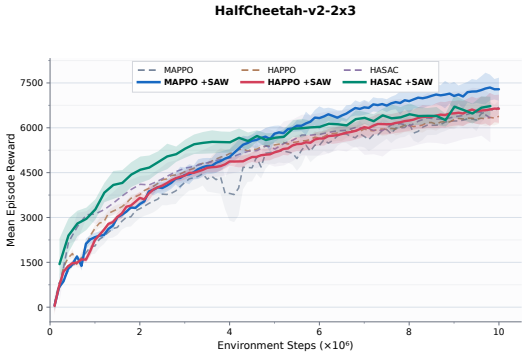


Figure 6: Learning curves of the six methods on `HalfCheetah-v2-2x3`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Hopper-v2-3x1

Hopper assigns one actuator to each of three agents, so forward movement depends on a tightly ordered interaction among leg extension, body posture, and landing balance. The shared task objective rewards forward hopping while penalizing unstable motion and control effort.

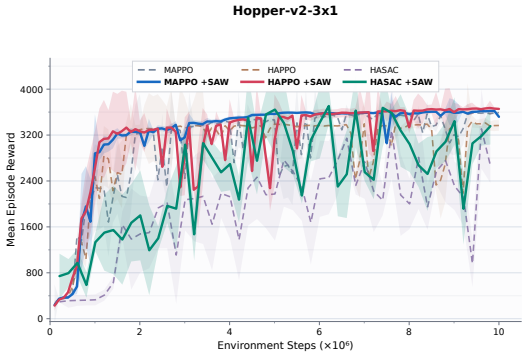


Figure 7: Learning curves of the six methods on `Hopper-v2-3x1`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 7 shows positive final differences for all three backbones, with the largest gap between HASAC and HASAC+SAW-v2. The final average rewards increase from 3484.65 to 3596.13 for MAPPO, from 3244.92 to 3652.09 for HAPPO, and from 2273.41 to 2990.57 for HASAC. These changes yield relative improvements of 3.2%, 12.5%, and 31.5%.

Humanoid-v2-17x1

Humanoid assigns each of the 17 actuators to a separate agent, creating the most fragmented control structure in the MAMuJoCo group. The agents must jointly maintain balance and produce coordinated whole-body locomotion despite receiving only their local actuator-related observations.

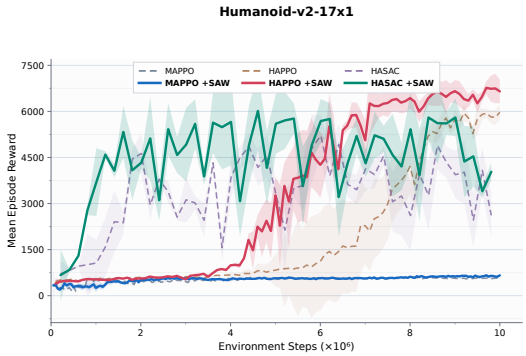


Figure 8: Learning curves of the six methods on `Humanoid-v2-17x1`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 8 shows that the SAW-v2 variants of all three algorithms achieve higher final performance than their baselines. The final score increases from 564.32 to 632.61 for MAPPO, from 5780.41 to 6559.82 for HAPPO, and from 3423.36 to 4428.46 for HASAC. The relative improvements are 12.1%, 13.5%, and 29.4%, respectively.

HumanoidStandup-v2-17x1

HumanoidStandup uses the same 17-agent actuator decomposition but requires the system to rise from the ground and maintain an elevated posture rather than sustain a periodic gait. Performance depends on the coordinated sequence of ground contact, body lifting, and posture stabilization executed by the independently controlled actuators.

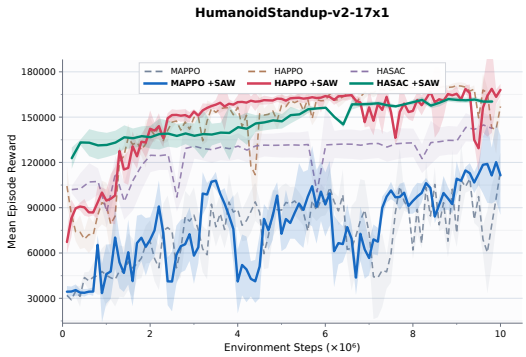


Figure 9: Learning curves of the six methods on `HumanoidStandup-v2-17x1`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 9 shows clear final advantages for MAPPO+SAW-v2 and HASAC+SAW-v2, while the two HAPPO curves remain close. MAPPO increases from 90266.01 to 113099.58 and HASAC increases from 141397.87 to 160958.64, corresponding to gains of 25.3% and 13.8%. HAPPO changes from 161143.80 to 158782.39, which corresponds to a decrease of 1.5%.

Walker2d-v2-2x3

Walker2d divides six leg actuators between two agents, which must coordinate alternating support and swing phases while keeping the torso upright. Each agent controls three motors, and both agents share the reward for forward movement and stable bipedal locomotion.

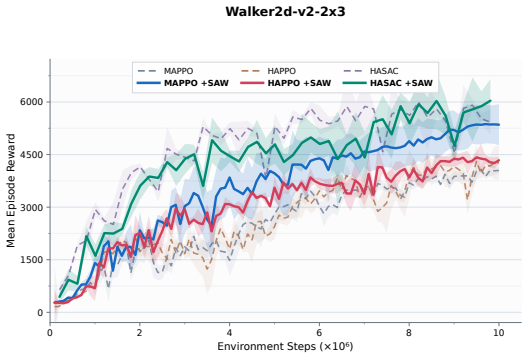


Figure 10: Learning curves of the six methods on `Walker2d-v2-2x3`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 10 shows the largest separation in the MAPPO pair, together with smaller positive differences for HAPPO and HASAC. The final rewards increase from 3942.25 to 5302.21 for MAPPO, from 4041.29 to 4345.18 for HAPPO, and from 5443.96 to 5630.20 for HASAC. The corresponding relative improvements are 34.5%, 7.5%, and 3.4%.

DEXHANDS

DexHands contains high-dimensional bimanual manipulation tasks with complex object contacts, heterogeneous hand roles, and long chains of coordinated actions. Across the six DexHands tasks, SAW-v2 improves 16 of the 18 backbone comparisons and achieves an average relative gain of 20.1%.

ShadowHandCatchAbreast

CatchAbreast arranges two movable Shadow Hands side by side and requires them to coordinate the catching motion under coupled object dynamics. The hands must adjust their positions and orientations according to the object trajectory while completing the manipulation within an episode limit of 150 steps.

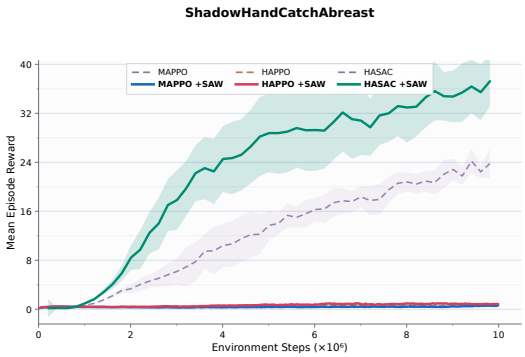


Figure 11: Learning curves of the six methods on `ShadowHandCatchAbreast`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 11 shows substantial upward separation for the HAPPO and HASAC variants, whereas MAPPO+SAW-v2 finishes slightly below MAPPO. MAPPO changes from 0.553 to 0.527, corresponding to a decrease of 4.7%. HAPPO increases from 0.526 to 0.852 and HASAC increases from 23.003 to 35.842, corresponding to improvements of 62.0% and 55.8%.

ShadowHandCatchOver2Underarm

CatchOver2Underarm requires the vertically oriented hand to throw an object to the palm-up hand, so the two agents perform different but temporally dependent roles. The task covers the complete transfer process from object release and flight to alignment and reception, with an episode limit of 75 steps.

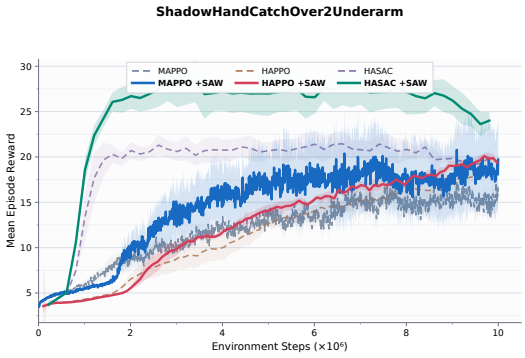


Figure 12: Learning curves of the six methods on `ShadowHandCatchOver2Underarm`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 12 shows that all three SAW-v2-enhanced algorithms finish above their original versions. The final scores increase from 15.39 to 18.48 for MAPPO, from 17.95 to 19.40 for HAPPO, and from 19.53 to 24.78 for HASAC. These changes correspond to gains of 20.1%, 8.1%, and 26.9%, respectively.

ShadowHandDoorCloseInward

DoorCloseInward begins with an inward-open door and requires coordinated hand contacts to move it back to the closed configuration. The two hands must manipulate the door and handle under contact constraints, and each episode contains at most 250 steps.

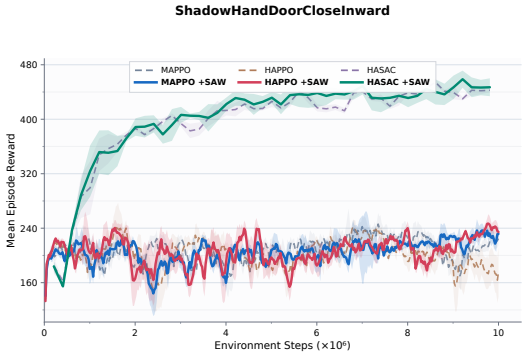


Figure 13: Learning curves of the six methods on `ShadowHandDoorCloseInward`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 13 shows positive final differences for all three algorithm pairs, with the largest relative change obtained by HAPPO. The final reward increases from 205.72 to 224.88 for MAPPO, from 180.63 to 229.94 for HAPPO, and from 439.26 to 449.22 for HASAC. The corresponding improvements are 9.3%, 27.3%, and 2.3%.

ShadowHandDoorOpenInward

DoorOpenInward starts from a closed door and requires the hands to pull it inward while maintaining a useful grasp and generating torque in the permitted direction. The episode limit is 250 steps, during which the hands must establish contact and increase the inward opening angle.

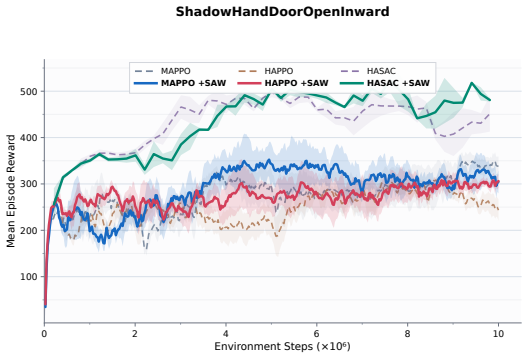


Figure 14: Learning curves of the six methods on `ShadowHandDoorOpenInward`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 14 shows higher final curves for HAPPO+SAW-v2 and HASAC+SAW-v2, while MAPPO+SAW-v2 finishes below MAPPO. MAPPO changes from 337.97 to 317.87, corresponding to a decrease of 5.9%. HAPPO increases from 263.67 to 299.46 and HASAC increases from 428.78 to 488.65, corresponding to improvements of 13.6% and 14.0%.

ShadowHandDoorOpenOutward

DoorOpenOutward requires the closed door to be pushed outward, changing the contact geometry and force direction relative to the inward-opening task. The hands must generate outward motion while maintaining effective contact with the door throughout the 250-step episode.

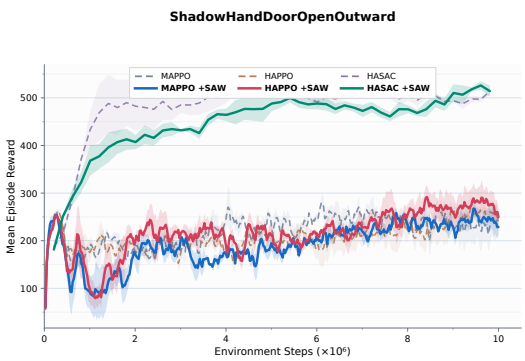


Figure 15: Learning curves of the six methods on `ShadowHandDoorOpenOutward`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 15 shows moderate positive differences for all three SAW-v2 variants. The final scores increase from 231.45 to 241.40 for MAPPO, from 256.15 to 273.06 for HAPPO, and from 497.96 to 515.03 for HASAC. The corresponding relative gains are 4.3%, 6.6%, and 3.4%.

ShadowHandLiftUnderarm

LiftUnderarm requires the two hands to grasp the handles of a pot and lift it to a target position while maintaining balanced support. The task includes the approach, grasping, load-sharing, and lifting stages and uses an episode limit of 500 steps.

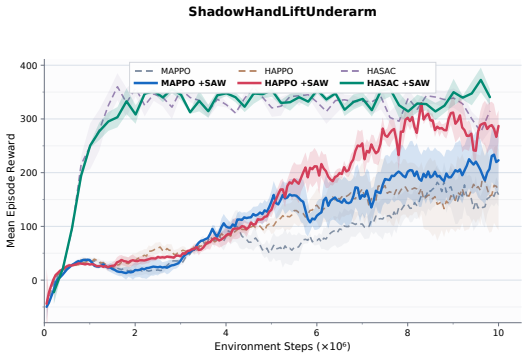


Figure 16: Learning curves of the six methods on `ShadowHandLiftUnderarm`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 16 shows clear and sustained separation for MAPPO+SAW-v2 and HAPPO+SAW-v2, together with a smaller positive difference for HASAC+SAW-v2. The final rewards increase from 151.69 to 210.29 for MAPPO, from 166.88 to 281.06 for HAPPO, and from 311.58 to 350.01 for HASAC. These changes correspond to improvements of 38.6%, 68.4%, and 12.3%.

MPE

The three continuous-action MPE tasks isolate lightweight coordination and communication problems in a particle world. Across the three MPE tasks, SAW-v2 improves eight of the nine backbone comparisons and achieves an average relative gain of 6.4%.

simple_reference_v2-continuous

Simple Reference contains two cooperative agents that must communicate their respective target landmarks while moving toward the landmark specified by the other agent. Both agents can move and communicate, and the experiments use the continuous versions of their physical actions.

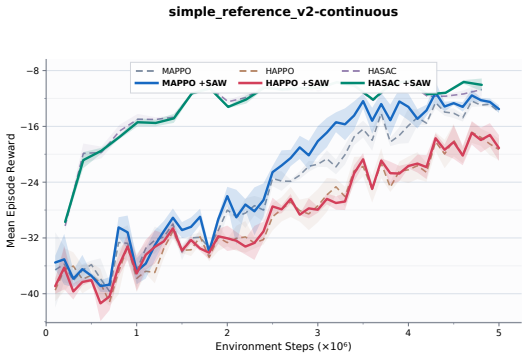


Figure 17: Learning curves of the six methods on `simple_reference_v2-continuous`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 17 shows that all three SAW-v2 variants achieve less negative final returns than their baselines. The final scores change from -13.49 to -12.62 for MAPPO, from -18.40 to -18.28 for HAPPO, and from -11.03 to -9.84 for HASAC. The corresponding relative improvements are 6.5%, 0.7%, and 10.8%.

simple_speaker_listener_v3-continuous

Simple Speaker Listener assigns asymmetric roles: the immobile speaker observes the target and communicates it, while the listener must decode the message and navigate to the correct landmark. Only the listener performs physical movement, making successful navigation dependent on the communication sent by the speaker.

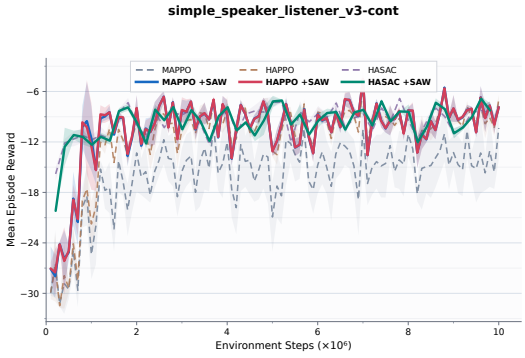


Figure 18: Learning curves of the six methods on `simple_speaker_listener_v3-continuous`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 18 shows the largest separation in the MAPPO pair and smaller positive final differences in the HAPPO and HASAC pairs. The final returns change from -13.81 to -8.61 for MAPPO, from -8.94 to -8.63 for HAPPO, and from -9.29 to -9.08 for HASAC. These changes correspond to improvements of 37.6%, 3.6%, and 2.3%.

simple_spread_v2-continuous

Simple Spread requires three homogeneous agents to cover three landmarks while avoiding collisions, so success depends on implicit assignment and spatial dispersion. The agents use continuous movement actions and receive rewards based on landmark coverage together with collision penalties.

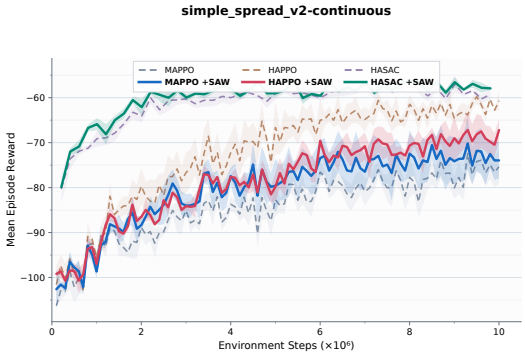


Figure 19: Learning curves of the six methods on `simple_spread_v2-continuous`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 19 shows modest improvements for MAPPO and HASAC, while the HAPPO+SAW-v2 curve finishes below HAPPO. MAPPO changes from -76.28 to -73.22 and HASAC changes from -58.85 to -57.49, corresponding to improvements of 4.0% and 2.3%. HAPPO changes from -62.31 to -68.79, corresponding to a decrease of 10.4%.

SMACv2

SMACv2 evaluates decentralized micromanagement with discrete actions, randomized unit compositions, and randomized starting positions. Each scenario contains five allied agents and five enemies from the same StarCraft race, but individual unit types have different movement, attack, healing, or area-effect capabilities. Across the three SMACv2 tasks, SAW-v2 improves seven of the nine backbone comparisons and achieves an average relative gain of 3.1%.

protoss_5_vs_5

The Protoss scenario samples teams from Zealots, Stalkers, and the less frequent Colossi, producing mixtures of melee, ranged, and high-impact area-damage roles. The five allied agents fight five enemies under randomized team compositions and starting positions.

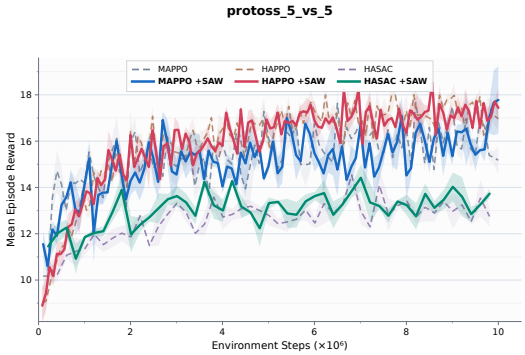


Figure 20: Learning curves of the six methods on `protoss_5_vs_5`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 20 shows small positive final differences for all three SAW-v2 variants. The final rewards increase from 16.04 to 16.33 for MAPPO, from 17.22 to 17.23 for HAPPO, and from 13.01 to 13.49 for HASAC. The corresponding relative improvements are 1.8%, 0.1%, and 3.8%.

terran_5_vs_5

The Terran scenario samples Marines, Marauders, and less frequent healing-only Medivacs, creating heterogeneous damage and support roles. The five-agent team must coordinate focus fire, positioning, and healing against five enemies generated from the same race-specific distribution.

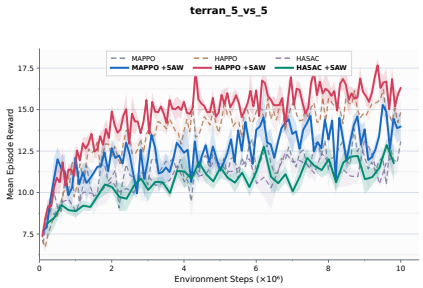


Figure 21: Learning curves of the six methods on `terran_5_vs_5`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 21 shows higher final curves for MAPPO+SAW-v2 and HAPPO+SAW-v2, while HASAC+SAW-v2 finishes slightly below HASAC. MAPPO increases from 11.89 to 13.55 and HAPPO increases from 15.03 to 16.21, corresponding to improvements of 13.9% and 7.9%. HASAC changes from 12.03 to 11.57, corresponding to a decrease of 3.8%.

zerg_5_vs_5

The Zerg scenario samples Zerglings, Hydralisks, and less frequent explosive Banelings, so short-range engagements can change abruptly when a Baneling detonates. The five allied units and five enemies are regenerated from this composition distribution and placed at randomized starting positions in each episode.

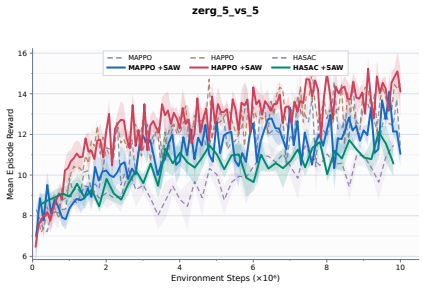


Figure 22: Learning curves of the six methods on `zerg_5_vs_5`. Solid lines denote the SAW-v2-enhanced variants, dashed lines denote the corresponding backbones, and shaded regions represent one standard error around the mean over three random seeds.

Figure 22 shows positive final differences for HAPPO and HASAC, while MAPPO+SAW-v2 finishes below MAPPO. HAPPO increases from 13.71 to 14.17 and HASAC increases from 10.67 to 11.17, corresponding to improvements of 3.4% and 4.7%. MAPPO changes from 12.79 to 12.34, corresponding to a decrease of 3.5%.