MAGIC: Multi-Step Advantage-Gated Causal Influence for Multi-agent Reinforcement Learning

Haohan Yu[1] Lu Wang[2] Jinmiao Cong[1] Shengzhi Wang[1] Chanjuan Liu[1,*]
[1]Dalian University of Technology [2]Microsoft, Beijing, China
15040488807@mail.dlut.edu.cn wlu@microsoft.com cjm111@mail.dlut.edu.cn
wangshengzhi@mail.dlut.edu.cn chanjuanliu@dlut.edu.cn
[*]Corresponding author

Abstract

A key challenge in multi-agent reinforcement learning (MARL) lies in designing learning signals that effectively promote coordination among agents. Designing such signals requires estimating how one agent's current action affects its teammates over future interaction steps. To address this, we introduce Multi-step Advantage-Gated Interventional Causal MARL (MAGIC), a framework that estimates multi-step action effects between agents and selectively converts them into intrinsic rewards. MAGIC uses counterfactual action interventions to compare teammate futures under factual and counterfactual branches, and introduces a gate based on advantage to direct exploration toward beneficial behaviors aligned with the task goal. Experiments on Multi-Agent Particle Environments (MPE) and StarCraft micromanagement benchmarks (SMAC and SMACv2) show that MAGIC consistently outperforms leading prior methods, with average relative final performance improvements of 26.9% and 10.1%, respectively.

1 Introduction

Cooperative multi-agent reinforcement learning (MARL) studies how multiple agents learn policies to optimize a shared team objective [Lowe et al., 2017, Rashid et al., 2018, Yu et al., 2022]. In this setting, each agent acts from its local observation, but the value of its action often depends on how it affects other agents and the future team outcome. This makes coordination difficult when team rewards are sparse or delayed, because an action that helps a teammate may not receive immediate feedback from the environment [Ma et al., 2022, Liu et al., 2023, Devidze et al., 2022, Forbes et al., 2024]. Centralized training with decentralized execution (CTDE) is widely used to alleviate this difficulty [Lowe et al., 2017, Rashid et al., 2018, Yu et al., 2022]. It allows the learner to use centralized information during training while keeping each agent's policy decentralized during execution. However, the feedback provided by the team reward remains indirect for each individual action. The shared return can evaluate the outcome of a joint behavior, but it does not directly indicate whether one agent's current local action will help its teammates coordinate in the future. This motivates an additional training signal that can make the cooperative value of local actions more explicit.

One common approach is to use inter-agent influence as an intrinsic reward [Jaques et al., 2019, Ma et al., 2022, Li et al., 2022, Du et al., 2024]. Early influence-based methods encourage coordination by measuring how much one agent affects others through immediate action responses, trajectory dependence, or mutual information [Jaques et al., 2019, Li et al., 2022, Jiang et al., 2024, Liu et al., 2024]. These signals show that influence can be useful for cooperative learning, but they also leave two key issues. First, useful influence may be delayed. A current action may have little effect on a teammate in the next transition, but may change the teammate's future position, route, or attack

Preprint.

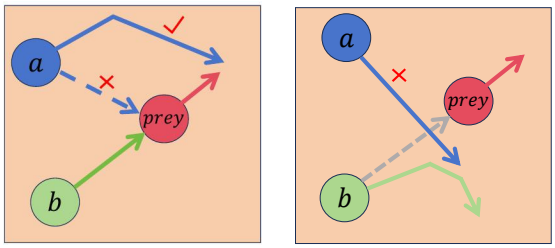


Figure 1: Motivating predator–prey example. Left: a delayed give-way action helps teammate b; right: a high-influence action disrupts b and hurts team return.

timing after several interaction steps. Second, strong influence is not necessarily useful. An agent can strongly affect its teammates while still reducing the team return. Thus, a useful coordination signal should capture delayed influence and should also be aligned with task improvement [Li et al., 2022, Forbes et al., 2024, Qin et al., 2025].

Figure 1 illustrates delayed useful influence and harmful influence in a cooperative predator-prey task. In the left panel, agent a's give-way action does not immediately bring it closer to the prey, but helps agent b reach a better future capture position. In the right panel, agent a strongly changes agent b's route, but this interference moves the team away from a better capture outcome. These examples make the desired training signal more specific. It should look beyond immediate action effects and encourage such effects only when they support the team objective.

Based on this principle, we propose Multi-step Advantage-Gated Interventional Causal MARL (MAGIC). MAGIC augments a standard CTDE learner with a causal and task-aligned intrinsic reward computed only during training. To estimate whether one agent's current action affects its teammates' future states, MAGIC constructs factual and counterfactual branches at each sampled time step. The factual branch keeps the realized action of the source agent, while the counterfactual branches replace only this source action with valid alternatives. All branches share the same current state and the same actions of the other agents. MAGIC then compares the teammate futures produced by these branches over multiple rollout steps. If replacing the source agent's current action leads to clearly different teammate futures, this indicates that the realized action has a strong counterfactual action effect in the current situation. This comparison is different from measuring statistical dependence between actions and future states in collected trajectories. It directly asks whether the teammates' future states would change if only the source agent's action were replaced in the same situation.

To compare the teammate futures produced by these branches over multiple rollout steps, MAGIC uses a learned forward model to roll out the factual and counterfactual branches from the same sampled decision context. In this design, the forward model is not used as an exact long-term simulator. Its role is to preserve the differences between branches that are relevant to teammate future states within a finite horizon. When the rollout keeps the relative strength of factual and counterfactual effects separable, MAGIC can estimate action effects reliably even if the absolute predicted states are imperfect.

After the action effect score is obtained, MAGIC aligns it with the task objective through an extrinsic advantage gate. The action effect score is agent-specific because it measures the effect of each agent's own realized action on its teammates' futures. The gate uses an extrinsic advantage computed at the team level to estimate whether the realized team transition improves expected task return. This gate keeps more of the action effect signal when the transition benefits the task and suppresses it when the transition is harmful. As a result, MAGIC encourages agents to produce effects on teammates only when such effects are supported by task improvement.

Our main contributions are as follows.

- We propose a multi-step counterfactual action effect estimator. It measures how replacing one agent's current action changes its teammates' future states over multiple rollout steps, which allows the learning signal to capture delayed coordination effects.
- We introduce an advantage-gated intrinsic reward that converts action effects into rewards only when they are supported by task improvement. The gate retains more intrinsic reward for beneficial team transitions and suppresses high-effect behaviors that reduce the expected task return.

- We evaluate MAGIC across multiple MARL benchmarks and task families, including Multi-Agent Particle Environments (MPE), StarCraft Multi-agent Challenge (SMAC), and SMACv2. MAGIC outperforms representative methods from the two related lines studied in this paper, including influence-based intrinsic-reward methods and task-aligned coordination methods. Beyond final performance, we use ablations to identify the role of each component and diagnostic studies to examine the reliability of the proposed method.

2 Related Work

Our work studies training signals for improving coordination in cooperative MARL. Related efforts include influence-based intrinsic rewards and task-aligned coordination signals.

Influence-based intrinsic rewards. Intrinsic rewards are widely used to encourage exploration and coordination under sparse or delayed rewards [Du et al., 2019, Ma et al., 2022, Liu et al., 2023, Devidze et al., 2022, Forbes et al., 2024]. In MARL, recent social-influence and coordinated-exploration methods encourage agents to affect each other through mutual information, action response, or interaction-state discovery [Jiang et al., 2024, Liu et al., 2024]. These methods show the value of inter-agent influence, but mainly capture correlation or immediate response. SCIC [Du et al., 2024] estimates single-step interventional causal influence under CTDE and uses it as an intrinsic reward, but it does not capture delayed action effects over multiple rollout steps or distinguish task-beneficial effects from harmful high-effect behaviors. Unlike mutual information based signals, MAGIC directly computes teammate future differences between factual and counterfactual branches, which avoids a separate mutual information critic in the action effect module and makes the score easier to diagnose.

Task-aligned coordination signals. Task consistency has long been recognized as important in reward shaping [Devidze et al., 2022, Wang et al., 2023, Forbes et al., 2024]. PMIC [Li et al., 2022] aligns mutual-information-based coordination with task return by increasing mutual information on high-return trajectories and decreasing it on low-return trajectories. However, trajectory-level alignment provides coarse and delayed feedback, so it cannot directly decide whether a specific action effect at the current transition should be encouraged. GradPS [Qin et al., 2025] improves cooperation by adapting parameter sharing and policy diversity, which helps form more suitable shared representations for coordination. However, this adjustment acts at the policy-structure level and does not provide an immediate training signal for evaluating whether an agent's current action benefits its teammates.

3 Method

We propose MAGIC, an intrinsic reward module used during training on top of a standard CTDE backbone. As shown in Figure 2, MAGIC compares factual and counterfactual branches that differ only in the source agent's action. A learned forward model rolls out these branches for a finite horizon, and teammate future differences are aggregated into an action effect score $c_i(t)$. An extrinsic team advantage gate then filters this score to form the intrinsic reward $r_{i,t}^{\text{int}}$, which is added to the environment reward during training. The module is removed during execution. Section 3.1 defines the counterfactual action effect, Section 3.2 describes action effect estimation and aggregation with learned forward model rollouts, and Section 3.3 presents the advantage-gated training objective.

3.1 Counterfactual Action Effect

The quantity of interest is the causal effect of a source agent's realized action on its teammates' future states under the same decision context. This differs from statistical dependence in collected trajectories. A source action may be correlated with a teammate future simply because it appears in states that already lead to that future. Such dependence does not tell whether the teammate future would change if only the source action were replaced.

We therefore define the action effect through an intervention that replaces the source action. For source agent i at time t, the decision context consists of the centralized state s_t and the joint action of the other agents a_t^{-i}. The factual branch keeps the realized source action, written as (a_t^i, a_t^{-i}, s_t). The counterfactual branches use the same a_t^{-i} and s_t, but replace a_t^i with K valid alternatives, written as $\{(a_t^{i,k}, a_t^{-i}, s_t)\}_{k=1}^{K}$. Thus, the only changed component is the source action.

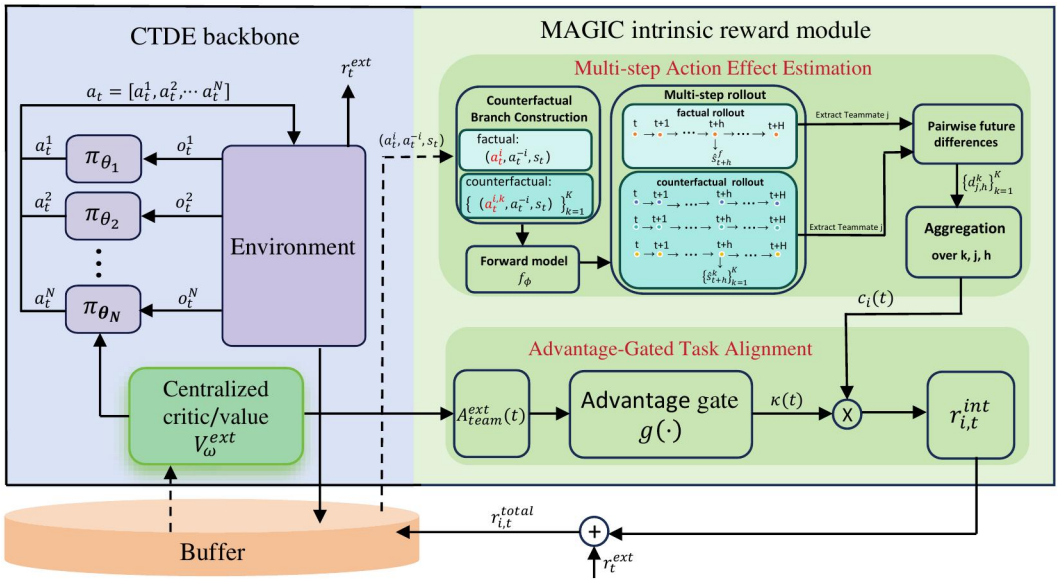


Figure 2: Overview of MAGIC. MAGIC adds an intrinsic reward module used during training to a CTDE backbone. It builds factual and counterfactual branches by replacing only the source agent's action, rolls out predicted centralized states with a learned forward model, extracts teammate future features, and aggregates factual and counterfactual differences into an action effect score. An extrinsic team-advantage gate modulates this score before it is added as intrinsic reward. The module is removed during execution.

The effect of the realized action is then defined by the difference between teammate futures under the factual and counterfactual branches. If replacing a_t^i leads to substantially different teammate futures, then the realized action has a strong counterfactual action effect in the current context. If the teammate futures remain similar after replacement, then the realized action has a weak effect in that context.

3.2 Multi-Step Action Effect Estimation

The factual and counterfactual branches defined above specify which action inputs should be compared. To obtain their future outcomes, MAGIC uses a learned forward model during training. The forward model takes the joint action and centralized state as input and predicts the next centralized state,

$$\hat{s}_{t+1} = f_\phi(a_t, s_t). \tag{1}$$

It is trained from real environment transitions with a one step prediction loss. During action effect estimation, the model starts from the same sampled decision context and rolls out the factual and counterfactual branches over a finite horizon. These predicted branch futures are then used for teammate feature comparison and aggregation, rather than being treated as action effect scores directly. The exact loss and rollout details are given in Appendix A.2.

For a source agent i, the factual branch starts from (a_t^i, a_t^{-i}, s_t), while the k-th counterfactual branch starts from $(a_t^{i,k}, a_t^{-i}, s_t)$. At the first rollout step, the state s_t and the non-source actions a_t^{-i} are shared, and only the source action differs across branches. After this first step, all agents act according to their current policies on the model-predicted states. In this way, the rollout is closed-loop, allowing the effect of the source action to propagate through later policy responses.

The forward model rolls out full centralized states rather than isolated teammate states. We denote the factual rollout states by $\{\hat{s}_{t+h}^f\}_{h=1}^H$ and the states of the k-th counterfactual rollout by $\{\hat{s}_{t+h}^k\}_{h=1}^H$. For each horizon h, teammate features are extracted from these predicted centralized states before computing the effect. Let $z_j(\hat{s})$ denote the normalized future-state feature vector of teammate j extracted from a predicted centralized state $\hat{s}$. The pairwise future difference between the factual

branch and the k-th counterfactual branch is

$$d_{j,h}^{(k)} = \mathrm{dist}\left(z_j(\hat{s}_{t+h}^{f}), z_j(\hat{s}_{t+h}^{k})\right), \tag{2}$$

where $\mathrm{dist}(\cdot,\cdot)$ is computed in the normalized teammate-feature space. Thus, for each teammate j and horizon h, MAGIC obtains a set of pairwise differences $\{d_{j,h}^{(k)}\}_{k=1}^{K}$ by comparing the factual future with each counterfactual future.

The pairwise differences are first averaged over the K counterfactual branches,

$$d_{j,h} = \frac{1}{K}\sum_{k=1}^{K} d_{j,h}^{(k)}. \tag{3}$$

The resulting effects are then aggregated over teammates and rollout horizons to obtain the multi-step action effect score for source agent i,

$$\tilde{c}_i(t) = \sum_{h=1}^{H} w_h \frac{1}{N-1}\sum_{j\neq i} d_{j,h}, \qquad w_h \geq 0, \quad \sum_{h=1}^{H} w_h = 1. \tag{4}$$

Here, w_h controls the contribution of each rollout horizon. When $H = 1$, the score only captures immediate teammate-state changes. When $H > 1$, the score can include effects that appear after several interaction steps.

To keep the intrinsic signal on a stable scale, we normalize and clip $\tilde{c}_i(t)$ with running statistics, yielding the scaled action effect score $c_i(t)$ used by the reward module. Details of counterfactual action sampling, forward model rollouts, teammate-feature normalization, action effect score scaling, and rollout pseudocode are provided in Appendix A.1–A.6.

3.3 Advantage-Gated Training Objective

The scaled action effect score $c_i(t)$ measures how strongly the realized action of agent i changes teammate futures. This effect is not necessarily aligned with the task objective. A large action effect may help teammates coordinate, but it may also disturb their future states in a harmful way. We therefore gate the scaled action effect score with an extrinsic team advantage before using it as intrinsic reward.

The extrinsic team advantage measures whether the realized team transition is beneficial under the original environment reward. Using the centralized value estimate associated with the extrinsic task reward, we compute

$$A_{\mathrm{team}}^{\mathrm{ext}}(t) = r_t^{\mathrm{ext}} + \gamma V_\omega^{\mathrm{ext}}(s_{t+1}) - V_\omega^{\mathrm{ext}}(s_t). \tag{5}$$

The superscript ext indicates that this advantage is computed with respect to the original task reward rather than the shaped reward. This keeps the gate tied to the task objective.

The gate is defined as a bounded monotone function of the extrinsic team advantage,

$$\kappa(t) = g\big(A_{\mathrm{team}}^{\mathrm{ext}}(t)\big), \qquad 0 \leq \kappa(t) \leq 1. \tag{6}$$

A larger extrinsic advantage leads to a larger gate value, so action effects observed in beneficial team transitions contribute more to the intrinsic reward. A smaller or negative extrinsic advantage suppresses the contribution of the scaled action effect score.

The intrinsic reward for source agent i is then

$$r_{i,t}^{\mathrm{int}} = \lambda_{\mathrm{int}}\kappa(t)c_i(t), \tag{7}$$

where λ_{int} controls the strength of the intrinsic reward. The total reward used for policy learning is

$$r_{i,t}^{\mathrm{total}} = r_t^{\mathrm{ext}} + r_{i,t}^{\mathrm{int}}. \tag{8}$$

The gate $\kappa(t)$ is intentionally defined at the team level. MAGIC separates two roles. The score $c_i(t)$ measures whether the realized action of source agent i changes its teammates' future states. The gate $\kappa(t)$ decides whether the realized team transition is aligned with the extrinsic task objective. Thus, the gate is not used as an individual credit-assignment estimator. It acts as a conservative filter for

task alignment shared by all agents at the same transition. Agent specificity is still preserved by $c_i(t)$, since different source agents receive different intrinsic rewards through their own action effect scores.

The CTDE learner is optimized with $r_{i,t}^{\text{total}}$. The forward model is updated with the one-step prediction loss defined in Section 3.2. The forward model, counterfactual rollouts, action effect computation, and intrinsic reward are used only during training. During execution, agents act with their decentralized policies without access to the MAGIC module. Details of the gate implementation, advantage normalization, reward scaling, full training pseudocode, and execution-time behavior are provided in Appendix A.5–A.7.

3.4 Theoretical Properties and Scope of the Analysis

We provide the formal properties of MAGIC in Appendix A.8–A.10. The analysis focuses on the action effect estimator defined above rather than on a mutual-information surrogate. First, we show that the unnormalized multi-step action effect score is nonnegative and becomes zero when factual and counterfactual branches induce identical teammate futures. Second, we show that the one-step version can be blind to delayed action effects, while the multi-step score detects effects that appear after several interaction steps. Third, we bound the error of the estimated action effect score in terms of forward model rollout errors. Finally, we analyze the extrinsic advantage gate as a bounded task-alignment multiplier.

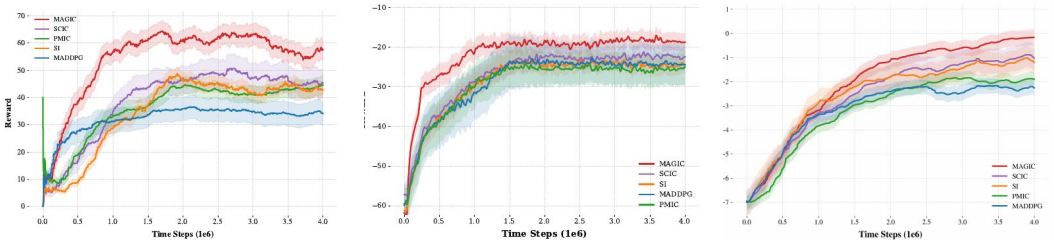
The theoretical statements are separated from the normalization used for numerical stability. The exact nonnegativity and zero-effect properties are stated for the unnormalized score $\tilde{c}_i(t)$. In implementation, we apply a bounded normalization and clipping map before constructing the intrinsic reward. Appendix A.9 shows that this transformation keeps the reward bounded and, when the map is monotone, does not reverse the relative ordering of action effect scores except for possible saturation caused by clipping. Thus, the theory describes the underlying action effect estimator, while the practical normalization controls scale without changing the intended comparison between factual and counterfactual branches.

4 Experiments

We evaluate MAGIC on two benchmark families that cover complementary coordination regimes. The first family uses three continuous control tasks from the Multi-agent Particle Environment (MPE) [Lowe et al., 2017]: *Predator Prey*, *Cooperative Navigation*, and *Cooperative Competitive*. These tasks test cooperative pursuit, landmark coverage, and mixed cooperative-competitive interaction in particle world environments. The second family uses StarCraft micromanagement benchmarks, including SMAC [Samvelyan et al., 2019] and SMACv2 [Ellis et al., 2023], where agents coordinate under partial observability, discrete actions, and delayed team rewards. Across these benchmarks, we first test whether the proposed intrinsic signal improves standard MARL performance, then examine whether the same design remains effective in harder micromanagement tasks, and finally analyze which components drive the gains and when the action effect score estimated with learned forward model rollouts remains reliable.

Baselines and backbones. We use matched protocols within each benchmark group for fair comparison. On MPE, all methods share the same MADDPG CTDE backbone [Lowe et al., 2017], following the common protocol used by prior MPE methods with intrinsic rewards, including SI [Jaques et al., 2019], PMIC [Li et al., 2022], and SCIC [Du et al., 2024]. MADDPG is the plain backbone without intrinsic rewards.

On SMAC and SMACv2, methods based on policy gradients are evaluated under a unified MAPPO CTDE protocol [Yu et al., 2022] implemented in PyMARL2. We include MAPPO as the plain policy gradient backbone, QMIX [Rashid et al., 2018] as a representative value decomposition baseline, PMIC as a mutual information baseline aligned with task return, SCIC as a single step causal influence baseline using intrinsic rewards, and GradPS [Qin et al., 2025] as a parameter sharing method for task aligned coordination. Within each benchmark group, methods use matched observation spaces, action masks, training budgets, random seeds, and evaluation procedures. Full implementation details are provided in Appendix B.



(a) Predator Prey (b) Cooperative Navigation (c) Cooperative Competitive

Figure 3: Learning curves on MPE tasks over five random seeds. Shaded regions denote standard deviation across seeds.

Table 1: Benchmark performance on SMAC and SMACv2 over five random seeds. Each entry reports final Win% / AUC. AUC is computed from the win-rate learning curve. Standard deviations are provided in Appendix B.8.

Method	3s5z	5m_vs_6m	corridor	6h_vs_8z	MMM2	Protoss5v5	Avg.
QMIX	92.4 / 78.4	78.5 / 52.3	48.2 / 25.1	58.6 / 32.5	74.2 / 42.1	36.8 / 15.4	64.8 / 41.0
MAPPO	94.1 / 80.2	90.5 / 68.5	54.8 / 30.6	71.2 / 44.1	81.4 / 51.5	47.5 / 22.3	73.3 / 49.5
PMIC	89.3 / 72.1	83.7 / 58.4	61.2 / 35.2	65.8 / 38.6	72.1 / 41.8	44.3 / 19.5	69.4 / 44.3
GradPS	94.8 / 81.5	91.2 / 70.1	64.5 / 38.4	74.0 / 46.8	81.8 / 53.2	49.5 / 24.1	76.0 / 52.4
SCIC	95.2 / 82.3	91.8 / 71.4	72.6 / 45.8	76.5 / 49.2	80.3 / 51.8	52.4 / 26.5	78.1 / 54.5
MAGIC	**97.8 / 86.5**	**95.6 / 78.2**	**83.4 / 55.6**	**85.2 / 59.4**	**87.3 / 62.3**	**66.5 / 38.1**	**86.0 / 63.4**

Evaluation protocol. All experiments are run with five random seeds under matched training budgets, observation settings, and evaluation procedures within each benchmark group. For MPE, we use episodic team return as the main metric, since it is the standard task metric for these continuous-control environments and directly reflects team reward. We report final return and AUC, where AUC is computed from the team-return learning curve and summarizes learning efficiency over training. We also include learning curves to show convergence behavior. For SMAC and SMACv2, we use win rate as the main task metric and report final win rate, standard deviation across seeds, and AUC computed from the win-rate learning curve. Additional environment configurations, implementation details, hyperparameters, statistical tests, hardware setup, and runtime analysis are provided in Appendix B.

4.1 Benchmark Performance

MPE continuous-control tasks. Figure 3 compares learning curves on *Predator Prey*, *Cooperative Navigation*, and *Cooperative Competitive*. MAGIC improves convergence and final team return across all three MPE tasks. With relative gains normalized by the magnitude of the SCIC return, MAGIC improves over SCIC by 26.9%, 17.5%, and 36.4% on *Predator Prey*, *Cooperative Navigation*, and *Cooperative Competitive*, respectively, yielding an average relative final-return gain of 26.9%. These results indicate that the proposed signal is useful not only in sparse pursuit but also in dense reward and mixed cooperative-competitive interaction regimes. Appendix B.7 reports the full MPE scalar metrics, including standard deviations, best returns, AUC, significance tests, and a 10-agent Predator Prey check where MAGIC remains strongest among the compared methods.

SMAC and SMACv2 micromanagement tasks. Table 1 reports final win rate and AUC over five seeds, with standard deviations provided in Appendix B.8. MAGIC obtains the highest average performance, improving over SCIC by 10.1% in average final win rate and 16.3% in average AUC. The gains are larger on harder maps such as `corridor`, `6h_vs_8z`, and `Protoss5v5`, where agents must coordinate under bottlenecks, heterogeneous combat roles, or stronger partial observability. Compared with GradPS, MAGIC also achieves higher average win rate and AUC. These results suggest that multi-step action effect estimation remains useful when coordination effects are not immediately visible in the next transition.

Table 2: Component analysis across MPE and SMAC. Panel A reports final and best returns on MPE Predator Prey. Panel B reports final Win% on representative SMAC maps, with Avg. denoting the average across the three maps. Standard deviations and full MPE scalar metrics are provided in Appendix B.9.

Panel A. Module ablation on MPE Predator Prey

Method	Final return	Best return	Change from MAGIC
MAGIC	**57.6**	**62.1**	–
MAGIC w/o Advantage Gating	47.3	48.9	-17.9%
MAGIC H=1 Action Effect	41.8	43.4	-27.4%

Panel B. Core component attribution on SMAC

Method	`corridor`	`5m_vs_6m`	`MMM2`	Avg.
MAGIC H=3	**83.4**	**95.6**	**87.3**	**88.8**
MAGIC H=3 w/o Gate	77.5	91.8	83.8	84.4
MAGIC H=1+Gate	73.1	92.4	82.1	82.5
MAPPO	54.8	90.5	81.4	75.6

4.2 Ablation Study of Multi-Step Action Effects and Advantage Gating

We next examine which parts of MAGIC are responsible for the improvement. The two components of interest are action effect estimation over multiple rollout steps and advantage gating. We use two complementary analyses. The first analysis ablates individual modules on MPE Predator Prey, where the full set of MAGIC components can be cleanly isolated. The second analysis compares the full method with a one step variant and an ungated variant on representative SMAC maps, testing whether the two components remain useful when coordination is harder and rollout error becomes more relevant.

Panel A of Table 2 shows that both design components contribute to performance on MPE. Replacing the multi-step action effect estimator with a one step version causes the largest drop, which indicates that one-step effects are insufficient for capturing delayed cooperative consequences. Removing the advantage gate also reduces performance, but the ungated multi-step variant still remains stronger than the one-step variant. This suggests that the multi-step action effect signal itself captures useful delayed coordination information, while the advantage gate further improves this signal by emphasizing effects observed in task-beneficial transitions.

Panel B of Table 2 tests whether the same two design choices remain useful on harder SMAC maps. Using a one-step action effect signal with the gate improves over MAPPO on all three maps, but remains below the full method. Using the multi-step action effect signal without the gate also improves over MAPPO on all three maps, showing that delayed action effect estimation is useful by itself. Adding the advantage gate further improves performance on all three maps, indicating that the gate helps convert the delayed action effect signal into a training signal that is better aligned with the task. Together, these results suggest that the two components are complementary. Multi-step action effect estimation provides the delayed interaction signal, while advantage gating makes this signal more task-aligned and more effective across environments.

4.3 Reliability and Horizon Sensitivity of Multi-Step Action Effect Estimation

Since MAGIC estimates multi-step action effects through learned forward model rollouts, we evaluate the reliability of this estimation from three complementary views in Table 3. Panel A fixes the main horizon at $H = 3$ and tests whether separability of branch effects tracks downstream performance as the forward model update ratio changes. Panel B studies the sensitivity to the horizon parameter H, which guides the choice of rollout depth. Panel C explicitly corrupts the forward model at $H = 3$ to stress-test the estimator.

We diagnose the reliability of the learned rollouts from two sides: whether the forward model predicts future states accurately, and whether its predicted branch differences preserve the true strength of action effects. *In-MSE* measures prediction error on normal policy rollouts, where all agents follow

Table 3: Reliability and horizon sensitivity of multi-step action effect estimation. Panel A fixes $H = 3$ and varies the forward model update ratio on `5m_vs_6m`. Panel B reports horizon sensitivity averaged over three maps. Panel C reports explicit forward model corruption on `5m_vs_6m` with $H = 3$.

Panel A. Fixed-horizon reliability

Update ratio	In-MSE	Int-MSE	Sep. AUC	Win%
0.25×	0.135	0.228	0.57	86.8
0.50×	0.072	0.108	0.74	91.2
0.75×	0.038	0.051	0.87	94.3
1.00×	0.025	0.032	0.91	**95.6**

Panel B. Horizon sensitivity

H	In-MSE	Int-MSE	Sep. AUC	Win%
1	0.010	0.013	0.94	82.5
2	0.018	0.024	0.92	86.1
3	0.031	0.039	0.90	**88.8**
5	0.065	0.088	0.82	86.3
8	0.151	0.218	0.58	74.7
10	0.228	0.345	0.49	68.2

Panel C. Forward model corruption

Noise	Int-MSE	Sep. AUC	Win%
0.0	0.032	0.91	**95.6**
0.1	0.055	0.88	94.2
0.5	0.120	0.79	91.5
1.0	0.280	0.55	82.4

their learned policies. *Int-MSE* measures prediction error on intervention rollouts, where the selected agent's action is replaced while the other agents follow their learned policies. These two errors indicate how accurate the learned dynamics are under factual and counterfactual rollout conditions. *Sep. AUC* measures whether the model-predicted teammate-future differences reflect the true action effect strength. A value near 0.5 indicates random separation between large-effect and small-effect branches. A higher value means that the predicted branch differences better identify actions that truly cause larger changes in teammate futures.

Panel A of Table 3 fixes the main horizon at $H = 3$ and varies only the forward model update ratio. As the update ratio increases from $0.25\times$ to $1.00\times$, both in-distribution and intervention rollout errors decrease, Sep. AUC increases from 0.57 to 0.91, and the win rate improves from 86.8 to 95.6. With the horizon fixed, this shows that separability of branch effects tracks downstream performance under the main setting.

Panel B of Table 3 is a horizon sensitivity study. Increasing the horizon from $H = 1$ to $H = 3$ improves performance while keeping rollout error in a reliable range. This indicates that one-step action effects miss useful delayed effects. Increasing the horizon further eventually reverses this trend. At $H = 8$ and $H = 10$, intervention error becomes much larger, Sep. AUC approaches random separation, and win rate drops. Thus the useful horizon is not simply the largest possible rollout depth. It is the range in which delayed interaction effects become visible before model error destroys separability of branch effects. The same rise-then-decline pattern also appears on MPE Predator Prey, with full results reported in Appendix B.12.

Panel C of Table 3 tests robustness by explicitly corrupting the forward model at the main horizon $H = 3$. Moderate corruption increases intervention MSE, but performance remains strong as long as Sep. AUC stays high. When corruption becomes large, Sep. AUC drops toward 0.5 and task performance falls accordingly. This supports the view that separability of branch effects is a useful reliability diagnostic for MAGIC, while raw MSE is an upstream diagnostic.

Additional appendix diagnostics examine this reliability criterion under counterfactual sampling, stochasticity, and artificial delay. Appendix B.13 varies the number of counterfactual branches on continuous MPE Predator Prey. Appendix B.14 and Appendix B.15 test stochasticity in SMAC action execution and MPE dynamics. Appendix B.16 studies artificial reward delays and shows that larger horizons help only while separability remains preserved. Together, these results show that MAGIC is useful when learned rollouts preserve effect differences between branches, and that longer rollouts or more branch samples help only within this reliable regime.

5 Conclusion

We propose MAGIC, a multi-step advantage-gated action effect framework for cooperative MARL. MAGIC estimates whether one agent's current action changes its teammates' future states over multiple rollout steps, and uses an extrinsic advantage gate to convert this signal into task-aligned intrinsic reward. This separates action effect estimation from task-value alignment and reduces the risk of rewarding harmful high-effect behaviors. Experiments across MPE, SMAC, and SMACv2 show that MAGIC achieves the best average performance among strong influence-based and task-aligned coordination baselines. Ablations and diagnostics show that the gains depend on both multi-step estimation and advantage gating, and that the method remains reliable when learned rollouts preserve separability of branch effects.

References

Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. Exploration-guided reward shaping for reinforcement learning under sparse rewards. In *Advances in Neural Information Processing Systems*, volume 35, pages 5829–5842, 2022.

Xiao Du, Yutong Ye, Pengyu Zhang, Yaning Yang, Mingsong Chen, and Ting Wang. Situation-dependent causal influence-based cooperative multi-agent reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17362–17370, 2024.

Yali Du, Lei Han, Meng Fang, Tianhong Dai, Ji Liu, and Dacheng Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 4403–4414, 2019.

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

Grant C. Forbes, Nitish Gupta, Leonardo Villalobos-Arias, Colin M. Potts, Arnav Jhala, and David L. Roberts. Potential-based reward shaping for intrinsic motivation. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 589–597. International Foundation for Autonomous Agents and Multiagent Systems, 2024.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3040–3049. PMLR, 2019.

Haobin Jiang, Ziluo Ding, and Zongqing Lu. Settling decentralized multi-agent coordinated exploration by novelty sharing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17444–17452, 2024.

Pengyi Li, Hongyao Tang, Tianpei Yang, Xiaotian Hao, Tong Sang, Yan Zheng, Jianye Hao, Matthew E. Taylor, Wenyuan Tao, and Zhen Wang. Pmic: Improving multi-agent reinforcement learning with progressive mutual information collaboration. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 12979–12997. PMLR, 2022.

Boyin Liu, Zhiqiang Pu, Yi Pan, Jianqiang Yi, Yanyan Liang, and Du Zhang. Lazy agents: A new perspective on solving sparse reward problem in multi-agent reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 21937–21950. PMLR, 2023.

Zeyang Liu, Lipeng Wan, Xinrui Yang, Zhuoran Chen, Xingyu Chen, and Xuguang Lan. Imagine, initialize, and explore: An effective exploration method in multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17487–17495, 2024.

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

Zixian Ma, Rose E. Wang, Li Fei-Fei, Michael S. Bernstein, and Ranjay Krishna. Elign: Expectation alignment as a multi-agent intrinsic reward. In *Advances in Neural Information Processing Systems*, volume 35, pages 8304–8317, 2022.

Haoyuan Qin, Zhengzhu Liu, Chenxing Lin, Chennan Ma, Songzhu Mei, Siqi Shen, and Cheng Wang. GradPS: Resolving futile neurons in parameter sharing network for multi-agent reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 50246–50268. PMLR, 2025.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4295–4304. PMLR, 2018.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2186–2188, 2019.

Justin K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S. Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall L. Williams, Yashas Lokesh, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 15032–15043, 2021.

Yiming Wang, Ming Yang, Renzhi Dong, Binbin Sun, Furui Liu, and Leong Hou U. Efficient potential-based exploration in reinforcement learning using inverse dynamic bisimulation metric. In *Advances in Neural Information Processing Systems*, volume 36, 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. In *Advances in Neural Information Processing Systems*, volume 35, pages 24611–24624, 2022.

Impact Statement

This work develops a training-time method for improving coordination in cooperative MARL. The experiments are limited to simulated benchmark environments and the method is not deployed in real-world systems. As with general MARL methods, real-world use would require domain-specific safety evaluation, especially in settings involving autonomous coordination.

A Additional Method Details

This appendix provides the implementation details and theoretical properties referenced by Section 3. We keep the terminology consistent with the main text: MAGIC computes an agent-specific action effect score $c_i(t)$ from factual and counterfactual teammate futures, and then modulates this score with a team-level extrinsic advantage gate $\kappa(t)$. The forward model, counterfactual rollouts, action effect computation, and advantage gate are used only during training.

A.1 Counterfactual Branch Construction

For each sampled transition $(s_t, a_t, r_t^{\text{ext}}, s_{t+1})$, MAGIC constructs branch comparisons separately for each possible source agent i. The realized joint action is written as $a_t = (a_t^i, a_t^{-i})$, where a_t^i is the action of the source agent and a_t^{-i} denotes the actions of all other agents. The factual branch is

$$b_t^{f,i} = (a_t^i, a_t^{-i}, s_t). \tag{9}$$

The k-th counterfactual branch is

$$b_t^{k,i} = (a_t^{i,k}, a_t^{-i}, s_t), \qquad k = 1, \ldots, K, \tag{10}$$

where $a_t^{i,k}$ is a valid alternative action for agent i. This construction changes only the source action. The centralized state s_t and the non-source actions a_t^{-i} are kept fixed across the factual and counterfactual branches. Therefore the branch difference measures the effect of replacing the source agent's current action under the same decision context.

Counterfactual actions are sampled from the valid action set of the source agent. In discrete-action tasks, invalid actions are excluded using the environment action mask. If the executed action is drawn as a counterfactual candidate, we resample it when another valid action is available, so that each counterfactual branch represents an actual replacement. In continuous-control tasks, counterfactual actions are sampled within the bounded action range of the environment and clipped to the same

action limits used by the policy. The number of counterfactual branches K is fixed for a run. Unless otherwise stated, the main experiments use $K = 64$. The counterfactual branches are not used to search for an optimal replacement action. They provide a Monte Carlo estimate of how much teammate futures change under valid replacements of the source action. Thus, increasing K improves the coverage of alternative actions, but the estimator does not require dense optimization over the action space.

A.2 Forward Model Training and Closed-Loop Rollouts

The forward model f_ϕ predicts the next centralized state from the current centralized state and joint action:

$$\hat{s}_{t+1} = f_\phi(a_t, s_t). \tag{11}$$

It is trained with real environment transitions and the one-step squared prediction loss

$$\mathcal{L}_{\text{fm}}(\phi) = \|f_\phi(a_t, s_t) - s_{t+1}\|_2^2 . \tag{12}$$

The forward model is not trained to predict the action effect score. It only provides predicted future centralized states for the factual and counterfactual branches.

Rollouts are closed-loop after the first action-replacement step. At the first model step, the factual branch uses (a_t^i, a_t^{-i}) and the k-th counterfactual branch uses $(a_t^{i,k}, a_t^{-i})$. After this step, each branch evolves with the current decentralized policies applied to the model-predicted state. At each closed-loop rollout step, the predicted centralized state is converted into each agent's local observation using the same observation construction as the environment or the benchmark wrapper. For discrete-action tasks, the valid action mask from the corresponding predicted or current rollout state is used when available; otherwise invalid actions are excluded according to the benchmark action constraints. In SMAC and SMACv2, the predicted centralized state is decoded into unit-level features following the PyMARL2 state representation. We recompute local observations and action availability from these predicted unit features using the same visibility, attack-range, alive-state, and movement-boundary rules as the environment wrapper. Predicted continuous fields are clipped to valid ranges before observation and mask construction. If a unit is predicted as dead, only the no-op action is marked available. When a policy is stochastic, rollout actions are sampled from the current policy during training. For diagnostic evaluation, we use the same deterministic evaluation convention as the corresponding backbone. This closed-loop procedure allows the first action replacement to influence later states through subsequent policy responses, rather than only measuring a one-step state perturbation.

For a source agent i, the factual rollout produces $\{\hat{s}_{t+h}^f\}_{h=1}^H$ and the k-th counterfactual rollout produces $\{\hat{s}_{t+h}^k\}_{h=1}^H$. All rollout states are full centralized states. Teammate-specific features are extracted after the rollout, rather than being predicted by separate teammate-specific models.

A.3 Teammate Feature Extraction and Normalization

Let $z_j(\hat{s})$ denote the feature vector extracted for teammate j from a predicted centralized state $\hat{s}$. The extractor uses the centralized state representation available during CTDE training and selects the components associated with teammate j. In MPE tasks, these components include task-relevant physical state variables such as position and velocity. In SMAC and SMACv2 tasks, they include the centralized unit features associated with the teammate unit, such as position, health-related state, alive status, and other unit-level state variables available to the centralized critic. The extractor does not use hidden information at execution time, because the entire MAGIC module is training-time only.

Before computing distances, each feature dimension is normalized using running statistics collected from training batches. For a feature vector z, we use

$$\bar{z} = \frac{z - \mu_z}{\sigma_z + \epsilon}, \tag{13}$$

where μ_z and σ_z are running mean and standard deviation estimates and ϵ is a small constant for numerical stability. The feature statistics are updated once per training update using batch exponential moving averages with momentum 0.99, pooled over sampled transitions and agents in

the training batch. The same normalization statistics are used for factual and counterfactual branches. The pairwise branch difference is then computed in this normalized feature space. In the main experiments we use the Euclidean distance,

$$d_{j,h}^{(k)} = \left\| \bar{z}_j(\hat{s}_{t+h}^f) - \bar{z}_j(\hat{s}_{t+h}^k) \right\|_2 . \tag{14}$$

This distance measures how much the predicted future state of teammate j changes when only the source action is replaced.

A.4 Action Effect Score Aggregation and Scaling

For each source agent i, MAGIC first averages the pairwise differences over the K counterfactual branches:

$$d_{j,h} = \frac{1}{K} \sum_{k=1}^{K} d_{j,h}^{(k)}. \tag{15}$$

It then averages over teammates and aggregates over rollout horizons:

$$\tilde{c}_i(t) = \sum_{h=1}^{H} w_h \frac{1}{N-1} \sum_{j \neq i} d_{j,h}, \qquad w_h \geq 0, \quad \sum_{h=1}^{H} w_h = 1. \tag{16}$$

In the main experiments we use uniform horizon weights unless otherwise stated. The quantity $\tilde{c}_i(t)$ is the unnormalized multi-step action effect magnitude. To keep the reward scale stable, we divide it by a running scale estimate and clip the result:

$$c_i(t) = \mathrm{clip}\left(\frac{\tilde{c}_i(t)}{\sigma_c + \epsilon}, 0, c_{\max} \right). \tag{17}$$

where σ_c is a running standard deviation estimate of $\tilde{c}_i(t)$ over training batches. It is updated with the same batch exponential moving average rule used for teammate-feature normalization and is pooled over source agents and sampled transitions. This scaling preserves the non-negativity of the action effect score while preventing a small number of large branch differences from dominating the intrinsic reward.

A.5 Advantage Gate, Advantage Normalization, and Reward Scaling

The gate is based on an extrinsic team advantage. We maintain or reuse a centralized value estimate V_ω^{ext} trained only with the original environment reward. In the MADDPG-based MPE experiments, V_ω^{ext} is implemented as a separate centralized state-value network and is trained with one-step TD targets $r_t^{\mathrm{ext}} + \gamma V_{\bar{\omega}}^{\mathrm{ext}}(s_{t+1})$. In the MAPPO-based SMAC and SMACv2 experiments, we maintain an extrinsic-only value head with the same value architecture and update it from extrinsic-return targets. In both cases, this value estimate is never trained with the intrinsic reward or the total shaped reward. The one-step extrinsic TD advantage is

$$A_{\mathrm{team}}^{\mathrm{ext}}(t) = r_t^{\mathrm{ext}} + \gamma V_\omega^{\mathrm{ext}}(s_{t+1}) - V_\omega^{\mathrm{ext}}(s_t). \tag{18}$$

This advantage does not include the intrinsic reward or the total shaped reward. Thus the gate is tied to the original task objective and does not use the shaped reward to decide its own strength.

In implementation, we normalize the scalar advantage before applying the gate:

$$\bar{A}_{\mathrm{team}}^{\mathrm{ext}}(t) = \frac{A_{\mathrm{team}}^{\mathrm{ext}}(t) - \mu_A}{\sigma_A + \epsilon}, \tag{19}$$

where μ_A and σ_A are running statistics over training batches. They are updated once per training update using batch exponential moving averages with momentum 0.99, pooled over transitions in the training batch. The gate is then

$$\kappa(t) = g\big(\bar{A}_{\mathrm{team}}^{\mathrm{ext}}(t)\big), \tag{20}$$

where g is a bounded monotone function. We use a sigmoid gate in the experiments. The normalization is part of the implementation of g and does not change the definition in the main text.

The gate $\kappa(t)$ is shared by all agents at the same time step because it evaluates the task value of the realized team transition. The action effect score $c_i(t)$ remains agent-specific because it is computed from the branch comparisons of source agent i. The intrinsic reward is

$$r_{i,t}^{\text{int}} = \lambda_{\text{int}} \kappa(t) c_i(t), \tag{21}$$

where λ_{int} controls the strength of the intrinsic reward. The total reward used for the policy and critic update is

$$r_{i,t}^{\text{total}} = r_t^{\text{ext}} + r_{i,t}^{\text{int}}. \tag{22}$$

The shaped reward is kept agent-specific. Each agent's policy and value target use its own $r_{i,t}^{\text{total}}$, while the extrinsic gate is shared across agents. We do not average $r_{i,t}^{\text{int}}$ across agents before constructing policy or value targets. After $c_i(t)$ is clipped by Eq. (17), we do not apply an additional intrinsic-reward clipping step in the main experiments. The intrinsic reward scale is controlled by $c_{\max}$ and λ_{int}.

A.6 Full Training Procedure

Algorithm 1 summarizes the training procedure. The algorithm is written for a generic CTDE backbone. In off-policy experiments, sampled transitions come from a replay buffer. In on-policy MAPPO experiments, they come from the current rollout batch. In both cases, the intrinsic reward is computed from training samples without additional environment interaction.

Algorithm 1 Training procedure of MAGIC

Initialize CTDE policies, centralized critic or value functions, extrinsic value estimate V_ω^{ext}, forward model f_ϕ, and training buffer or rollout storage.
for each training iteration **do**
  Collect environment transitions $(s_t, a_t, r_t^{\text{ext}}, s_{t+1})$ using the current decentralized policies.
  Update the forward model f_ϕ on real one-step transitions with $\mathcal{L}_{\text{fm}}(\phi)$.
  for each sampled transition and each source agent i **do**
    Construct the factual branch (a_t^i, a_t^{-i}, s_t).
    Sample K valid counterfactual source actions $\{a_t^{i,k}\}_{k=1}^K$ and construct $(a_t^{i,k}, a_t^{-i}, s_t)$.
    Roll out the factual and counterfactual branches with f_ϕ for H steps using the closed-loop procedure.
    Extract normalized teammate features from the predicted centralized states.
    Compute $d_{j,h}^{(k)}$, aggregate over k, teammates, and horizons, and obtain $c_i(t)$ after scaling and clipping.
    Compute $A_{\text{team}}^{\text{ext}}(t)$ using the extrinsic value estimate and form $\kappa(t)$.
    Compute $r_{i,t}^{\text{int}} = \lambda_{\text{int}} \kappa(t) c_i(t)$ and $r_{i,t}^{\text{total}} = r_t^{\text{ext}} + r_{i,t}^{\text{int}}$.
  end for
  Update the CTDE backbone using $r_{i,t}^{\text{total}}$ with the same actor-critic or policy-gradient update rule as the underlying backbone.
end for

A.7 Execution-Time Behavior and Computational Cost

MAGIC adds computation only during training. During execution, each agent uses the same decentralized policy as the underlying CTDE backbone. The forward model, counterfactual branches, action effect computation, extrinsic advantage gate, and intrinsic reward are not used at execution time.

The main additional training cost comes from model rollouts. For a batch with B sampled transitions, N agents, K counterfactual branches per source agent, and horizon H, the rollout cost scales as $O(BNKH)$ forward model steps, up to constants from feature extraction and distance computation. The cost is controlled by using a moderate horizon and a fixed number of counterfactual branches. For very large agent populations, MAGIC can be applied with sparse source-agent or teammate subsets. Instead of computing the action effect score for every source agent in every sampled transition, one can sample a subset of source agents per batch. Similarly, the teammate aggregation in Eq. (16) can be

restricted to a local neighborhood or communication graph, replacing the average over all $j \neq i$ with an average over a subset $\mathcal{N}(i)$. This changes the computation from all-pair action effect estimation to local action effect estimation while keeping the same factual and counterfactual branch construction. We do not use this approximation in the main experiments because the evaluated benchmarks have moderate agent counts, but it is a natural scaling option for swarm settings. In our measurements, the training overhead is about 10% relative to SCIC under matched settings, while execution-time cost is unchanged.

A.8 Theoretical Properties of the Action Effect Module

This subsection gives the formal properties referenced in Section 3. The analysis is stated for the action effect estimator used in the main text. It is stated directly for the factual and counterfactual branch-difference estimator used by MAGIC. We first analyze the unnormalized score $\tilde{c}_i(t)$, then state what remains true after the bounded normalization and clipping map used to obtain $c_i(t)$, and finally analyze the extrinsic advantage gate.

Notation. For a fixed source agent i and time t, let $\hat{s}^f_{t+h}$ and $\hat{s}^k_{t+h}$ denote the model-predicted factual and k-th counterfactual centralized states at horizon h. The empirical pairwise branch difference is

$$\hat{d}^{(k)}_{j,h} = \mathrm{dist}\left(z_j(\hat{s}^f_{t+h}), z_j(\hat{s}^k_{t+h})\right), \tag{23}$$

where $z_j(\cdot)$ extracts normalized teammate features and dist is the Euclidean distance in the normalized feature space in our implementation. The unnormalized empirical score is

$$\hat{\tilde{c}}_i(t) = \sum_{h=1}^{H} w_h \frac{1}{N-1} \sum_{j \neq i} \frac{1}{K} \sum_{k=1}^{K} \hat{d}^{(k)}_{j,h}, \qquad w_h \geq 0, \quad \sum_{h=1}^{H} w_h = 1. \tag{24}$$

When discussing approximation error, we use the same notation without hats, s^f_{t+h}, s^k_{t+h}, $d^{(k)}_{j,h}$, and $\tilde{c}^\star_i(t)$, for the corresponding quantities under the true environment dynamics and the same branch construction.

Proposition A.1 (Basic properties of the unnormalized score)**.** *Assume $w_h \geq 0$ and $\sum_{h=1}^{H} w_h = 1$, and assume $\mathrm{dist}(x, y) \geq 0$ with equality if and only if $x = y$. Then the unnormalized empirical action effect score satisfies:*

1. $\hat{\tilde{c}}_i(t) \geq 0$.
2. $\hat{\tilde{c}}_i(t) = 0$ *if and only if every positive-weight factual and counterfactual teammate feature difference is zero, i.e., for all h with $w_h > 0$, all $j \neq i$, and all k, we have $z_j(\hat{s}^f_{t+h}) = z_j(\hat{s}^k_{t+h})$.*
3. *If there exists a horizon h with $w_h > 0$, a teammate $j \neq i$, and a counterfactual branch k such that $z_j(\hat{s}^f_{t+h}) \neq z_j(\hat{s}^k_{t+h})$, then $\hat{\tilde{c}}_i(t) > 0$.*

Proof. Each pairwise difference $\hat{d}^{(k)}_{j,h}$ is nonnegative by the definition of the distance. The score $\hat{\tilde{c}}_i(t)$ is a nonnegative weighted sum of these nonnegative terms, which proves the first claim. The sum can be zero only if every term with positive coefficient is zero. Since $w_h > 0$, $1/(N-1) > 0$, and $1/K > 0$, this is equivalent to $\hat{d}^{(k)}_{j,h} = 0$ for every positive-weight horizon, teammate, and counterfactual branch. By the identity-of-indiscernibles property of the distance, this holds if and only if the corresponding teammate features are identical. The third claim is the contrapositive of the zero-score characterization. □

Corollary A.2 (Reduction to one-step action effect estimation)**.** *If $H = 1$ and $w_1 = 1$, then MAGIC reduces to a one-step action effect estimator:*

$$\hat{\tilde{c}}_i(t) = \frac{1}{N-1} \sum_{j \neq i} \frac{1}{K} \sum_{k=1}^{K} \mathrm{dist}\left(z_j(\hat{s}^f_{t+1}), z_j(\hat{s}^k_{t+1})\right). \tag{25}$$

It only measures the effect of replacing the source action on teammate features at the next predicted step.

Proof. Substituting $H = 1$ and $w_1 = 1$ into the definition of $\hat{\tilde{c}}_i(t)$ removes all horizons except $h = 1$, yielding the stated expression. □

Proposition A.3 (One-step blindness to delayed action effects)**.** *Consider a fixed decision context and source agent i. Suppose there exists an integer $d > 1$ such that, for all teammates $j \neq i$ and all counterfactual branches k,*

$$z_j(s^f_{t+1}) = z_j(s^k_{t+1}), \tag{26}$$

but for some teammate $j^\star \neq i$ and some counterfactual branch $k^\star$,

$$z_{j^\star}(s^f_{t+d}) \neq z_{j^\star}(s^{k^\star}_{t+d}). \tag{27}$$

Then the one-step action effect score is zero. In contrast, any multi-step score with $H \geq d$ and $w_d > 0$ is strictly positive under the true dynamics.

Proof. For $H = 1$, all teammate feature differences at $t + 1$ are zero by assumption, so Corollary A.2 and Proposition A.1 imply that the one-step score is zero. If $H \geq d$ and $w_d > 0$, then the difference at horizon d for teammate $j^\star$ and branch $k^\star$ is nonzero. Proposition A.1 then implies that the multi-step score is strictly positive. □

Forward model approximation error. The preceding properties describe the branch differences produced by a given rollout process. We now relate the learned-model score to the score that would be obtained under the true dynamics with the same factual and counterfactual branch construction. This statement explains why rollout errors are relevant to the reliability diagnostics in Table 3. For the error-bound analysis, we consider deterministic environment dynamics or branch comparisons coupled with the same exogenous randomness. Equivalently, the bounds can be read conditionally on the sampled exogenous noise.

Proposition A.4 (Forward model error bound for action effect scores)**.** *Assume each teammate feature extractor z_j is L_z-Lipschitz with respect to the centralized state norm, i.e.,*

$$\|z_j(s) - z_j(s')\|_2 \leq L_z \|s - s'\|_2 \quad \textit{for all } j. \tag{28}$$

Let the model rollout errors at horizon h be

$$\epsilon^f_h = \|\hat{s}^f_{t+h} - s^f_{t+h}\|_2, \qquad \epsilon^k_h = \|\hat{s}^k_{t+h} - s^k_{t+h}\|_2. \tag{29}$$

Then each pairwise branch-difference error is bounded by

$$\left|\hat{d}^{(k)}_{j,h} - d^{(k)}_{j,h}\right| \leq L_z \left(\epsilon^f_h + \epsilon^k_h\right), \tag{30}$$

and the aggregated score error satisfies

$$\left|\hat{\tilde{c}}_i(t) - \tilde{c}^\star_i(t)\right| \leq L_z \sum_{h=1}^{H} w_h \frac{1}{K} \sum_{k=1}^{K} \left(\epsilon^f_h + \epsilon^k_h\right). \tag{31}$$

Proof. For Euclidean distance, the reverse triangle inequality gives

$$\left|\hat{d}^{(k)}_{j,h} - d^{(k)}_{j,h}\right| \leq \left\|z_j(\hat{s}^f_{t+h}) - z_j(s^f_{t+h})\right\|_2 + \left\|z_j(\hat{s}^k_{t+h}) - z_j(s^k_{t+h})\right\|_2. \tag{32}$$

The Lipschitz property of z_j then gives

$$\left|\hat{d}^{(k)}_{j,h} - d^{(k)}_{j,h}\right| \leq L_z \epsilon^f_h + L_z \epsilon^k_h. \tag{33}$$

Substituting this bound into the weighted average defining the score gives

$$\left|\hat{\tilde{c}}_i(t) - \tilde{c}^\star_i(t)\right| \leq \sum_{h=1}^{H} w_h \frac{1}{N-1} \sum_{j \neq i} \frac{1}{K} \sum_{k=1}^{K} \left|\hat{d}^{(k)}_{j,h} - d^{(k)}_{j,h}\right| \tag{34}$$

$$\leq L_z \sum_{h=1}^{H} w_h \frac{1}{K} \sum_{k=1}^{K} \left(\epsilon^f_h + \epsilon^k_h\right), \tag{35}$$

because the bound does not depend on j. This proves the claim. □

A.9 Normalization, Clipping, and Reward-Level Properties

The structural properties above are stated for the unnormalized score $\tilde{c}_i(t)$ or its learned-model estimate $\hat{\tilde{c}}_i(t)$. In implementation, MAGIC uses a bounded transformation to obtain the reward-level score $c_i(t)$. In the main implementation this transformation is

$$c_i(t) = \psi(\hat{\tilde{c}}_i(t)), \qquad \psi(x) = \mathrm{clip}\left(\frac{x}{\sigma_c + \epsilon}, 0, c_{\max}\right), \tag{36}$$

where σ_c is a running scale estimate, $\epsilon > 0$, and $c_{\max}$ is the clipping threshold. This transformation is used for numerical stability. It is not part of the definition of the underlying factual and counterfactual action effect.

Proposition A.5 (Properties preserved by normalization and clipping)**.** *Let ψ be defined as in Equation* (36)*. Then:*

1. $0 \le c_i(t) \le c_{\max}$ *for all* i, t.
2. ψ *is monotone nondecreasing. Therefore, if* $x_1 \le x_2$*, then* $\psi(x_1) \le \psi(x_2)$*. Clipping may make two large scores equal after saturation, but it does not reverse their order.*
3. ψ *is globally Lipschitz with constant* $L_\psi = 1/(\sigma_c + \epsilon)$*. Hence*

$$|\psi(x) - \psi(y)| \le L_\psi |x - y|. \tag{37}$$

Proof. The first claim follows directly from clipping to $[0, c_{\max}]$. The map $x \mapsto x/(\sigma_c + \epsilon)$ is monotone increasing because $\sigma_c + \epsilon > 0$, and clipping to an interval is monotone nondecreasing, so their composition is monotone nondecreasing. The clipping map is 1-Lipschitz, and the scaling map has Lipschitz constant $1/(\sigma_c + \epsilon)$; the composition is therefore $1/(\sigma_c + \epsilon)$-Lipschitz. □

Corollary A.6 (Score-error propagation after scaling)**.** *Under the assumptions of Proposition A.4, the normalized and clipped score error satisfies*

$$\left|\psi(\hat{\tilde{c}}_i(t)) - \psi(\tilde{c}_i^\star(t))\right| \le L_\psi L_z \sum_{h=1}^{H} w_h \frac{1}{K} \sum_{k=1}^{K} \left(\epsilon_h^f + \epsilon_h^k\right). \tag{38}$$

Proof. Apply the Lipschitz bound in Proposition A.5 to the raw score error bound in Proposition A.4. □

Remark A.7 (Scope of the zero-effect statement). The exact zero-effect characterization in Proposition A.1 is a structural statement about the unnormalized branch-difference score. The implemented score $c_i(t) = \psi(\hat{\tilde{c}}_i(t))$ remains nonnegative and bounded. Because $\psi(0) = 0$, identical factual and counterfactual teammate futures still produce zero intrinsic action effect score. However, due to clipping, very large but different raw scores can map to the same saturated value. We therefore use the structural properties to characterize the underlying estimator, and use Proposition A.5 and Corollary A.6 to characterize the normalized reward-level score.

A.10 Properties of the Extrinsic Advantage Gate

The gate uses the extrinsic team advantage defined in the main text. Let

$$u(t) = \frac{A_{\mathrm{team}}^{\mathrm{ext}}(t) - \mu_A}{\sigma_A + \epsilon} \tag{39}$$

be the normalized gate input used in implementation, and let

$$\kappa(t) = g(u(t)), \qquad 0 \le g(u) \le 1, \tag{40}$$

where g is monotone nondecreasing. For fixed running statistics, $u(t)$ is monotone in the raw extrinsic team advantage, so larger extrinsic team advantage gives a larger gate. The same $\kappa(t)$ is shared by all agents at time t, while the action effect score $c_i(t)$ remains source-agent specific.

Proposition A.8 (Bounded task-alignment multiplier)**.** *Assume* $\lambda_{\text{int}} \geq 0$*,* $0 \leq \kappa(t) \leq 1$*, and* $0 \leq c_i(t) \leq c_{\max}$*. Then the intrinsic reward*

$$r_{i,t}^{\text{int}} = \lambda_{\text{int}} \kappa(t) c_i(t) \tag{41}$$

satisfies

$$0 \leq r_{i,t}^{\text{int}} \leq \lambda_{\text{int}} c_{\max}. \tag{42}$$

For a fixed $c_i(t)$*,* $r_{i,t}^{\text{int}}$ *is monotone nondecreasing in the gate input* $u(t)$*, and therefore monotone nondecreasing in* $A_{\text{team}}^{\text{ext}}(t)$ *for fixed running statistics.*

Proof. The bound follows immediately from multiplying the inequalities $0 \leq \kappa(t) \leq 1$ and $0 \leq c_i(t) \leq c_{\max}$ by $\lambda_{\text{int}} \geq 0$. If $c_i(t)$ is fixed, then $r_{i,t}^{\text{int}}$ is an affine nonnegative multiple of $\kappa(t)$. Since $\kappa(t) = g(u(t))$ and g is monotone nondecreasing, the intrinsic reward is monotone nondecreasing in $u(t)$. Because $u(t)$ is a positive affine transformation of $A_{\text{team}}^{\text{ext}}(t)$ for fixed running statistics, the same monotonicity holds with respect to the extrinsic team advantage. □

Proposition A.9 (Controlled contribution from low-gate transitions)**.** *Let* $\mathcal{L}$ *be any set of transitions whose normalized gate input satisfies* $u(t) \leq \rho$*. Define* $\epsilon_\rho = \sup_{u \leq \rho} g(u)$*. Then for any transition in* $\mathcal{L}$*,*

$$r_{i,t}^{\text{int}} \leq \lambda_{\text{int}} \epsilon_\rho c_{\max}.$$

In particular, transitions assigned a small gate can contribute only a controlled amount of intrinsic reward even when their action effect score is large.

Proof. For any transition in $\mathcal{L}$, we have $u(t) \leq \rho$. By the definition of ϵ_ρ, $\kappa(t) = g(u(t)) \leq \epsilon_\rho$. Combining this with $c_i(t) \leq c_{\max}$ gives the bound. □

Proposition A.10 (Second-moment control by gating)**.** *Let* $R_i^{\text{ungated}}(t) = \lambda_{\text{int}} c_i(t)$ *and* $R_i^{\text{gated}}(t) = \lambda_{\text{int}} \kappa(t) c_i(t)$*. If* $0 \leq \kappa(t) \leq 1$*, then*

$$\mathbb{E}\left[(R_i^{\text{gated}}(t))^2\right] \leq \mathbb{E}\left[(R_i^{\text{ungated}}(t))^2\right]. \tag{43}$$

Proof. For every transition, $|R_i^{\text{gated}}(t)| = \kappa(t)|R_i^{\text{ungated}}(t)| \leq |R_i^{\text{ungated}}(t)|$. Squaring both sides and taking expectations proves the claim. □

Proposition A.11 (Bounded perturbation of the extrinsic objective)**.** *Consider the discounted shaped objective obtained by adding the intrinsic reward to the extrinsic team reward. Suppose the extrinsic objective* J_{ext} *has a unique maximizer* $\pi^\star$ *with margin* $\Delta > 0$*, meaning*

$$J_{\text{ext}}(\pi^\star) \geq J_{\text{ext}}(\pi) + \Delta \quad \textit{for all } \pi \neq \pi^\star. \tag{44}$$

If $0 \leq c_i(t) \leq c_{\max}$ *and* $0 \leq \kappa(t) \leq 1$*, then the total discounted intrinsic return over* N *agents is bounded by*

$$0 \leq J_{\text{int}}(\pi) \leq \frac{N \lambda_{\text{int}} c_{\max}}{1 - \gamma}. \tag{45}$$

Therefore, if

$$\frac{N \lambda_{\text{int}} c_{\max}}{1 - \gamma} < \Delta, \tag{46}$$

then $\pi^\star$ *remains the unique maximizer of* $J_{\text{ext}} + J_{\text{int}}$*.*

Proof. The per-agent bound in Proposition A.8 gives $0 \leq r_{i,t}^{\text{int}} \leq \lambda_{\text{int}} c_{\max}$. Summing over N agents and over discounted time gives

$$0 \leq J_{\text{int}}(\pi) \leq \sum_{t=0}^{\infty} \gamma^t N \lambda_{\text{int}} c_{\max} = \frac{N \lambda_{\text{int}} c_{\max}}{1 - \gamma}. \tag{47}$$

For any $\pi \neq \pi^\star$,

$$(J_{\text{ext}} + J_{\text{int}})(\pi^\star) - (J_{\text{ext}} + J_{\text{int}})(\pi) \geq \Delta - J_{\text{int}}(\pi) \tag{48}$$

$$\geq \Delta - \frac{N \lambda_{\text{int}} c_{\max}}{1 - \gamma}. \tag{49}$$

The right-hand side is positive under the stated condition, so no policy $\pi \neq \pi^\star$ can overtake $\pi^\star$ under the shaped objective. □

Remark on objective shift. The intrinsic reward used by MAGIC is a training-time shaping signal, so the shaped training objective is not identical to the original extrinsic objective. Proposition A.11 should be read as a bounded-perturbation result rather than a general convergence guarantee for function-approximation MARL. It shows that the intrinsic term cannot arbitrarily dominate the extrinsic objective when λ_{int} and $c_{\max}$ are fixed, and that the original optimum is preserved whenever the extrinsic margin is larger than the maximum discounted intrinsic perturbation. In the main experiments, this theoretical control is paired with the extrinsic advantage gate, which further suppresses action effect rewards on low-advantage transitions.

B Additional Experimental Details and Results

This appendix provides the experimental details and additional results referenced by Section 4. We first describe environments, baselines, architectures, training protocols, evaluation metrics, and compute. We then provide the full tables supporting the main experimental claims.

B.1 Environment Details and Reward Protocols

The MPE experiments use three PettingZoo MPE tasks: Predator Prey, Cooperative Navigation, and Cooperative Competitive. All MPE tasks use five learning agents and an episode length of 25 steps unless otherwise stated. Predator Prey evaluates cooperative pursuit, Cooperative Navigation evaluates landmark coverage and collision avoidance, and Cooperative Competitive evaluates mixed cooperative-competitive interaction. We use the default environment dynamics and reward functions of the benchmark implementation. The team return is computed from the environment rewards and is used as the extrinsic reward for all methods. For the mixed cooperative-competitive MPE task, we follow the cooperative-team evaluation protocol and compute the reported team return over the controlled cooperative agents; the same scalar team reward is used as r_t^{ext} for all compared methods.

The SMAC and SMACv2 experiments use six micromanagement maps: `3s5z`, `5m_vs_6m`, `corridor`, `6h_vs_8z`, `MMM2`, and `Protoss5v5`. These tasks evaluate coordination under partial observability, discrete actions, action masking, and delayed team rewards. We use the standard win rate as the main metric. For all benchmark groups, MAGIC and the baselines are evaluated under the same environment reward. The intrinsic reward is used only as an additional training signal.

Existing assets and licenses. We use existing benchmark environments and software frameworks only for academic research comparisons. The MPE experiments use the PettingZoo implementation of MPE [Terry et al., 2021]. PettingZoo is publicly released by the Farama Foundation, with Farama-owned code released under the MIT license and included third-party components under their corresponding MIT or Apache-2.0 licenses. The SMAC and SMACv2 experiments use the public StarCraft micromanagement benchmark interfaces [Samvelyan et al., 2019, Ellis et al., 2023]. SMACv2 is released under the MIT license, and the PyMARL2 codebase used for the MAPPO-based protocol is released under the Apache-2.0 license. StarCraft II is used only as a third-party simulator through the benchmark interface, and we do not redistribute any StarCraft II game assets. All baseline methods and benchmark assets are cited in the paper and are used under their released research terms.

Table 4: MPE task configurations used in our experiments.

Task	Agents	Team structure	Episode length
Predator Prey	5	cooperative predators	25
Cooperative Navigation	5	fully cooperative	25
Cooperative Competitive	5	mixed cooperative-competitive	25

B.2 Baselines and Fairness Protocol

Within each benchmark group, all compared methods use matched training budgets, random seeds, observation settings, and evaluation schedules. On MPE, all methods are implemented on the same MADDPG-based CTDE backbone. MADDPG is the plain backbone without intrinsic rewards. SI, PMIC, SCIC, and MAGIC use the same environment interaction budget and the same replay-buffer training schedule. No intrinsic-reward method receives extra environment interaction.

On SMAC and SMACv2, MAGIC is evaluated with a MAPPO-based CTDE protocol implemented in PyMARL2. MAPPO is included as the plain policy-gradient backbone. QMIX is included as a representative value-decomposition baseline. PMIC, SCIC, and GradPS are evaluated under matched observation spaces, action masks, training budgets, and evaluation procedures whenever applicable. The purpose of this protocol is to compare the coordination signal rather than changes in environment access or evaluation budget. For baseline-specific hyperparameters, we use the official or recommended settings when available, while keeping the shared backbone, training budget, observation space, action masks, and evaluation protocol matched within each benchmark group.

B.3 Network Architectures and Hyperparameters

For MPE, actors use two hidden layers of size $(128, 128)$ with ReLU activations and a final tanh squashing layer for bounded continuous actions. The centralized critic and the extrinsic value network use two hidden layers of size $(256, 256)$. The extrinsic value network is trained only from environment rewards and supplies V_ω^{ext} for the gate. The forward model is a two-layer MLP with hidden sizes $(256, 256)$ and ReLU activations. It maps the current centralized state and joint action to the next centralized state. MAGIC does not use an additional mutual-information estimator in the action effect module.

For SMAC and SMACv2, we use the default MAPPO configuration from PyMARL2. The recurrent actor uses a one-layer GRU with hidden dimension 64, and we keep the default centralized value architecture and PPO update settings. The SMAC/SMACv2 forward model uses the same two-layer MLP architecture with hidden sizes $(256, 256)$ and ReLU activations. Its input is the centralized state concatenated with one-hot joint actions, and its output is the next centralized state. MAGIC introduces no modifications to the backbone policy network, action masking, or PPO objective. The only value-side addition is the extrinsic-only value head used for the gate. MAGIC adds only the training-time action effect module: a forward model, counterfactual action-replacement branches, teammate-future feature differences, action effect aggregation, and the extrinsic team-advantage gate. The policy used at execution is unchanged from the MAPPO backbone.

Table 5: Main hyperparameters for the MADDPG-based MPE experiments.

Hyperparameter	Value
Discount factor γ	0.95
Episode length	25
Total environment steps per task	4×10^6
Optimizer	Adam
Learning rate for actor, critic, and auxiliary networks	1×10^{-3}
Replay buffer size	10^6 transitions
Mini-batch size	1024
Target-network update	Polyak, $\tau = 0.01$
Gradient clipping	global norm 5.0
Main rollout horizon H	3
Number of counterfactual branches K	64
Intrinsic weight λ_{int}	0.05
Number of random seeds	5
Evaluation episodes per checkpoint	10
Evaluation policy	deterministic

For SMAC and SMACv2, we use StarCraft II version 4.10.0 and a unified MAPPO protocol. The shared settings are Adam with learning rate 5×10^{-4}, 8 parallel workers, 3200 collected transitions per update cycle, 5 PPO epochs, PPO clipping ratio 0.2, and GAE parameter $\lambda = 0.95$. Each method is trained for 10 million environment steps without map-specific tuning. Evaluation is performed every 10,000 environment steps using 32 deterministic test episodes.

Table 6 lists the MAGIC-specific hyperparameters used in the main experiments. We keep the intrinsic-reward hyperparameters fixed across tasks and vary them only in the diagnostic studies explicitly reported below. The $1.00\times$ forward model update ratio in Table 3 corresponds to this main setting. The ratios $0.25\times$, $0.50\times$, $0.75\times$, and $1.00\times$ correspond to 2, 5, 8, and 10 forward model epochs per RL iteration, respectively.

Table 6: Core MAGIC-specific hyperparameters used in the main experiments. The same intrinsic-reward hyperparameters are used across benchmark groups unless the entry is tied to the action space. No task-specific tuning of H, K, λ_{int}, $c_{\max}$, or the gate temperature is used.

Hyperparameter	Value
Intrinsic reward weight λ_{int}	0.05
Main rollout horizon H	3
Number of counterfactual branches K	64
Horizon weights w_h	Uniform
Score clipping threshold $c_{\max}$	5.0
Score normalization ϵ	10^{-5}
Advantage normalization ϵ	10^{-5}
Running-stat update rule	Batch EMA with momentum 0.99
Running-stat pooling scope	Training-batch transitions; source agents pooled for σ_c
Gate function $g(\bar{A}^{\text{ext}}_{\text{team}})$	$\sigma(\bar{A}^{\text{ext}}_{\text{team}}/\tau)$
Gate temperature τ	1.0
Forward model updates per RL iteration	10 epochs
Teammate-feature distance	Euclidean distance in normalized feature space
Discrete counterfactual sampling (SMAC)	Uniform over valid alternatives, excluding the executed action when possible
Continuous counterfactual sampling (MPE)	Uniform over valid bounds $[-1, 1]^d$

B.4 Training Protocols

For MPE, the MADDPG-based experiments use an off-policy CTDE training loop. At each environment step, the learner stores $(s_t, a_t, r^{\text{ext}}_t, s_{t+1})$ in a shared replay buffer. The actor-critic backbone, extrinsic value estimate, and forward model are updated from minibatches sampled from this buffer. The extrinsic value estimate is updated from one-step TD targets that use only r^{ext}_t. MAGIC computes the intrinsic reward from sampled transitions using the learned forward model and the factual and counterfactual branch procedure described in Appendix A. Each agent critic target uses the corresponding agent-specific shaped reward $r^{\text{total}}_{i,t}$. This computation does not require additional environment interaction.

For SMAC and SMACv2, the MAPPO-based experiments are on-policy. MAGIC computes intrinsic rewards inside the MAPPO training batches before the policy and value updates. The intrinsic reward is added to the extrinsic team reward for each source agent separately, while the gate itself is computed from the extrinsic team advantage. PPO advantages and value targets are computed from the resulting agent-specific shaped rewards. The extrinsic-only value head used by the gate is updated from extrinsic-return targets and does not receive intrinsic rewards. The forward model is trained on one-step transitions from the same rollout batches. The MAGIC module is removed during evaluation and execution.

B.5 Evaluation Metrics, Smoothing, and Statistical Testing

For MPE, the main metric is episodic team return. For SMAC and SMACv2, the main metric is win rate. We report means over five random seeds. When a table reports standard deviation, it is computed across the five seed-level scalar values.

The scalar metrics are computed as follows. Final performance is the average of the last $K_{\text{eval}} = 10$ evaluation points of a training run, using the same K_{eval} across methods within a benchmark group. Best performance is the maximum evaluation value observed over training. AUC is the area under the evaluation curve, normalized by the training horizon. The learning curves in the main text are smoothed only for visualization, and the scalar results are computed from the per-seed evaluation sequences.

For the MPE scalar tables, we report paired two-sided t-tests comparing each baseline against MAGIC using matched seed indices. These p-values are descriptive statistics and are not corrected for multiple comparisons. They are used to indicate whether the final-return differences are consistent across seeds.

B.6 Compute Resources and Runtime

All runtime measurements are conducted under the same software stack and hardware setup used for the corresponding benchmark group. The experiments are run on a workstation equipped with an NVIDIA RTX 3090 GPU. For each runtime comparison, we use the same environment steps, random seeds, evaluation schedule, and backbone configuration as in the corresponding performance experiment. We report representative GPU-hours for one seed on one task or map; total compute scales approximately linearly with the number of seeds and evaluated tasks.

MAGIC adds computation only during training. The additional cost comes from constructing counterfactual branches, rolling them out with the learned forward model, extracting teammate future features, and aggregating factual and counterfactual differences. In the main setting, we use a moderate rollout horizon $H = 3$ and $K = 64$ counterfactual branches. During execution, MAGIC uses the same decentralized policies as the underlying CTDE backbone and does not use the forward model, counterfactual branches, action effect computation, or advantage gate. Therefore, MAGIC introduces no additional execution-time overhead.

Under matched settings, MAGIC increases training time by approximately 10% relative to SCIC. For context, SCIC is approximately 15% slower than MADDPG in our MPE implementation and approximately 25% slower than MAPPO in our SMAC/SMACv2 implementation. Thus, MAGIC has a higher training-time cost than standard CTDE backbones, but this overhead is limited to training and is accompanied by substantial performance gains. Table 7 summarizes the runtime-performance trade-off.

Table 7: Representative compute cost and performance trade-off. GPU-hours are measured for one seed on one task or map using an NVIDIA RTX 3090, with 4M environment steps for MPE and 10M environment steps for SMAC/SMACv2. Execution-time overhead is zero because MAGIC uses only the decentralized backbone policy during execution. For MPE, percentage gains are normalized by the magnitude of the reference final return because some MPE tasks have negative returns.

Benchmark group	MAGIC GPU-hours	Reference	Training overhead	Performance gain
MPE	3.9	SCIC	$\approx 10\%$	$+17.5\%$ to $+36.4\%$ final return
SMAC/SMACv2	11.4	SCIC	$\approx 10\%$	$+2.6$ to $+14.1$ Win% points
MPE	3.9	MADDPG	$\approx 26.5\%$	$+68.9\%$ Predator Prey final return
SMAC/SMACv2	11.4	MAPPO	$\approx 37.5\%$	$+12.7$ average Win% points

The reported GPU-hours are representative single-seed measurements for one task or map. Since all main results use five random seeds, the total reported compute for a benchmark group is approximately five times the single-seed cost times the number of evaluated tasks or maps, plus the corresponding baseline runs.

B.7 Full MPE Results

Table 8 reports the full scalar metrics for the three MPE tasks. These results correspond to the learning curves in Figure 3. Table 9 reports the 10-agent Predator Prey setting.

Table 8: Full results on the MPE benchmark over five random seeds.

	Predator Prey					Cooperative Navigation					Cooperative Competitive				
Method	Final ↑	Std ↓	Best ↑	AUC ↑	p-val	Final ↑	Std ↓	Best ↑	AUC ↑	p-val	Final ↑	Std ↓	Best ↑	AUC ↑	p-val
MADDPG	34.1	0.87	35.7	28.4	< 0.001	-24.5	1.07	-24.2	-28.3	< 0.001	-2.3	0.17	-1.6	-3.1	< 0.001
SI	42.9	0.59	43.9	30.5	< 0.001	-23.9	0.46	-23.7	-27.5	< 0.001	-1.3	0.12	-0.5	-1.6	< 0.001
PMIC	44.6	0.69	45.8	39.8	< 0.001	-25.2	0.83	-24.6	-28.5	< 0.001	-1.2	0.14	-0.8	-1.4	< 0.001
SCIC	45.4	1.05	47.9	34.8	< 0.001	-22.8	0.52	-21.7	-25.9	< 0.001	-1.1	0.23	-0.4	-1.4	< 0.001
MAGIC	**57.6**	0.75	**62.1**	**48.1**	–	**-18.8**	0.46	**-18.4**	**-21.3**	–	**-0.7**	0.08	**-0.1**	**-1.2**	–

Table 9: Results on the 10-agent cooperative Predator Prey task over five random seeds.

Method	Final ↑	Std ↓	Best ↑	AUC ↑	p-val
MADDPG	15.3	1.12	17.1	11.5	< 0.001
SI	26.8	0.94	28.4	21.2	< 0.001
PMIC	31.5	0.88	33.2	27.6	< 0.001
SCIC	34.9	1.37	39.5	26.9	< 0.001
MAGIC	**51.2**	0.82	**56.3**	**41.8**	–

B.8 Full SMAC and SMACv2 Results

Tables 10 and 11 provide standard deviations for the final win-rate and AUC results reported in Table 1.

Table 10: Final win rates on SMAC and SMACv2 over five random seeds.

Method	3s5z	5m_vs_6m	corridor	6h_vs_8z	MMM2	Protoss5v5	Avg.
QMIX	92.4±2.6	78.5±3.8	48.2±5.1	58.6±4.5	74.2±4.2	36.8±5.5	64.8
MAPPO	94.1±2.2	90.5±2.8	54.8±5.3	71.2±3.9	81.4±3.5	47.5±5.8	73.3
PMIC	89.3±3.1	83.7±4.2	61.2±4.8	65.8±4.6	72.1±4.4	44.3±5.1	69.4
GradPS	94.8±2.0	91.2±2.6	64.5±4.5	74.0±3.8	81.8±3.5	49.5±5.2	76.0
SCIC	95.2±1.8	91.8±2.5	72.6±4.1	76.5±3.4	80.3±3.1	52.4±4.8	78.1
MAGIC	**97.8**±1.5	**95.6**±2.1	**83.4**±3.2	**85.2**±2.8	**87.3**±2.4	**66.5**±3.6	**86.0**

Table 11: AUC of win-rate learning curves on SMAC and SMACv2 over five random seeds.

Method	3s5z	5m_vs_6m	corridor	6h_vs_8z	MMM2	Protoss5v5	Avg.
QMIX	78.4±2.1	52.3±3.0	25.1±3.8	32.5±3.2	42.1±3.5	15.4±3.6	41.0
MAPPO	80.2±1.8	68.5±2.2	30.6±4.1	44.1±2.8	51.5±2.6	22.3±4.2	49.5
PMIC	72.1±2.5	58.4±3.5	35.2±3.6	38.6±3.5	41.8±3.2	19.5±3.8	44.3
GradPS	81.5±1.6	70.1±2.0	38.4±3.5	46.8±2.9	53.2±2.8	24.1±4.0	52.4
SCIC	82.3±1.5	71.4±1.9	45.8±3.1	49.2±2.5	51.8±2.4	26.5±3.5	54.5
MAGIC	**86.5**±1.2	**78.2**±1.6	**55.6**±2.5	**59.4**±2.2	**62.3**±1.9	**38.1**±2.8	**63.4**

B.9 Full Component Analysis Results

This section provides the detailed results corresponding to Table 2. Table 12 reports the MPE module ablation on Predator Prey. Table 13 reports component attribution results on representative SMAC maps with standard deviations over five random seeds.

Table 12: MPE module ablation on Predator Prey over five random seeds.

Method	Final ↑	Best ↑	AUC ↑	p-val
MAGIC	57.6 ± 0.8	62.1	48.1	–
MAGIC w/o Advantage Gating	47.3 ± 0.9	48.9	38.6	< 0.001
MAGIC $H = 1$ Action Effect	41.8 ± 0.7	43.4	37.5	< 0.001

Table 13: Component attribution on representative SMAC maps over five random seeds.

Map	Method	Win% ↑	Std ↓
corridor	MAPPO	54.8	±5.3
	MAGIC $H = 1$+Gate	73.1	±3.8
	MAGIC $H = 3$ w/o Gate	77.5	±3.6
	MAGIC $H = 3$	**83.4**	±3.2
5m_vs_6m	MAPPO	90.5	±2.8
	MAGIC $H = 1$+Gate	92.4	±2.3
	MAGIC $H = 3$ w/o Gate	91.8	±3.9
	MAGIC $H = 3$	**95.6**	±2.1
MMM2	MAPPO	81.4	±3.5
	MAGIC $H = 1$+Gate	82.1	±3.4
	MAGIC $H = 3$ w/o Gate	83.8	±3.7
	MAGIC $H = 3$	**87.3**	±2.4

B.10 Comparison with Agent-Specific Gate Variants

We further compare the shared team-advantage gate with an agent-specific counterfactual-Q gate. The agent-specific gate uses an extrinsic centralized Q estimate to compare the executed joint action with counterfactual replacements of the source agent's action, giving each source agent its own gate value. This comparison tests whether the shared gate is only a simplification or a useful task-level filter.

For this variant, we compute

$$A_i^Q(t) = Q^{\text{ext}}(s_t, a_t) - \frac{1}{K}\sum_{k=1}^{K} Q^{\text{ext}}(s_t, (a_t^{i,k}, a_t^{-i})),$$

and use $\kappa_i^Q(t) = \sigma(\bar{A}_i^Q(t)/\tau)$ as the agent-specific gate. The extrinsic Q estimate is trained only with environment rewards, and we use the same K, normalization rule, and gate temperature as the main MAGIC setting.

Table 14 shows that the agent-specific gate improves over the no-gate variant on both Predator Prey and corridor, but it does not improve over the shared team-advantage gate. This supports our design choice. Source specificity is already provided by the action effect score, while the shared gate gives a stable task-level filter.

Table 14: Comparison between the shared team-advantage gate and an agent-specific counterfactual-Q gate. Results are averaged over five seeds. The agent-specific gate improves over the no-gate variant, but it does not improve over the shared team-advantage gate. This supports the use of a shared task-level filter, while source specificity is already provided by the action effect score.

Gate variant	Predator Prey Final Return	corridor Win%
No gate	47.3 ± 0.9	77.5 ± 3.6
Agent-specific counterfactual-Q gate	55.9 ± 2.6	80.5 ± 3.9
Shared team-advantage gate (MAGIC)	$\mathbf{57.6 \pm 0.8}$	$\mathbf{83.4 \pm 3.2}$

B.11 Full Reliability Diagnostics

This section provides the detailed diagnostics corresponding to Table 3. In-MSE measures prediction error on normal policy rollouts. Int-MSE measures prediction error under action-replacement rollouts. Sep. AUC measures separability of branch effects, namely whether model-predicted teammate-future differences reflect the true strength of action effects.

Table 15: Fixed-horizon reliability on `5m_vs_6m` with $H = 3$ under different forward model update ratios.

Update ratio	In-MSE	Int-MSE	Sep. AUC	Win%
$0.25\times$	0.135	0.228	0.57	86.8
$0.50\times$	0.072	0.108	0.74	91.2
$0.75\times$	0.038	0.051	0.87	94.3
$1.00\times$	0.025	0.032	0.91	**95.6**

Table 16: Full horizon sensitivity and rollout-error diagnostics.

Map	H	In-MSE	Int-MSE	Sep. AUC	Win%
corridor	1	0.012	0.015	0.94	73.1 ± 3.8
	2	0.022	0.028	0.92	78.6 ± 3.4
	3	0.038	0.046	0.90	83.4 ± 3.2
	5	0.075	0.098	0.82	80.5 ± 3.6
	8	0.165	0.230	0.58	68.4 ± 4.8
	10	0.245	0.360	0.49	62.5 ± 5.4
5m_vs_6m	1	0.008	0.010	0.95	92.4 ± 2.3
	2	0.015	0.019	0.93	94.1 ± 2.4
	3	0.025	0.032	0.91	95.6 ± 2.1
	5	0.056	0.078	0.84	93.8 ± 2.5
	8	0.138	0.205	0.59	82.5 ± 4.3
	10	0.210	0.325	0.48	75.3 ± 5.1
MMM2	1	0.010	0.013	0.94	82.1 ± 3.4
	2	0.018	0.024	0.92	85.6 ± 2.8
	3	0.030	0.038	0.89	87.3 ± 2.4
	5	0.065	0.088	0.81	84.5 ± 3.1
	8	0.150	0.220	0.58	73.2 ± 4.5
	10	0.230	0.350	0.49	66.8 ± 5.3

Table 17: Robustness to explicit forward model corruption on `5m_vs_6m` with $H = 3$.

Noise level	Int-MSE	Sep. AUC	Win%
0.0	0.032	0.91	**95.6**
0.1	0.055	0.88	94.2
0.5	0.120	0.79	91.5
1.0	0.280	0.55	82.4

B.12 MPE Horizon Sensitivity

Table 18 reports the horizon sensitivity of MAGIC on MPE Predator Prey. The result follows the same rise-then-decline pattern as the SMAC horizon study in the main text.

Table 18: Sensitivity of MAGIC to horizon H on MPE Predator Prey.

H	Final ↑	Best ↑	AUC ↑	p-val
1	41.8 ± 0.7	43.4	37.5	< 0.001
2	55.0 ± 0.5	57.8	44.7	< 0.001
3	$\mathbf{57.6 \pm 0.8}$	**62.1**	**48.1**	–
5	54.4 ± 0.8	55.9	47.7	< 0.001
8	53.6 ± 0.4	55.1	43.5	< 0.001
10	44.6 ± 0.1	45.8	42.4	< 0.001

B.13 Sensitivity to the Number of Counterfactual Branches

Table 19 studies the number of counterfactual branches on MPE Predator Prey, a continuous-control task. Very small K gives a noisy action effect estimate and lower separability of branch effects. Increasing K improves both final return and Sep. AUC, but the gains saturate around $K = 32$–64. Increasing K to 128 gives only a marginal additional improvement. These results indicate that MAGIC does not rely on dense search over the continuous action space. A moderate number of counterfactual branches is sufficient to estimate useful action effects.

Table 19: Sensitivity to the number of counterfactual branches K on MPE Predator Prey. Results are averaged over five seeds. Performance and separability of branch effects improve from very small K and empirically saturate around $K = 32$–64.

Branches K	Final return	Sep. AUC
$K = 4$	48.2 ± 4.7	0.61
$K = 8$	52.1 ± 3.9	0.73
$K = 16$	55.8 ± 2.5	0.83
$K = 32$	57.1 ± 1.8	0.87
$K = 64$ (main)	57.6 ± 0.8	0.89
$K = 128$	$\mathbf{57.8 \pm 1.5}$	**0.90**

B.14 Robustness to Stochastic Action Execution

To test whether the same reliability trend also appears outside MPE, we add a stochastic action-execution stress test on the SMAC corridor map. At each environment step, each agent's selected action is replaced with a uniformly sampled valid action with probability p_{slip}. The random action is sampled from the valid action set given by the action mask. The same action-slip setting is used during training and evaluation.

Table 20 shows that MAGIC remains stronger than MAPPO and SCIC under low and medium action stochasticity. As p_{slip} increases, Sep. AUC decreases, and the performance gap becomes smaller. This follows the same reliability pattern as the MPE stochasticity study. When separability of branch effects becomes weak, the benefit of multi-step action effect estimation is reduced.

Table 20: Robustness to stochastic action execution on SMAC corridor. Each selected action is replaced by a uniformly sampled valid action with probability p_{slip}. Results are averaged over five seeds. As stochasticity increases, separability of branch effects decreases. MAGIC remains strongest under low and medium action stochasticity, and stays close to the one-step influence baseline when separability becomes weak.

p_{slip}	MAPPO	SCIC	MAGIC	Sep. AUC
0.00	54.8 ± 5.3	72.6 ± 4.1	$\mathbf{83.4 \pm 3.2}$	0.90
0.05	48.3 ± 6.1	64.1 ± 5.5	$\mathbf{75.8 \pm 4.4}$	0.85
0.10	41.2 ± 6.8	51.5 ± 7.2	$\mathbf{62.3 \pm 5.6}$	0.76
0.20	30.5 ± 8.2	32.8 ± 7.9	$\mathbf{34.2 \pm 7.4}$	0.58

B.15 Robustness to Transition Stochasticity

To test the robustness of the action effect estimator beyond deterministic dynamics, we inject zero-mean Gaussian noise with standard deviation σ_{noise} into the physical position and velocity components after each environment step, followed by the same state clipping used by the environment. The same stochastic transition setting is used during training and evaluation. Table 21 reports the results.

MAGIC remains stronger than MADDPG and SCIC under low and medium stochasticity. As the noise level increases, Sep. AUC decreases, showing that separability of branch effects becomes harder to preserve. Under severe noise, Sep. AUC approaches random separation and MAGIC no longer provides a clear gain over the one-step influence baseline. However, the ungated variant collapses below the backbone, while the gated variant remains close to MADDPG and SCIC. This supports the

role of the shared advantage gate as a conservative task-alignment filter. When the estimated action effect signal becomes unreliable, the gate reduces the damage from noisy intrinsic rewards.

Table 21: Robustness to transition stochasticity and the role of the advantage gate on MPE Predator Prey. We inject zero-mean Gaussian noise with standard deviation σ_{noise} into the physical position and velocity components after each environment step. MAGIC remains effective under low and medium stochasticity. When severe noise destroys separability of branch effects, the ungated variant degrades sharply, while the gated variant remains close to the backbone and the one-step influence baseline.

Noise level	MADDPG	SCIC	MAGIC w/o Gate	MAGIC	Sep. AUC
$\sigma_{\text{noise}} = 0.00$	34.1 ± 0.9	45.4 ± 1.1	47.3 ± 0.9	$\mathbf{57.6 \pm 0.8}$	0.89
$\sigma_{\text{noise}} = 0.05$	32.8 ± 2.4	43.1 ± 3.0	49.5 ± 3.8	$\mathbf{55.2 \pm 2.1}$	0.84
$\sigma_{\text{noise}} = 0.15$	28.5 ± 3.1	36.7 ± 4.2	39.4 ± 4.7	$\mathbf{46.3 \pm 3.8}$	0.72
$\sigma_{\text{noise}} = 0.30$	21.4 ± 4.5	23.2 ± 5.1	14.2 ± 6.8	22.8 ± 6.2	0.54

B.16 Artificial Delay Analysis

Table 22 studies how the useful horizon changes when additional reward delay is injected into the corridor map. The best horizon shifts toward larger values as delay increases, but the benefit saturates once longer rollouts accumulate excessive model error. The delay-0 MAPPO value is kept consistent with the corridor MAPPO result in Table 1.

Table 22: Impact of artificial delay on corridor win rates.

Delay	MAPPO	$H = 1$	$H = 3$	$H = 5$	$H = 8$	$H = 10$
0	54.8	73.1	83.4	80.5	68.4	62.5
2	52.4	64.2	78.5	81.2	70.1	64.3
4	35.5	55.4	68.2	75.6	76.4	68.5
6	24.1	45.1	55.3	68.4	71.5	69.1

B.17 Boundary under Severe Delay

Table 23 reports a severe-delay setting on corridor with delay $d = 12$. Larger horizons still help relative to $H = 1$, but absolute performance remains low and eventually drops at $H = 10$. This marks a boundary of the current learned-rollout regime and shows why the rollout horizon should remain finite.

Table 23: Boundary breakdown on corridor with delay $d = 12$.

Method	H	Int-MSE	Sep. AUC	Win%
MAPPO	–	–	–	12.4
MAGIC	1	0.020	0.88	18.2
MAGIC	3	0.055	0.82	26.5
MAGIC	5	0.120	0.71	33.4
MAGIC	8	0.280	0.55	39.5
MAGIC	10	0.450	0.48	31.2

C Limitations

MAGIC estimates action effects with finite-horizon learned rollouts. This design is intended for settings where the learned model can preserve effect differences between branches over a moderate horizon. Our reliability diagnostics and horizon studies provide an empirical criterion for selecting this range, and the main experiments use $H = 3$.

MAGIC adds computation during training because it evaluates counterfactual branches. It introduces no execution-time overhead because the module is removed after training. For larger agent popula-

tions, the same estimator can be combined with source-agent subsampling or neighborhood-based teammate aggregation to reduce rollout cost.

This work uses point-estimate forward rollouts and distances between predicted teammate features. The stochasticity studies in Appendix B.14 and Appendix B.15 show that MAGIC remains useful when separability of branch effects is preserved. Future work can extend this design with probabilistic or ensemble forward models for settings with stronger transition uncertainty.

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 state the proposed multi-step advantage-gated action-effect framework and the experimental scope; the claims are supported by Sections 3 and 4.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Appendix C discusses limitations related to learned forward-model rollouts, finite rollout horizons, training-time computation, and experimental scope.

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification: The assumptions, statements, and proofs for the action-effect score, forward-model error bound, normalization/clipping, and advantage gate are provided in Appendix A.8–A.10.

   Guidelines:

   - The answer [N/A] 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**

   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 algorithm, network architectures, hyperparameters, evaluation metrics, and statistical testing protocol are described in the method section and appendix.

   Guidelines:

   - The answer [N/A] 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: We include an anonymized supplementary package with the main method implementation. The paper provides the remaining details needed to understand and reproduce the experimental setup, including the algorithm, architectures, hyperparameters, evaluation protocol, and compute information.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: The training setup, environments, optimization details, and evaluation procedure are specified in the experiments section and appendix.

   Guidelines:

   - The answer [N/A] 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 experiments use five random seeds. The paper reports standard deviations or shaded bands for the main results, and Appendix B.5 describes the scalar metrics and statistical testing procedure.

Guidelines:

- The answer [N/A] 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**

   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: Appendix B.6 reports the hardware setup, runtime comparison protocol, training-time overhead, and execution-time cost. MAGIC adds computation only during training and introduces no additional execution-time overhead.

   Guidelines:

   - The answer [N/A] 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: The work is a methodological study on cooperative MARL and does not involve human subjects, private data, or prohibited uses.

   Guidelines:

   - The answer [N/A] 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: [Yes]

    Justification: The paper includes an impact statement and discusses the work as a general machine learning method for MARL coordination.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a dataset, foundation model, or deployed system that requires additional safeguards.

    Guidelines:

    - The answer [N/A] 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: Appendix B.1 lists the existing benchmark environments and software frameworks used in the experiments, cites the corresponding papers, and states the relevant released licenses or usage terms where applicable.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The paper does not introduce a new dataset, benchmark suite, or model artifact as a reusable asset.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The paper does not involve crowdsourcing or human-subject data collection.

Guidelines:

- The answer [N/A] 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: [N/A]

    Justification: The paper does not involve human participants, so participant risks and institutional review are not applicable.

    Guidelines:

    - The answer [N/A] 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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The proposed method and experiments do not use LLMs as an original or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.