ADVERSARIALLY ROBUST MULTI-AGENT RL VIA TEST-TIME IMAGINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in model-free multi-agent reinforcement learning (MARL) have improved policy robustness against adversarial observation perturbations at test time. However, training such robust policies requires extensive interaction with the environment. Multi-agent world models could reduce the interaction cost by enabling robust policy learning from imagined experience. Multi-agent world models, however, typically condition their predictions on global information unavailable to individual agents at execution time, making them unsuitable for decentralized defense. Moreover, policies trained from imagined experience are not inherently robust to adversarial observation perturbations. We propose MATRIX, a world-model-based MARL framework that unifies sample-efficient policy learning with training- and test-time mechanisms for robustness to test-time adversarial observations. MATRIX applies classifier-free guidance to a diffusion-based world model that predicts each agent's next observation using either local information alone or using local and global information jointly. This design enables MATRIX to use a single world model for three complementary functions: training policies on model-generated trajectories, robustifying policies using model-generated trajectories, and predicting observations from local information for decentralized test-time defense. Building on locally conditioned observation prediction, we propose a low-overhead execution protocol that uses the trained world model to mitigate the effects of adversarial observation perturbations at test time, without requiring a separate defense model. Evaluations in environments with discrete and continuous action spaces demonstrate that MATRIX outperforms state-of-the-art baselines in both sample efficiency and robustness to test-time observation attacks.

1 INTRODUCTION

Model-free multi-agent reinforcement learning (MARL) has enabled effective coordination in applications ranging from robotic teams to autonomous control (Nayak et al., 2023; Wei et al., 2019; Rashid et al., 2020; Yu et al., 2022). Despite these advances, learning effective policies typically requires extensive interaction with the environment. Moreover, the resulting policies are highly vulnerable to test-time adversarial attacks, in which manipulated observations can disrupt agents' decisions and degrade team performance (Lin et al., 2020). Improving robustness against such attacks can further increase the number of environment interactions, as agents must learn to perform under a broad range of adversarial conditions (Kazari & Dán, 2026).

World models offer a promising means of improving sample-efficiency in single agent reinforcement learning (RL) and MARL by allowing policies to learn from imagined experience (Hafner et al., 2025; Zhang et al., 2024). However, their potential to improve robustness against adversarial attacks remains largely unexplored in MARL, where existing defenses are predominantly model-free (Bukharin et al., 2023; Zhou et al., 2025; Kazari & Dán, 2026). This motivates us to investigate whether the predictive capabilities of world models can support not only efficient policy learning but also robust decision-making at test time.

In single-agent RL, predictive models have been used as a defense mechanism to detect suspicious observations and replace them with predicted or corrected estimates (Lin et al., 2017; Yang & Xu, 2024). Extending this defense to MARL requires a world model that each agent can use independently during execution. However, existing approaches based on centralized training with decentral-

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

ized execution (CTDE) typically condition their world models on global information and use them only for centralized policy training (Zhang et al., 2026; 2024). Because global information is unavailable to individual agents at test time, the world models cannot support decentralized observation prediction and correction.

One approach to enabling decentralized observation prediction would be to train an independent world model for each agent. However, doing so would forgo centralized modeling of agent interactions, potentially impairing coordination and policy quality. Another approach is to retain a centralized world model for policy learning and train additional agent-specific models for deployment. While preserving joint modeling during training, this approach introduces additional models and training procedures, thereby eroding the sample efficiency offered by model-based learning.

Even when decentralized prediction is available, prediction errors can alter an agent's actions and propagate through interactions with other agents, potentially turning small discrepancies into team miscoordination. Moreover, reconstruction-based detection requires a complete observation prediction at every step, even when no attack occurs (Haider et al., 2023; Lin et al., 2017). For diffusion world models, for instance, this involves multiple denoiser evaluations per prediction (Alonso et al., 2024), introducing computational overhead that can hinder real-time defense.

These limitations motivate our central research question: **How can a multi-agent world model be designed and trained to support both sample-efficient policy learning and low-overhead, robust decentralized execution under test-time observation attacks?** To address this question, we propose **M**ulti-**A**gent **T**raining with **R**obust **I**ndividual E**X**ecution (MATRIX). MATRIX is an architecture and a training protocol that enable sample-efficient training of MARL policies under the CTDE paradigm, while allowing the trained model to be used by each agent independently during test-time execution. This design eliminates the need to train separate decentralized models. It also provides a low-overhead execution protocol that leverages the decentralized predictions at test time to detect and mitigate adversarial attacks, without requiring additional training. Our main contributions are as follows:

(1) We adapt classifier-free guidance to multi-agent diffusion world models by selectively dropping global conditioning information while preserving each agent's local context. This design enables a single world model to support both globally and locally conditioned observation prediction, unifying sample-efficient policy learning and policy robustification in imagination with the predictive capabilities required for fully decentralized execution.

(2) We propose an attack-agnostic, low-overhead test-time defense protocol that reuses the trained world model without additional training. Each agent evaluates a denoising-loss-based anomaly score using a single denoiser forward pass and applies the CUSUM method to detect sustained deviations. Upon detecting an attack, an agent replaces the suspicious observation with a locally conditioned prediction from the same world model before selecting its action.

(3) We evaluate MATRIX in the StarCraft and MaMuJoCo environments covering both discrete and continuous action spaces. Our results show that MATRIX matches or exceeds the sample efficiency of state-of-the-art model-based MARL and achieves substantially higher performance than existing robust MARL methods under test-time observation attacks.

2 PRELIMINARIES

2.1 MARL MODEL

We consider a cooperative multi-agent task and model it as a decentralized partially observable Markov decision process (Dec-POMDP) (Oliehoek & Amato (2016)), $\mathcal{M} = \big(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, R, P, \{\Omega^i\}_{i\in\mathcal{N}}, \mathcal{O}, \mu, \gamma\big)$. $\mathcal{N} = \{1, \ldots, N\}$ is the set of agents, $\mathcal{S}$ is the global state space, and $\mathcal{A}^i$ and Ω^i are agent i's action and observation spaces, respectively, each discrete or continuous. At each step t, the joint observation $\mathbf{o}_t = (o_t^1, \ldots, o_t^N) \sim \mathcal{O}(\cdot \mid s_t)$ is drawn and agent i receives its own component o_t^i. Each agent's policy π^i conditions on its own observation history. The joint action $\mathbf{a}_t$ drives the transition $P(s_{t+1} \mid s_t, \mathbf{a}_t)$ and yields the shared reward $R(s_t, \mathbf{a}_t)$. The joint policy $\boldsymbol{\pi} = (\pi^1, \ldots, \pi^N)$ induces the value function $V^{\boldsymbol{\pi}}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t) \mid s_0 = s\right]$, and agents jointly maximize $V^{\boldsymbol{\pi}} = \mathbb{E}_{s_0\sim\mu}[V^{\boldsymbol{\pi}}(s_0)]$. We assume that the state s_t is unobservable, including during training; centralized training instead has access to the joint observation $\mathbf{o}_t$. This

assumption reflects practical settings in which agents can share sensor measurements during training but cannot directly observe the environment state.

Threat Model: We consider test-time adversarial observation perturbations, where an attacker modifies the observations received by a subset of agents $\mathcal{N}_{\mathrm{a}} \subseteq \mathcal{N}$ with $|\mathcal{N}_{\mathrm{a}}| = N_a$. An attacked agent receives $\tilde{o}_t^i = o_t^i + \delta_t^i$, where δ_t^i is constrained by a perturbation budget, $||\delta_t^i||_\infty \leq B$. The attacker aims to reduce the team's expected cumulative reward by disrupting action selection and coordination, without modifying the environment dynamics, the rewards or policy networks.

2.2 Diffusion Models and Classifier-Free Guidance

Diffusion Models: Diffusion models learn a data distribution by reversing a Gaussian corruption process. Following the EDM framework of (Karras et al. (2022)), let $y \sim p_{\mathrm{data}}$ denote a clean data sample. A noisy sample is constructed as $x = y + n$, where $n \sim \mathcal{N}(0, \sigma^2 I)$ and σ denotes the noise level. A denoiser $D_\theta(x, \sigma)$ is trained to recover y from x by minimizing $\mathcal{L}_{\mathrm{diff}}(\theta) = \mathbb{E}_{y,\sigma,n}\left[\lambda(\sigma)\, \|D_\theta(y+n,\sigma) - y\|_2^2\right]$. Here, $\lambda(\sigma)$ is a noise-dependent weighting function that balances the contribution of different noise levels to the training objective. EDM samples the training noise levels from a log-normal distribution and parametrizes the denoiser as

$$D_\theta(x,\sigma) = c_{\mathrm{skip}}(\sigma)x + c_{\mathrm{out}}(\sigma)F_\theta(c_{\mathrm{in}}(\sigma)x, c_{\mathrm{noise}}(\sigma))\,, \tag{1}$$

where F_θ is the underlying neural network and the noise-dependent preconditioning functions stabilize training across different noise levels. At inference, sampling begins from Gaussian noise and repeatedly applies the denoiser over a decreasing sequence of noise levels.

Classifier-Free Guidance: Classifier-free guidance (CFG) (Ho & Salimans (2022)) enables a single diffusion model to operate with and without conditioning information. During training, the condition c is randomly replaced by a null condition $\varnothing$, allowing the same network to learn both $D_\theta(x, \sigma; c)$ and $D_\theta(x, \sigma; \varnothing)$. At inference, the two predictions are combined as $D_{\mathrm{CFG}}(x, \sigma; c) = D_\theta(x, \sigma; \varnothing) + w\,[D_\theta(x, \sigma; c) - D_\theta(x, \sigma; \varnothing)]$. Thus, $w = 0$ gives the unconditional prediction, $w = 1$ gives the ordinary conditional prediction, and $w > 1$ strengthens the influence of the conditioning.

3 Multi-agent Training with Robust Individual Execution

Applying model-based observation defenses to MARL presents three main challenges. First, existing multi-agent world models typically depend on centralized information unavailable during decentralized execution. Second, model prediction errors can propagate through inter-agent dependencies and undermine coordination. Third, reconstruction-based attack detection can incur substantial test-time overhead. MATRIX addresses these challenges through a single diffusion world model that supports both global and agent-local conditioning, policy robustification through imagined experience, and a decentralized defense protocol that detects attacks using a single denoiser forward pass and generates replacement observations only when an attack is detected. We provide a more detailed discussion of these challenges and their relation to existing model-based defenses in Appendix A. We next present the design of MATRIX.

3.1 MATRIX: Structure and Training

3.1.1 Diffusion Model

A key architectural contribution of MATRIX is a CFG-inspired conditioning scheme tailored to the asymmetric information structure of CTDE (Ho & Salimans (2022)). Whereas standard CFG drops the entire conditioning signal, MATRIX selectively drops non-local information while preserving each agent's local context. This structured conditioning scheme enables a single, parameter-shared diffusion world model to learn both globally and locally conditioned observation prediction. The globally conditioned mode supports centralized imagination for sample-efficient policy learning, while the locally conditioned mode allows each agent to reuse the same model independently during decentralized execution.

Formally, let $h_t^i = (o_{t-H+1}^i, \ldots, o_t^i)$ denote agent i's observation history of length H. We define $\ell_t^i = (h_t^i, a_t^i)$ as the local conditioning information, and $g_t = (\mathbf{o}_{t-H+1:t}, \mathbf{a}_t)$ the global conditioning

information. Our objective is to learn both $p^i_\theta(o^i_{t+1} \mid \ell^i_t, g_t)$ and $p^i_\theta(o^i_{t+1} \mid \ell^i_t)$, using a single set of parameters θ. Unlike prior works such as MADiff (Zhu et al., 2024), our primary motivation for using CFG is not guidance itself, but rather its ability to enable a single model to generate predictions both with and without access to global information. Accordingly, we do not aim to learn the marginal observation distribution $p^i_\theta(o^i_{t+1})$.

In MATRIX we follow an EDM framework for the diffusion model and employ a denoiser $D^i_\theta(\tilde{o}^i_{t+1}, \sigma; \ell^i_t, g_t)$ for estimating agent i's next observation from a noisy candidate $\tilde{o}$. The model can operate using only local information by dropping the global conditioning, which we refer to as the *local mode*, denoted by $D^i_\theta(\tilde{o}^i_{t+1}, \sigma; \ell^i_t, \varnothing)$. In *global mode*, when both local and global conditioning information are available, the CFG-based inference of the denoiser is obtained by

$$D^i_\theta(\tilde{o}^i_{t+1}, \sigma; \ell^i_t, \varnothing) + w[D^i_\theta(\tilde{o}^i_{t+1}, \sigma; \ell^i_t, g_t) - D^i_\theta(\tilde{o}^i_{t+1}, \sigma; \ell^i_t, \varnothing)]. \tag{2}$$

The rationale behind this design choice is that the two modes can be used for different purposes. For policy learning in imagination we use global mode for centralized training of agents (Section 3.1.3). We use a combination of both modes for policy robustification in imagination (Section 3.1.4). Finally, at test time, agents can use the local mode to predict their own next observation when their real observations are attacked (Section 3.2).

Denoiser Training To train the denoiser using CFG, for each agent in a training sequence, we sample a binary conditioning mask m_i, with $\Pr(m_i = 1) = p_{\text{drop}}$. The mask is fixed for the entire sequence. We use $\widetilde{g}^i_t = g_t$ when $m_i = 0$ and $\widetilde{g}^i_t = \varnothing$ when $m_i = 1$. Accordingly, the denoising loss function will be

$$\mathcal{L}_{\text{den}}(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} \lambda(\sigma) \left\| D^i_\theta\big(\tilde{o}^i_{t+1}, \sigma; \ell^i_t, \widetilde{g}^i_t\big) - o^i_{t+1} \right\|_2^2\right], \tag{3}$$

where $\tilde{o}^i_{t+1} = o^i_{t+1} + \sigma\epsilon^i$ with $\epsilon^i \sim \mathcal{N}(0, I)$ is the noisy observation, σ is a sampled training noise level, $\lambda(\sigma)$ is the weight of different diffusion noise levels, and the expectation is over sequences from collected environment experience, conditioning masks, and diffusion noise.

3.1.2 Reward and Termination Prediction Model

The specific structure of MATRIX's observation prediction model necessitates a corresponding design for its reward and termination prediction model that differs from those used in existing multi-agent world models. Whereas MARIE (Zhang et al., 2024) uses a transformer to jointly predict observation tokens, rewards, and continuation from agent-wise token sequences augmented with global features, MATRIX handles high-dimensional observation generation using the diffusion model and uses a dedicated centralized transformer only for reward and termination prediction. This specialization enables a substantially smaller transformer and eliminates the need for a separate observation tokenizer. Instead, the transformer directly encodes continuous observation-action histories and aggregates the resulting agent representations, preserving centralized modeling of inter-agent dependencies while reducing the complexity associated with tokenizing and predicting observations.

Formally, for each agent i we encode $(h^i_t, a^i_{t-H+1:t})$ into a history token z^i_t. The tokens of all agents, together with a learned query token, form a block $B_t = (z^1_t, \ldots, z^N_t, q)$, which is fed to the transformer. The output at the query token summarizes the joint history and is passed to separate heads for the team reward $\hat{r}_t$ and continuation probability $\hat{c}_t$. We use a shared loss function, consisting of a regression term for the team reward and a cross-entropy term for the continuation, to jointly train the model. The loss function is given by $\mathcal{L}_{\text{r,e}}(\phi) = \mathbb{E}[\text{Huber}(\hat{r}_t - r_t) - (1 - d_t)\log\hat{c}_t - d_t\log(1 - \hat{c}_t)]$, where we use the Huber loss, ϕ denotes the set of parameters and d_t indicates episode termination.

3.1.3 MARL Policy Training in Imagination

MATRIX allows centralized training by learning agents policies from globally conditioned imagined trajectories; we train the agents using MAPPO (Yu et al., 2022) on trajectories generated by the global mode of the diffusion world model, capturing inter-agent dependencies without additional environment interaction. Let $\hat{o}^i_t$ denote the imagined observation of agent i, and let $P^{i,\text{G}}_\theta$ denote the

predictive distribution induced by the global mode according to (2). Agent i's action and its next observation are then obtained by

$$a_t^i \sim \pi_\psi^i(\cdot \mid \hat{o}_t^i), \qquad \hat{o}_{t+1}^i \sim P_\theta^{i,\mathrm{G}}(\cdot \mid \hat{\ell}_t^i, \hat{g}_t), \tag{4}$$

where π_ψ^i is the parametrized policy, and $\hat{\ell}_t^i$ and $\hat{g}_t$ are formed from the imagined histories and the selected actions. Note that the actor itself is memoryless.

Actor and critic: We use a multilayer perceptron network with parameters ψ as the policy network, and employ parameter sharing across the agents. The critic is a centralized transformer that estimates a single team value from the joint observation history $\hat{\mathbf{h}}_t = (\hat{h}_t^1, \ldots, \hat{h}_t^N)$: we embed each agent's history using a multilayer perceptron and apply self-attention across the agent representations and a learned value-query token, after which a scalar value head reads the resulting query representation.

3.1.4 Policy Robustification Against Local Prediction Errors in Imagination

Replacing an agent's observation with a model prediction can change its action and consequently affect other agents. To improve robustness to prediction errors during decentralized execution, MATRIX employs a mixed-mode imagination protocol. After initial global-mode policy learning, MATRIX continues to simulate the joint dynamics globally while providing some actors with locally predicted observations. This exposes the policies to local prediction errors and their effects on coordination while preserving the benefits of globally conditioned imagination.

During robustification, we maintain a global-mode observation $\hat{o}_t^i$, generated as in (4) and an actor observation u_t^i. At the start of an imagined rollout, each agent is independently selected with probability p_{loc}. Let $b_i \sim \mathrm{Bernoulli}(p_{\mathrm{loc}})$ denote this selection, which is retained throughout the rollout. For the selected agents, the actor input would be a predicted observation using the local mode of the diffusion model. Thus, denoting $h_t^{i,u} = (u_{t-H+1}^i, \ldots, u_t^i)$, the next actor input is

$$u_{t+1}^i = \begin{cases} \check{o}_{t+1}^i, & \check{o}_{t+1}^i \sim P_\theta^{i,\mathrm{L}}(\cdot \mid h_t^{i,u}, a_t^i), \quad b_i = 1, \\ \hat{o}_{t+1}^i, & b_i = 0, \end{cases} \tag{5}$$

where $P_\theta^{i,\mathrm{L}}$ denotes sampling with the local-mode denoiser. Selected agents thus repeatedly act based on their own locally predicted observations, while the remaining agents observe the globally simulated trajectory. The rationale behind this procedure is that, after sufficient training, the observations $\hat{o}_{t+1}^i$ predicted using global mode provide a good approximation of the observations generated by the real environment, while it also captures the test-time scenario in which some agents rely on locally predicted observations, and the remaining agents receive their true observations from the environment. By training over different subsets of selected agents, we shape the learned policies to maintain coordination even when some agents rely on local predictions.

We found that starting the robustification phase early during training can disrupt the learning of the other components. Therefore, we activate it only during the final α_{robust} fraction of the world-model training steps, by replacing the normal policy learning described in Section 3.1.3. During this phase, the critic continues to use the global-mode observation history, while the reward-termination model evaluates this history together with the actions actually selected by the actors. Figure 5 in Appendix C illustrates how the world model is used in different stages of training and execution.

3.2 Achieving Test Time Robustness in MATRIX

By predicting each agent's next observation from local information, MATRIX enables fully decentralized defense against observation perturbations. At test time, each agent uses a local copy of the trained denoiser to assess its incoming observation. If an attack is detected, the agent generates a locally conditioned replacement observation and uses it to select its action. Importantly, MATRIX is not trained against a specific attack. Instead, it treats adversarially perturbed observations as anomalies relative to the locally conditioned observation distribution learned by the world model.

Conventional model-based defenses assess each incoming observation by first reconstructing its expected value. Applying this approach to MATRIX would require iterative diffusion sampling at every time step, including when no attack is present. Instead, MATRIX uses a single-sample estimate of the local-mode denoising loss as its anomaly score. This score directly measures the

consistency of an observation with the locally conditioned dynamics learned during training and requires only one denoiser forward pass. Full diffusion sampling is therefore performed only after an anomaly is detected, substantially reducing the computational overhead of continuous monitoring.

Formally, after receiving o_t^i, agent i samples a noise standard deviation σ_t and Gaussian noise $\epsilon_t^i \sim \mathcal{N}(0, I)$, following the training noise distribution. It then constructs $\tilde{o}_t^i = o_t^i + \sigma_t \epsilon_t^i$, and computes

$$l_t^i = \lambda(\sigma_t) \left\| D_\theta^i\big(\tilde{o}_t^i, \sigma_t; \ell_{t-1}^i, \varnothing\big) - o_t^i \right\|_2^2 . \tag{6}$$

The local context ℓ_{t-1}^i contains the observation history used by the world model, including any predicted observations provided to the actor previously. Given this context, the denoising loss measures how well the current o_t^i agrees with the locally conditioned observation distribution learned by the world model. A benign observation is expected to yield a low loss, whereas an adversarially perturbed observation may be inconsistent with the predicted dynamics and thus yield a higher loss.

We apply a one-sided CUSUM detector to the loss sequence (Page, 1954), which is a well-established method for quickest detection of a change in the mean of a sequence of random variables. That is, we compute $S_t^i = \max\{0, S_{t-1}^i + l_t^i - \kappa\}$, with $S_{\text{init}}^i = 0$, where κ is a bias term. Whenever $S_t^i \geq \eta > 0$, the agent generates a replacement observation by diffusion sampling in local mode, conditioned on ℓ_{t-1}^i, and uses it for selecting its action. The agent continues to update the CUSUM statistic using newly received observations and resumes using them as policy inputs once $S_t^i < \eta$. Algorithm 1 in the Appendix C summarizes this procedure for one agent.

4 Related Works

Robust MARL methods address adversarial perturbations via policy robustification, including ROMANCE and BATPAL (Yuan et al., 2023; Kazari & Dán, 2026), and observation-focused adversarial training or regularization, including ATSA and ERNIE (Zhou et al., 2025; Bukharin et al., 2023). These approaches improve robustness without exploiting imagined experience for sample-efficient learning. In single-agent RL, predictive models support attack detection and observation replacement (Lin et al., 2017; Xiong et al., 2022), as well as out-of-distribution detection (Haider et al., 2023). DMBP (Yang & Xu, 2024) uses conditional diffusion to recover observations for offline-trained policies. MATRIX brings model-based defense to fully decentralized MARL while reusing the world model employed for policy learning and robustification.

World models improve sample efficiency by enabling policy learning in imagination. MAMBA (Egorov & Shpilman, 2022) uses communicating recurrent models, MARIE (Zhang et al., 2024) combines local autoregressive transformers with centralized information aggregation, and MABL (Venugopal et al., 2023) incorporates global information into hierarchical latent representations supporting decentralized policy execution. Diffusion-based approaches include DIAMOND (Alonso et al., 2024) for single-agent learning and DIMA (Zhang et al., 2026), which predicts global state transitions by sequentially conditioning on agents' actions. MADiff (Zhu et al., 2024) instead learns a diffusion trajectory model from offline data and derives actions through inverse dynamics, supporting both centralized and decentralized planning. Its CFG steers trajectory generation using attributes such as desired returns. MATRIX uses conditioning dropout for a different purpose: selectively removing global information while preserving local context, thereby jointly learning globally and locally conditioned predictions. This enables one world model to support online policy learning and robustification in imagination, together with decentralized test-time defense.

5 Experiments

We evaluate MATRIX on the StarCraft II Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) and MaMuJoCo (Peng et al., 2021), which feature discrete and continuous action spaces, respectively. Our evaluation considers both sample-efficient policy learning and test-time adversarial robustness, with comparisons against the corresponding state-of-the-art baselines. For SMAC, we consider the 2s3z, MMM, and 11m scenarios. For MaMuJoCo, we evaluate on Ant-4x2, HalfCheetah-6x1 (HC), and CoupledHalfCheetah-12x1 (CHC). In particular, MMM, 11m, and CHC, which involve 10, 11, and 12 agents, respectively, are deliberately included to assess the scalability of MATRIX to scenarios with many agents. Performance is evaluated using the win rate for SMAC and the

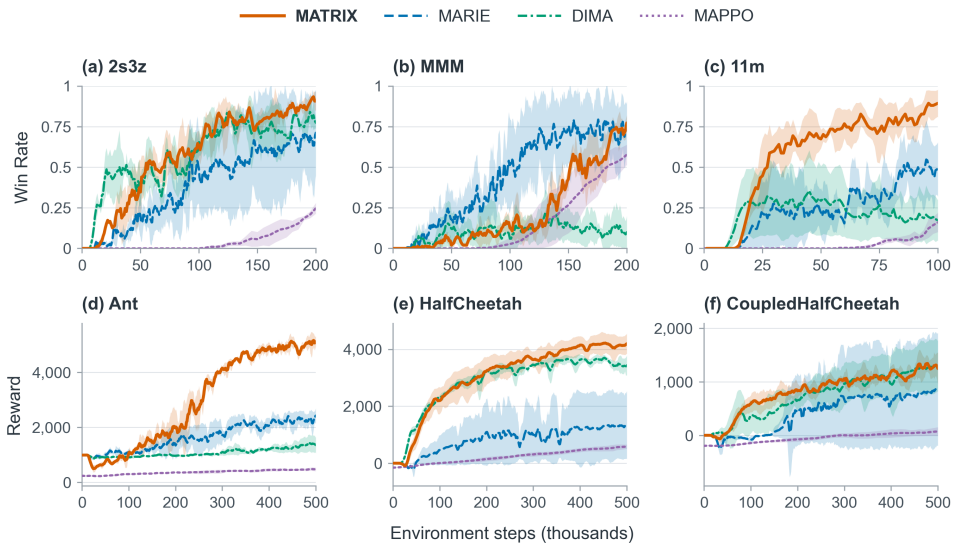


Figure 1: Learning curves for MATRIX and the baselines in SMAC (2s3z, MMM, and 11m) and MaMuJoCo (Ant, HC, CHC) using three seeds per run.

total episodic return for MaMuJoCo. In all scenarios we set $\alpha_{robust} = 0.1$. Other implementation details and hyperparameters can be found in the Appendix C.

Baselines for Sample Efficiency: We consider MARIE Zhang et al. (2024) and DIMA Zhang et al. (2026) as world-model baselines for evaluating sample-efficient policy learning. We additionally include MAPPO Yu et al. (2022) as a model-free baseline. DIMA assumes access to environment-provided global state information, which gives it an unfair advantage over the other approaches. To ensure a fair comparison, we instead provide DIMA with the joint observations of all agents.

Baselines for Test-time Robustness: We compare MATRIX against ATSA (Zhou et al., 2025), PAAD (Sun et al., 2021), and BATPAL Kazari & Dán (2026), which are model-free robust MARL approaches. We are not aware of model-based robust MARL methods directly comparable to MATRIX, we thus construct world-model-based variants of ATSA and PAAD. Specifically, we freeze a pretrained multi-agent world model and use it to generate imagined trajectories on which the robustification algorithms are subsequently trained. We refer to the resulting methods as WM-ATSA and WM-PAAD, respectively. For each scenario, we use the world-model baseline that achieved the best policy-learning performance as the underlying world model. Accordingly, DIMA is used for HC and CHC, while MARIE is used for other scenarios.

Test-Time Adversarial Attacks: We consider FGSM (Goodfellow et al., 2014) and MAD (Zhang et al., 2020) as non-targeted attacks, and the adversaries trained by ATSA and PAAD as targeted attacks for our test-time robustness evaluation. We additionally consider random Gaussian perturbations (RAND) as another attack baseline. Following the probabilistic attack-timing setting considered by Lin et al. (2017), we apply RAND perturbations independently at each time step with probability 0.9, allowing occasional unperturbed observations during evaluation.

5.1 Sample-Efficient Learning

Figure 1 shows the learning curves for all scenarios in the low-sample regime. MATRIX achieves the highest average performance in five of the six scenarios. In MMM, its final performance after 200K environment steps is comparable to that of MARIE. These results show that equipping the world model with the locally conditioned prediction capabilities required for decentralized defense does not compromise sample-efficient policy learning. In particular, MATRIX is generally at least as sample-efficient as existing multi-agent world models while simultaneously providing the capabilities required for robust test-time execution. This result is particularly encouraging considering that MATRIX achieves such performance with considerably lower complexity than the competing baselines, both in terms of model size and wall-clock training time (see Appendix D.1).

Table 1: Test-time robustness with attack budget $B = 0.25$. Left: win rate in SMAC scenarios; right: return on MaMuJoCo scenarios. **Bold**/underline: best/second best per scenario and condition. The real environment sample budget in parentheses.

Method	Clean	RAND	FGSM	MAD	ATSA	PAAD	Clean	RAND	FGSM	MAD	ATSA	PAAD
	2s3z (200K)						**Ant** (500K)					
MATRIX	**0.92**	**0.81**	**0.83**	**0.83**	**0.78**	**0.84**	**5267**	**3449**	**3539**	**3546**	**3537**	**3479**
ATSA	0.50	0.51	0.44	0.54	0.49	0.49	458	459	530	487	513	409
PAAD	0.35	0.36	0.35	0.34	0.31	0.34	420	431	541	430	525	525
BATPAL	0.27	0.23	0.08	0.27	0.14	0.06	460	468	532	499	508	489
WM-ATSA	0.03	0.05	0.03	0.04	0.07	0.07	528	532	599	529	555	556
WM-PAAD	0.01	0.01	0.05	0.02	0.02	0.03	−429	−436	−286	−434	−366	−355
	MMM (200K)						**HalfCheetah** (500K)					
MATRIX	0.72	0.65	0.70	0.66	**0.69**	**0.65**	**4386**	**2638**	**2353**	**2498**	**2760**	**2627**
ATSA	0.29	0.37	0.27	0.32	0.20	0.20	−105	−105	−94	−105	−104	−107
PAAD	0.72	0.79	**0.72**	0.72	0.67	0.53	−327	−327	−321	−327	−325	−329
BATPAL	**0.83**	**0.82**	0.57	**0.79**	0.51	0.53	6	3	−4	2	3	3
WM-ATSA	0.00	0.00	0.00	0.00	0.00	0.00	−567	−567	−564	−567	−565	−566
WM-PAAD	0.09	0.10	0.11	0.11	0.09	0.05	−530	−530	−523	−530	−527	−530
	11m (100K)						**CoupledHalfCheetah** (500K)					
MATRIX	**0.75**	**0.75**	**0.63**	**0.72**	**0.55**	**0.65**	**1169**	**1072**	**1142**	**1179**	**1063**	**1149**
ATSA	0.31	0.41	0.33	0.31	0.24	0.31	−161	−161	−155	−161	−160	−157
PAAD	0.56	0.55	0.53	0.53	0.45	0.50	−593	−593	−584	−593	−593	−593
BATPAL	0.04	0.03	0.02	0.02	0.02	0.03	125	121	90	122	112	128
WM-ATSA	0.00	0.00	0.00	0.00	0.00	0.00	−351	−355	−345	−351	−355	−357
WM-PAAD	0.00	0.00	0.00	0.00	0.00	0.00	−2	−2	−2	−2	−2	−3

5.2 Test-Time Adversarial Robustness

Table 1 shows the results of all robust methods against different adversarial attacks at test time for a perturbation budget $B = 0.25$ and one (randomly selected) victim agent, $N_a = 1$. We provide results for other budgets and number of attacked agents in Appendix E.2. For each scenario, all methods had the same budget of real environment samples. This means that model free approaches are trained using the same number of samples as the specified budget. For WM-ATSA, and WM-PAAD this budget is used for training the world model, and the robust algorithms are trained on 2M imaginary steps in the world model. Each result is the average over 200 episodes.

The results show that, in most scenarios, MATRIX achieves considerably higher performance against all types of attacks. We observe that the model-free robust baselines exhibit relatively consistent performance across attacked and non-attacked settings, but they cannot achieve competitive performance in the low-sample regime considered in our experiments. MATRIX maintains similar performance across clean and attacked settings in the SMAC and CHC scenarios. Larger reductions are observed in Ant and HC, although MATRIX continues to substantially outperform the baselines in absolute return. However, the superior performance of MATRIX cannot be attributed solely to the low-sample regime or to its use of a world model. As shown in the Appendix E.1, DIMA and MARIE, two other world-model-based approaches, experience substantial performance degradation when exposed to test-time attacks.

A somewhat counterintuitive observation is that WM-ATSA and WM-PAAD achieve considerably lower performance than their model-free counterparts, despite being trained on a much larger number of trajectories, albeit imagined ones. This result reveals an important limitation of using an existing world model as a substitute for the real environment: increasing the amount of imagined experience does not compensate for model bias when the world model and the robustification algorithm induce different trajectory distributions. Similar forms of policy-induced distribution shift and model bias are well documented in model-based RL, where a model trained under one state-action visitation distribution can become inaccurate when used to optimize a different policy (Janner et al., 2019; Wang et al., 2023). In our setting, the world model is trained on trajectories collected by the policy learned jointly with it and is therefore most accurate in regions visited by that policy. Ro-

Table 2: Ablation study on (a) test-time defense and (b) policy robustification

(a) Test-time defense

2s3z: win rate				
Variant	Clean	FGSM	ATSA	Time (ms/step)
MATRIX	**0.92**	**0.83**	**0.78**	14.0
Prediction error	0.85	0.41	0.61	41.7
Direct denoise	0.35	0.13	0.13	41.1
No defense	**0.92**	0.37	0.56	4.3

(b) Policy robustification

2s3z: win rate			
Variant	Clean	FGSM	ATSA
MATRIX	**0.92**	**0.83**	**0.78**
w/o robustification	**0.92**	0.66	0.62
HalfCheetah: return			
MATRIX	**4386**	**2353**	**2760**
w/o robustification	4215	1238	1773

bustification algorithms such as ATSA may induce different observation histories, actions, and state visitation distributions, causing their imagined rollouts to enter regions poorly represented in the world model's training data. Prediction errors in these regions may accumulate or be exploited during optimization, limiting the effectiveness of training solely on imagined trajectories. We provide a detailed analysis in the Appendix B.

5.3 Ablation Study

Test-Time Protocol We evaluate the individual contribution of our test-time robustness protocol and compare it against model-based defense mechanisms proposed for RL Haider et al. (2023); Yang & Xu (2024). The first defense uses prediction error as the detection signal, following the approach of Haider et al. (2023) for single-agent RL. The second approach denoises every incoming observation using an assumed noise level before being passed to the policy (Yang & Xu, 2024), even if no attack is present. We also evaluate the test-time performance of policies trained with our model but executed without the proposed test-time robustness protocol. In addition to robustness, we measure the average per-step processing time of each approach on the same hardware to compare their wall-clock computational cost. To eliminate the latency associated with CPU-GPU transfers, we conduct this experiment entirely on the CPU. Table 2-(a) shows results for SMAC 2s3z without attack and under FGSM and ATSA attacks. Removing the test-time defense substantially degrades the agents' performance under both attacks, demonstrating the contribution of the proposed test-time protocol to robustness. Using the full prediction error as the detection signal also results in lower performance than using our proposed denoising-loss-based metric. We attribute this improvement to the training-inference alignment that motivates our choice of detection metric, as the same denoising objective used to train the world model is directly employed for anomaly detection at test time. More importantly, prediction-error-based detection and direct denoising introduce considerably higher per-step computational overhead than our method, as both require multiple denoising steps.

Policy Robustification in Training To evaluate the contribution of the policy robustification phase (Section 3.1.4) to test-time performance, we compare the robustness of MATRIX variants trained with and without this phase on 2s3z and HC. The results in Table 2-(b) show that the policy robustification phase substantially improves the performance of MATRIX under adversarial attacks.

6 Conclusion

We introduced MATRIX, a diffusion-based multi-agent world model that unifies sample-efficient policy learning with robust decentralized execution. By selectively dropping global conditioning through classifier-free guidance, a single model supports policy learning and robustification in imagination, as well as local observation prediction for test-time defense. Our denoising-loss-based detection protocol reuses this model with low computational overhead and no additional training. Experiments across discrete and continuous action spaces demonstrate strong sample efficiency and robustness under observation attacks, while ablations highlight the complementary contributions of policy robustification and test-time defense. We also observed that existing world models can generalize poorly to independently trained policies. A promising direction for future work is to incorporate policy robustification into the data-collection process for world-model training.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing. The ideation, methodology, and evaluations are done solely by the authors.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

Alexander Bukharin, Yan Li, Yue Yu, Qingru Zhang, Zhehui Chen, Simiao Zuo, Chao Zhang, Songan Zhang, and Tuo Zhao. Robust multi-agent reinforcement learning via adversarial regularization: Theoretical foundation and stable algorithms. *Advances in Neural Information Processing Systems*, 36:68121–68133, 2023.

Vladimir Egorov and Aleksei Shpilman. Scalable multi-agent model-based reinforcement learning. *arXiv preprint arXiv:2205.15023*, 2022.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

Tom Haider, Karsten Roscher, Felippe Schmoeller da Roza, and Stephan Günnemann. Out-of-distribution detection for reinforcement learning agents with probabilistic dynamics models. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 851–859, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=k7FuTOWMOc7.

Kiarash Kazari and György Dán. Bayesian robust cooperative multi-agent reinforcement learning against unknown adversaries. In *International Conference on Learning Representations*, volume 2026, pp. 31613–31639, 2026.

Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot. On the robustness of cooperative multi-agent reinforcement learning. In *Proc. of IEEE Security and Privacy Workshops (SPW)*, 2020.

Yen-Chen Lin, Ming-Yu Liu, Min Sun, and Jia-Bin Huang. Detecting adversarial attacks on neural network policies with visual foresight. *arXiv preprint arXiv:1710.00814*, 2017.

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

Frans A. Oliehoek and Chris Amato. A concise introduction to decentralized pomdps. In *SpringerBriefs in Intelligent Systems*, 2016. URL https://api.semanticscholar.org/CorpusID:3263887.

E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1–2):100–115, 1954. doi: 10.1093/biomet/41.1-2.100. URL https://doi.org/10.1093/biomet/41.1-2.100.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in neural information processing systems*, 34:12208–12221, 2021.

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

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

Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. *arXiv preprint arXiv:2106.05087*, 2021.

Aravind Venugopal, Stephanie Milani, Fei Fang, and Balaraman Ravindran. Mabl: Bi-level latent-variable world model for sample-efficient multi-agent reinforcement learning. *arXiv preprint arXiv:2304.06011*, 2023.

Xiyao Wang, Wichayaporn Wongkamjan, Ruonan Jia, and Furong Huang. Live in the moment: Learning dynamics model adapted to evolving policy. In *Proc. of International Conference on Machine Learning (ICML)*, volume 202, pp. 36470–36493. PMLR, 2023.

Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1913–1922, 2019.

Zikang Xiong, Joe Eappen, He Zhu, and Suresh Jagannathan. Defending observation attacks in deep reinforcement learning via detection and denoising. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 235–250. Springer, 2022.

Zhihe Yang and Yunjian Xu. Dmbp: Diffusion model-based predictor for robust offline reinforcement learning against state observation perturbations. In *International Conference on Learning Representations*, volume 2024, pp. 38549–38578, 2024.

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

Lei Yuan, Ziqian Zhang, Ke Xue, Hao Yin, Feng Chen, Cong Guan, Lihe Li, Chao Qian, and Yang Yu. Robust multi-agent coordination via evolutionary generation of auxiliary adversarial attackers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11753–11762, 2023.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in neural information processing systems*, 33:21024–21037, 2020.

Yang Zhang, Chenjia Bai, Bin Zhao, Junchi Yan, Xiu Li, and Xuelong Li. Decentralized transformers with centralized aggregation are sample-efficient multi-agent world models. *arXiv preprint arXiv:2406.15836*, 2024.

Yang Zhang, Xinran Li, Jianing Ye, Shuang Qiu, Delin Qu, Xiu Li, Chongjie Zhang, and Chenjia Bai. Revisiting multi-agent world modeling from a diffusion-inspired perspective. *Advances in Neural Information Processing Systems*, 38:65439–65470, 2026.

Ziyuan Zhou, Guanjun Liu, Mengchu Zhou, and Weiran Guo. Robust multi-agent reinforcement learning with stochastic adversary. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=bnhFueOeav.

Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. Madiff: Offline multi-agent learning with diffusion models. *Advances in Neural Information Processing Systems*, 37:4177–4206, 2024.

APPENDIX

A CHALLENGES OF MODEL-BASED DEFENSES IN MULTI-AGENT RL

In this section, we revisit single-agent model-based approaches for robustness against test-time adversarial perturbations and discuss the challenges associated with adapting them to MARL. The aim of MATRIX is to address these challenges in one modeling framework.

A common model-based approach to single-agent robustness is to use a trained model to predict the next observation and compare the predicted and actual observations, denoted by $\hat{o}_t$ and o_t, respectively, using a detection metric. If this comparison indicates an anomaly, the predicted observation is used instead of the actual observation for policy execution. For instance, (Haider et al. (2023)) uses $||o_t - \hat{o}_t||_2$ as the detection metric, while (Lin et al. (2017)) measures the l_1 distance between the policy outputs for the two observations, i.e., $||\pi(\hat{o}_t) - \pi(o_t)||_1$. A closely related approach is proposed by (Yang & Xu (2024)), where a trained diffusion model is used to denoise the observation before it is provided to the policy.

Adapting these approaches to MARL introduces several challenges. First, existing multi-agent world models are typically designed such that their predictive capabilities depend on centralized information or components that are not available to individual agents during decentralized execution. For example, MARIE (Zhang et al. (2024)) uses aggregated features of the joint observations and actions of all agents as input to its transformer. DIMA (Zhang et al. (2026)) generates a joint state by sequentially conditioning on the actions of all agents, while MAMBA (Egorov & Shpilman (2022)) assumes the existence of a communication module that transfers the actions and hidden states of all agents. Consequently, under truly decentralized execution, such world models are primarily useful for policy training and cannot be directly reused by individual agents at test time, unlike their single-agent counterparts.

There is another challenge with adapting the mentioned model-based approaches to MARL as well, even if the individual prediction were available to individual agents. World models inevitably incur prediction errors and therefore cannot exactly reproduce observations from the true environment. While a small prediction error may have a limited effect in the single-agent setting, its consequences can be amplified in MARL. When an attacked agent replaces its corrupted observation with an imagined one, even a small discrepancy between the predicted and true observations can alter its action. Due to the interdependence among agents, this action change can in turn alter the dynamics experienced by the other agents, potentially affecting their subsequent actions as well. Such cascading deviations can ultimately lead to miscoordination among the agents.

Finally, existing model-based defense mechanisms can introduce substantial computational overhead, even in the single-agent setting when a world model is used. The aforementioned approaches require obtaining a fully reconstructed observation before anomaly detection can be performed. This can be computationally expensive, particularly for high-dimensional observations. For example, for a diffusion-based world model such as DIAMOND (Alonso et al. (2024)), generating a high-quality reconstruction requires multiple denoising steps, each involving a forward pass through the denoiser network. This overhead is particularly problematic for real-time test-time defense, as anomaly detection must be performed continuously at every environment step.

B World Models Generalize Poorly to Unseen Policies

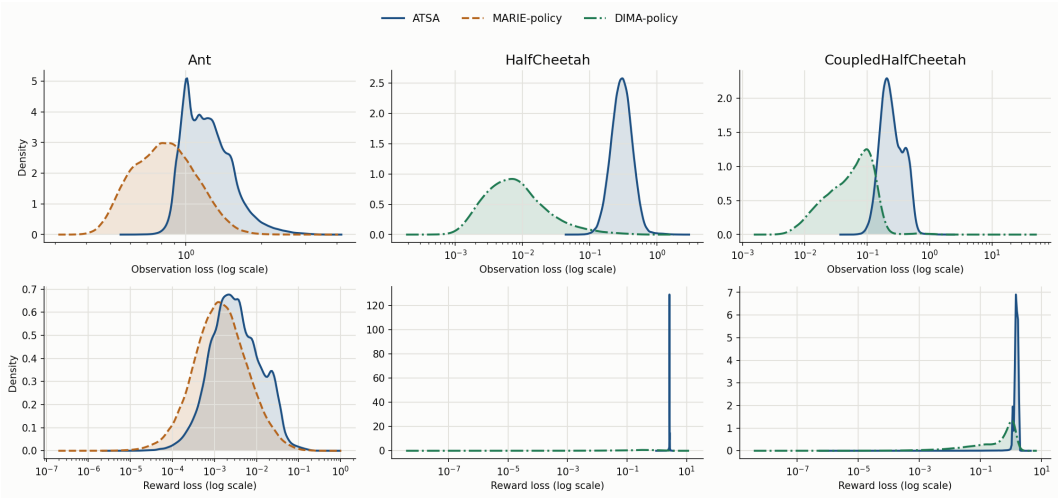


Figure 2: Distributions of per-step observation and reward loss for ATSA vs. each world model's co-trained policy on MaMuJoCo (Ant, HC, CHC).

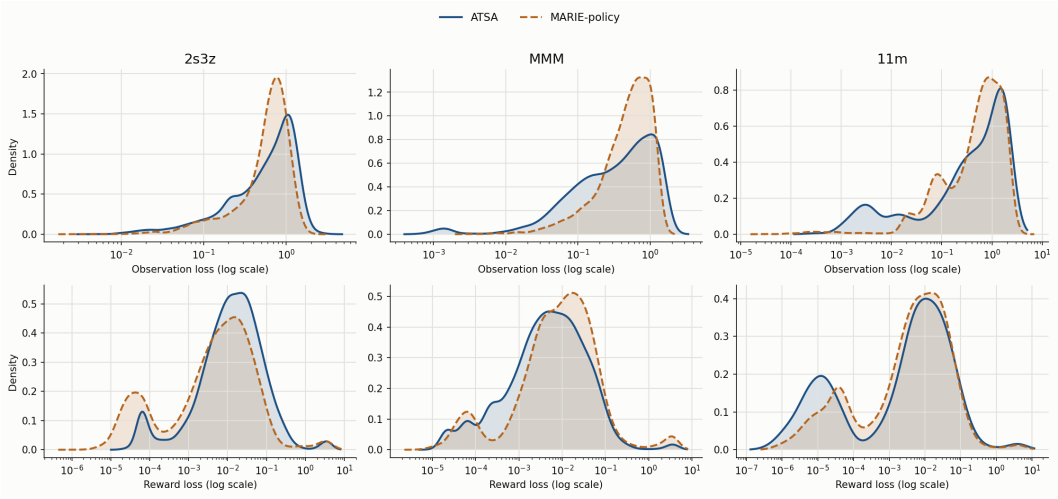


Figure 3: Distributions of per-step observation and reward loss for ATSA vs. each world model's co-trained policy on SMAC (2s3z, MMM, 11m).

Motivation. As we see in the experiments, pairing a policy with a frozen world model it was not jointly trained with does not result in good performance. We believe this is a result of the world model not generalizing well to scenarios it has not seen during training. To demonstrate our hypothesis, we present the following experiments.

Experimental setup. We consider two world models (WMs): DIMA, evaluated in MaMuJoCo (HC, CHC), and MARIE, evaluated in MaMuJoCo (Ant) and SMAC (2s3z, MMM, 11m). ATSA is the model-free policy. MARIE-policy and DIMA-policy are the policies co-trained with the respective world models. For each policy and scenario, we collect 200 real-environment episodes and replay them through the corresponding frozen WM. At each step, the WM is conditioned on the real

history rather than its own previous prediction, isolating one-step prediction accuracy from the compounding error of auto-regressive rollout. The results show the resulting per-step observation and reward loss distributions via kernel density estimation (KDE), which integrates to 1 by construction.

Interpretation. In the MaMuJoCo experiments (Figure 2), we clearly see that the WM prediction losses are concentrated at higher values in the distribution plots when the WM is tested on trajectories collected by ATSA, compared to trajectories from its own co-trained policy. This supports our hypothesis that a WM does not generalize well to policies it was not trained with. The SMAC experiments (Figure 3) do not offer as clear a view.

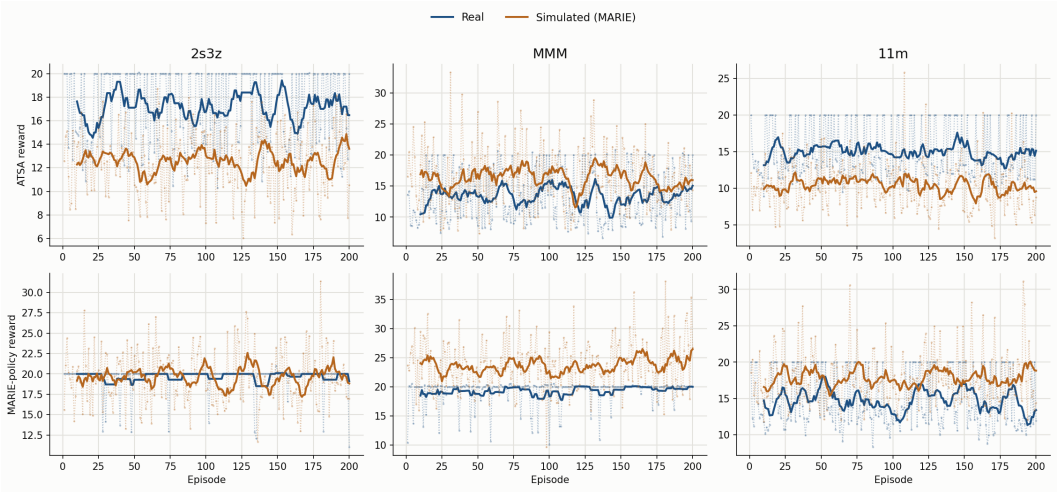


Figure 4: Per-episode reward in the real environment versus simulated through the frozen world model (MARIE), for ATSA (top row) and MARIE's own co-trained policy (bottom row), across SMAC (2s3z, MMM, 11m).

Further Experimentation. Since the per-step prediction losses do not offer a clear picture in SMAC, we additionally compare the reward each policy achieves in the real environment against the reward it achieves when evaluated through the frozen world model as a simulator (Figure 4). The solid line shows the 10-episode moving average, and the dotted line shows the actual per-episode reward. In 2s3z, ATSA's real and simulated rewards differ by 27.0%, while MARIE-policy's are essentially identical. In 11m, ATSA's real and simulated rewards differ by 30.7%, while MARIE-policy's differ by 20.6%, though this remaining gap is partly an artifact of the world model's reward prediction. A significant portion of MARIE-policy's simulated episodes yield a reward exceeding the environment's maximum achievable reward of around 20, unlike in the real environment, so MARIE-policy's real and simulated rewards are likely even closer than the raw numbers suggest. We find a similar interpretation holds for our MMM results. Together, these results support our hypothesis that world model is not a good simulator for a policy it was not trained with, closely matching real-environment reward only for the policy it was co-trained with.

C Implementation Details

In this section, we provide a more detailed description of the MATRIX architecture and its components. Figure 5 illustrates how the local and global modes of the diffusion model are used at different stages of training and execution. Algorithm 1 presents the pseudo-code for the MATRIX test-time protocol.

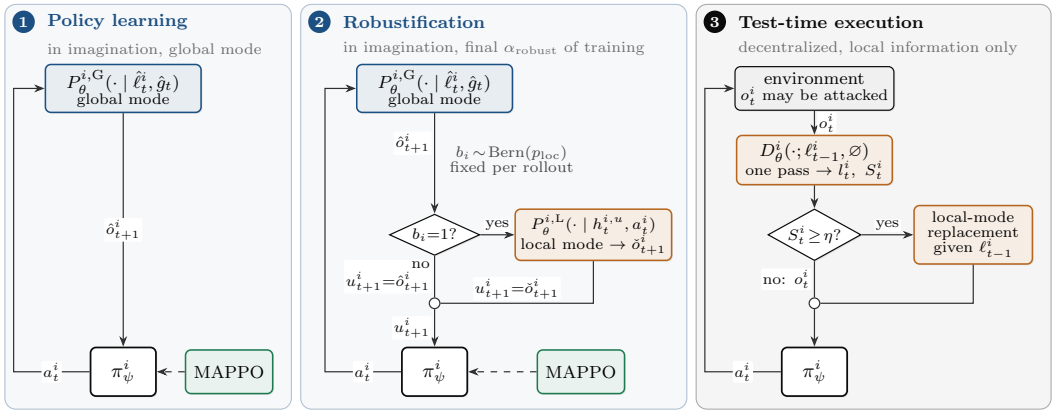


Figure 5: Interaction between the policy and the world model in the three phases of MATRIX.

Algorithm 1 MATRIX test-time defense for agent i

Require: Local-mode denoiser D^i_θ, policy π^i_ψ, bias κ, threshold η

Initialize $S^i \leftarrow 0$
for each time step t **do**
  Receive o^i_t; set policy input $u^i_t \leftarrow o^i_t$
  Construct ℓ^i_{t-1} from the local history
  Draw one σ_t and one $\epsilon^i_t \sim \mathcal{N}(0, I)$; set $\tilde{o}^i_t \leftarrow o^i_t + \sigma_t \epsilon^i_t$
  Compute l^i_t using Eq. 6 ▷ One denoiser pass
  $S^i \leftarrow \max\{0, S^i + l^i_t - \kappa\}$
  if $S^i \geq \eta$ **then**
    $u^i_t \leftarrow$ local-mode diffusion sample conditioned on ℓ^i_{t-1}
  end if
  Select and execute $a^i_t \sim \pi^i_\psi(\cdot \mid u^i_t)$
  Append (u^i_t, a^i_t) to the history
end for

C.1 MATRIX Structure

This section describes the network architecture underlying MATRIX, complementing the modeling objectives and training procedures in the main text. MATRIX comprises a diffusion-based observation predictor, a reward–termination predictor, a decentralized actor, and a centralized critic. Each component has its own parameters, with the observation predictor and actor sharing their parameters across agents. Centralized inputs are constructed from joint observations and actions; access to the underlying environment state is not required. Figure 6 shows a high level MATRIX's predictive components.

C.1.1 Diffusion-Based Observation Predictor

Observation representation. Observations are standardized using feature-wise running statistics estimated from collected environment experience, followed by a fixed scaling for diffusion modeling. These statistics are retained for deployment, and generated observations are transformed back

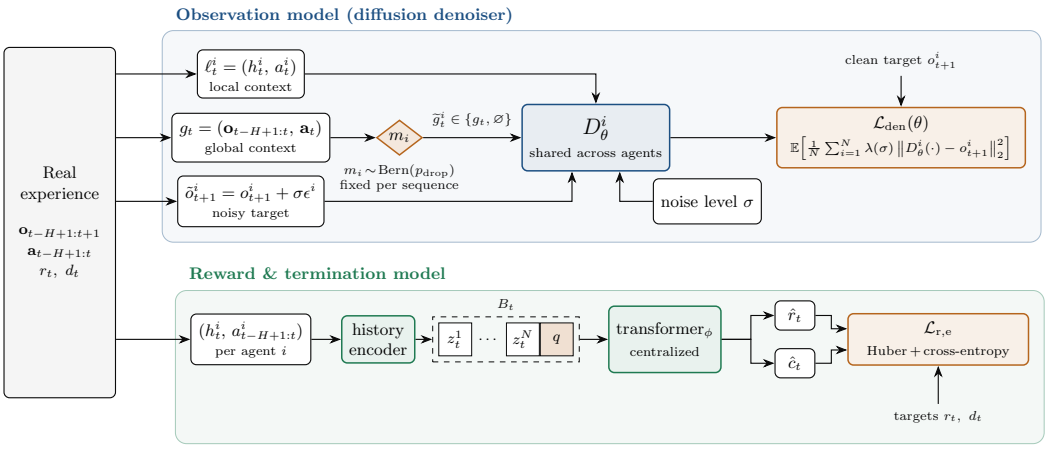


Figure 6: World model training in MATRIX: the diffusion denoiser (top) and the reward and termination model (bottom).

to the original observation coordinates before being passed to the actor. Below, x_t^i denotes agent i's observation in the coordinates used by the diffusion model, and H denotes the conditioning-history length.

Local and global observation inputs. For agent i, the denoising network receives a temporal sequence comprising its observation history and a noisy next-observation candidate. Global observation information is supplied through a separate sequence. At each historical time step τ, the joint observation is concatenated across agents and projected by a multilayer perceptron (MLP):

$$v_\tau = f_{\mathrm{glob}}\big([x_\tau^1; \ldots; x_\tau^N]\big), \qquad \widetilde{v}_\tau^i = \begin{cases} v_\tau, & m_i = 0, \\ v_\varnothing, & m_i = 1, \end{cases} \tag{7}$$

where m_i is the global-conditioning dropout mask and $v_\varnothing$ is a learned null embedding. The projection uses two affine layers separated by a SiLU activation. A second learned embedding, v_{fut}, occupies the next-observation position, where no global observation is available. The resulting input sequence is

$$X_t^i = \big([x_{t-H+1}^i; \widetilde{v}_{t-H+1}^i], \ldots, [x_t^i; \widetilde{v}_t^i], [\xi_{t+1}^i; v_{\mathrm{fut}}]\big), \tag{8}$$

where $[\cdot;\cdot]$ denotes feature concatenation and ξ_{t+1}^i is the noisy candidate after EDM input preconditioning. Thus, global information supplements the local history without replacing it. The future embedding is shared between the two conditioning modes and contains no observed future information.

Action encoding and conditioning embeddings. The current joint action is encoded separately from the observation sequence. Discrete actions use a learned embedding table, whereas continuous action vectors use an MLP with a SiLU activation. To produce the conditioning vector for agent i, we form an action token for every agent j:

$$\nu_t^{i,j} = e_{\mathrm{slot}}^j + \begin{cases} f_{\mathrm{act}}(a_t^j), & m_i = 0 \text{ or } j = i, \\ \nu_\varnothing, & m_i = 1 \text{ and } j \neq i, \end{cases} \tag{9}$$

where e_{slot}^j identifies the action's agent and $u_\varnothing$ is a learned null-action embedding. These tokens pass through multi-head self-attention and gated feed-forward layers, with pre-layer normalization and residual connections. Mean pooling over the output tokens, followed by layer normalization and a linear projection, yields an action-conditioning vector $e_{a,t}^i$. In local mode, other agents' action contents are replaced before attention; their fixed identity embeddings remain, but convey no information about their current actions. Then, we consider the following embedding

$$c_t^i = f_{\mathrm{cond}}(e_\sigma + e_{a,t}^i + e_{\mathrm{id}}^i), \tag{10}$$

where e_σ is a noise level embedding, e^i_{id} is a learned target-agent embedding, and f_{cond} is an MLP with a SiLU activation. Agent identities therefore distinguish both the sources of actions and the agent whose observation is being predicted. Dropping global conditioning removes the joint-observation contents and other agents' actions together, while retaining the target agent's history, action, and identity.

Temporal U-Net. An initial one-dimensional convolution projects X^i_t into hidden features. A temporal U-Net then processes these features using an encoder, bottleneck, and decoder with skip connections. Convolutions operate along the history dimension; agents are processed as separate prediction instances with shared weights. Strided convolutions reduce temporal resolution in the encoder, and transposed convolutions restore it in the decoder. Encoder features are concatenated with decoder features at corresponding resolutions.

Each residual block contains two temporal convolutions with SiLU activations and adaptive group normalization. The conditioning vector modulates normalized features through learned channel-wise scale and shift functions:

$$\mathrm{AdaGN}(z; c^i_t) = \big(1 + \alpha(c^i_t)\big) \odot \mathrm{GN}(z) + \beta(c^i_t). \tag{11}$$

The modulation is broadcast over temporal positions. Residual shortcuts use a linear channel projection when needed. The temporal backbone uses convolutional blocks; the self-attention described above belongs to the action encoder. After the decoder, group normalization, a SiLU activation, and an output convolution map features back to the observation dimension. Only the output at the final temporal position is used. EDM output preconditioning combines this output with the noisy candidate to obtain the denoised prediction D^i_θ. Local and global prediction use exactly this same network, differing only in the conditioning contents.

C.1.2 Reward–Termination Predictor

History tokens. The reward–termination model is a separate transformer operating on normalized observation–action histories. For agent i, its input is

$$q^i_t = [\bar{o}^i_{t-H+1}; \ldots; \bar{o}^i_t; a^i_{t-H+1}; \ldots; a^i_t], \qquad z^i_t = f_{\mathrm{hist}}(q^i_t) + e^i_{\mathrm{rew}}, \tag{12}$$

where $\bar{o}^i_t$ denotes the standardized observation and discrete actions are represented as one-hot vectors. The history encoder comprises an affine layer, layer normalization, a SiLU activation, and a second affine layer. Each complete history window is therefore represented by one continuous token per agent.

Joint temporal aggregation. The agent tokens and a learned summary-query token form a block $B_t = (z^1_t, \ldots, z^N_t, q_{\mathrm{rew}})$. Blocks are ordered in time, and learned position embeddings are added to the flattened token sequence. Transformer layers use multi-head self-attention, pre-layer normalization, residual connections, and feed-forward networks with GELU activations. The attention mask allows access to every token within the current block and to earlier blocks from the same episode, while excluding future blocks. This permits joint reasoning about simultaneous agent interactions without imposing an artificial causal ordering among agents. Padding is masked, and attention does not cross episode boundaries.

Prediction heads. The output at each summary-query position is passed to separate MLP heads for the shared team reward and continuation signal. Both heads use ReLU hidden activations; the reward head produces a scalar, and the continuation head parameterizes a Bernoulli distribution. They are trained using the reward-regression and continuation-classification objectives described in the main text. For SMAC, an additional head reads each agent-token output and predicts next-step action availability through a binary classification for each action. Observation generation is handled entirely by the diffusion model: the reward–termination transformer predicts the auxiliary quantities required to evaluate imagined transitions.

C.1.3 Centralized Critic

The critic embeds each agent's flattened, normalized observation history using an MLP with layer normalization and a SiLU activation, and adds a learned agent-identity embedding. A learned

Table 3: Principal hyperparameters of MATRIX. Per-phase counts refer to regular training after initialization. Transformer specifications are given as layers / attention heads / embedding dimension.

Hyperparameter	Value	
Shared architecture and diffusion settings		
Observation-history length H	3	
U-Net channel widths	$(64, 64, 64)$	
Conditioning / global-observation embedding dimension	$128 / 256$	
Action encoder: layers / heads / embedding dimension	$2 / 8 / 128$	
Reward–termination transformer	$6 / 4 / 256$	
Critic transformer	$1 / 4 / 256$	
Actor: hidden layers / hidden dimension	$3 / 128$	
Diffusion sampling steps	4	
Global-conditioning dropout p_{drop} / CFG scale w	$0.2 / 1.5$	
Shared optimization settings		
World-model optimizer / learning rate / weight decay	AdamW / 10^{-4} / 10^{-2}	
Actor and critic optimizer / learning rate	Adam / 5×10^{-4}	
Denoiser / reward–termination batch size (sequences)	$64 / 128$	
Denoiser updates per training phase	200	
Imagined trajectories per batch / rollout horizon	$600 / 15$	
Imagination batches per phase / PPO epochs per batch	$5 / 5$	
Discount γ / return parameter λ	$0.99 / 0.95$	
Initial entropy coefficient	10^{-3}	
Minimum replay size before training (real steps)	5,000	
Environment-family differences	**SMAC**	**MaMuJoCo**
Reward–termination updates per training phase	60	200
PPO clipping parameter ε	0.2	0.1
Entropy-coefficient decay per actor update	0.99998	1
Minimum new real steps between training phases	100	200

value-query token is appended to the agent tokens. Transformer encoder layers apply unrestricted self-attention across this set, using pre-layer normalization, residual connections, and GELU feed-forward networks. The final query representation is layer-normalized and projected to one scalar team value. Temporal information is contained within each history token; unlike the reward–termination model, the critic does not process a sequence of time-indexed token blocks. The resulting architecture conditions value estimation on joint observation histories while preserving local observation inputs for action selection. The critic and reward–termination model are used during training; decentralized deployment requires the actor and the locally conditioned denoiser.

C.2 HYPERPARAMETERS

Table 3 summarizes the principal architectural and training hyperparameters. Shared settings apply to all six scenarios; differences between SMAC and MaMuJoCo are listed separately. We set $\alpha_{\mathrm{robust}} = 0.1$ in all scenarios. For the local generation probability in the policy robustification phase, we set $p_{\mathrm{loc}} = 0.05$ in 2s3z, 11m, Ant, and HC. This value is $p_{\mathrm{loc}} = 0.1$ in MMM and CHC. The selection of test-time hyperparameters (κ and η in CUSUM) are described in Appendix D.2.

D EXPERIMENTATION DETAILS

D.1 COMPLEXITY COMPARISON OF THE WORLD MODELS

Table 4 compares the total number of parameters and training time of MATRIX, DIMA, and MARIE on 2s3z. All training times are measured on a dedicated NVIDIA RTX PRO 4500 Blackwell GPU and four CPUs.

As shown in the table, MATRIX achieves the performance reported in Figure 1 with considerably lower computational requirements than the other world-model baselines. Moreover, due to the per-agent design of the world model and parameter sharing across agents, the total parameter count of MATRIX does not increase substantially as the number of agents grows. For example, on MMM, the total number of parameters in MATRIX is only 8.7M.

Table 4: Total parameter count and training time of MATRIX, DIMA, and MARIE on 2s3z with a budget of 200K environment steps.

Method	Parameters (Millions)	Training time (hours)
MATRIX	8.20	36
DIMA	23.01	42
MARIE	14.60	70

D.2 TEST-TIME EVALUATION PROCEDURE

For each scenario, we evaluate a MATRIX policy and world-model checkpoint using separate calibration, validation, and test episodes. We first collect denoising-loss traces with the defense disabled under clean observations and two representative attack conditions (FGSM and ATSA), using 50 episodes per condition. These traces are used to propose CUSUM bias-threshold pairs (κ, η) that limit clean-episode alarms while favoring reliable and early attack detection. The two highest-ranked candidates are then evaluated with the defense enabled on separate validation episodes, again using 50 episodes per condition. We select the pair with the highest average validation return among those satisfying the clean-performance and false-alarm constraints, and fix it across all test attack types, budgets, and numbers of attacked agents within that scenario. Calibration, validation, and testing use disjoint random-seed ranges; test outcomes are never used for parameter selection, and neither the policy nor the world model is updated. Each final condition is evaluated over 200 episodes, with victim agents randomly selected per episode.

In all experiments, we use default SMAC configurations. For MaMuJoCo scenarios, in order to have a truly local observations, we set the variable `agent_obsk=0`.

D.3 BASELINES' IMPLEMENTATIONS

For DIMA, MARIE, ATSA, and BATPAL, we use their official implementations as follows:

- DIMA: https://github.com/breez3young/DIMA
- MARIE: https://github.com/breez3young/MARIE
- ATSA: https://github.com/zhou-ziyuan/ATSA
- BATPAL: https://github.com/kiarashkaz/BATPAL

For PAAD, we used the implementation included in the ATSA codebase.

E Additional Results

E.1 Adversarial Robustness of Baseline World Models

Table 5 shows the test-time performance of baseline world-models against no attack, as well as against ATSA and FGSM attacks. For each MARL scenario we chose the approach that resulted in the higher average performance at the end of its training, and compared its test time robustness with MATRIX. As it can be observed, with the exception of the CHC environment, DIMA and MARIE, experience substantial performance degradation when exposed to test-time attacks. This comparison indicates that the robustness of MATRIX arises from its robustness-specific design rather than merely from the use of model-based learning.

Table 5: Test-time performance of world model baselines in clean and attacked scenarios

Scenario	Method	Clean	FGSM	ATSA
2s3z	MATRIX	**0.92**	**0.83**	**0.78**
	DIMA	0.74	0.28	0.37
MMM	MATRIX	0.72	**0.70**	**0.69**
	MARIE	**0.88**	0.51	0.44
11m	MATRIX	**0.75**	**0.63**	**0.55**
	MARIE	0.44	0.08	0.05
Ant	MATRIX	**5267**	**3539**	**3537**
	MARIE	2286	509	1405
HalfCheetah	MATRIX	**4386**	**2353**	**2760**
	DIMA	3462	1170	1460
CoupledHalfCheetah	MATRIX	**1169**	**1142**	**1063**
	DIMA	1058	1006	965

E.2 Additional Robustness Evaluations

We further evaluate MATRIX, ATSA, PAAD, and BATPAL under FGSM and ATSA attacks with perturbation budgets $B \in \{0.05, 0.10, 0.25, 0.40\}$ and $N_a \in \{1, 2, 3\}$ attacked agents. Figures 7–9 report the average performance across all considered scenarios. The results show that MATRIX consistently outperforms the other baselines by a considerable margin, with only a few exceptions. Moreover, the performance of MATRIX remains generally consistent as the number of attacked agents increases, demonstrating its robustness under attacks affecting multiple agents simultaneously.

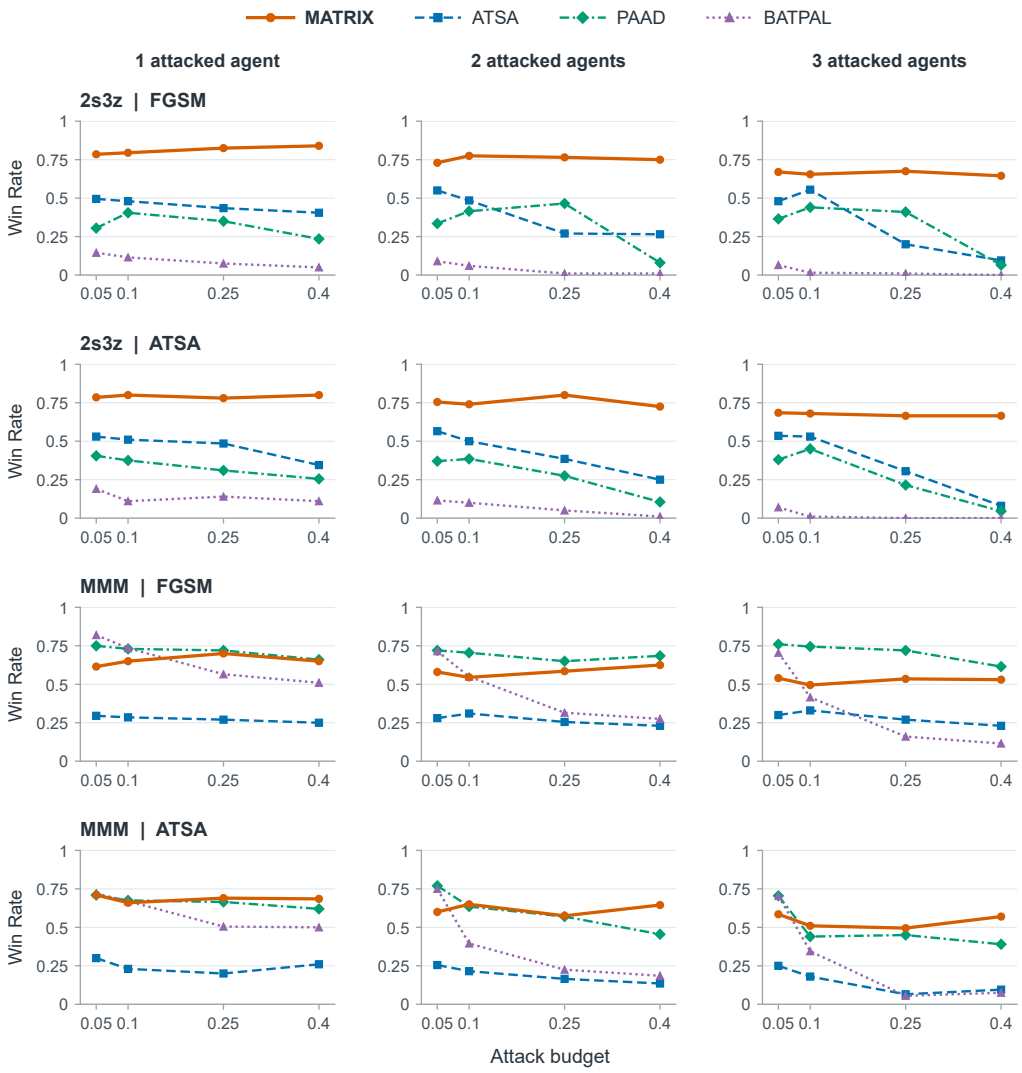


Figure 7: Additional robustness evaluations on 2s3z and MMM. Rows identify the scenario and attack; columns correspond to one, two, and three attacked agents.

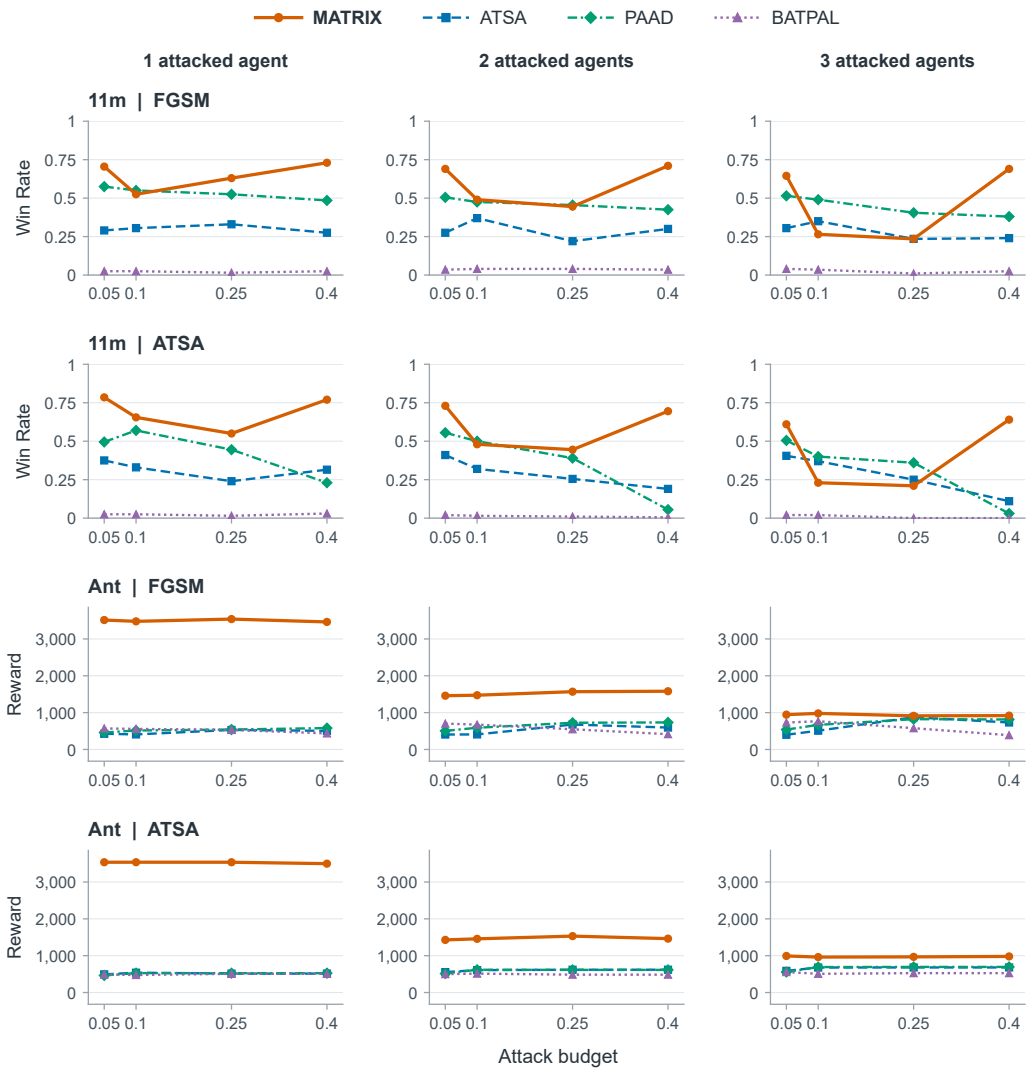


Figure 8: Additional robustness evaluations on 11m and Ant. Rows identify the scenario and attack; columns correspond to one, two, and three attacked agents.

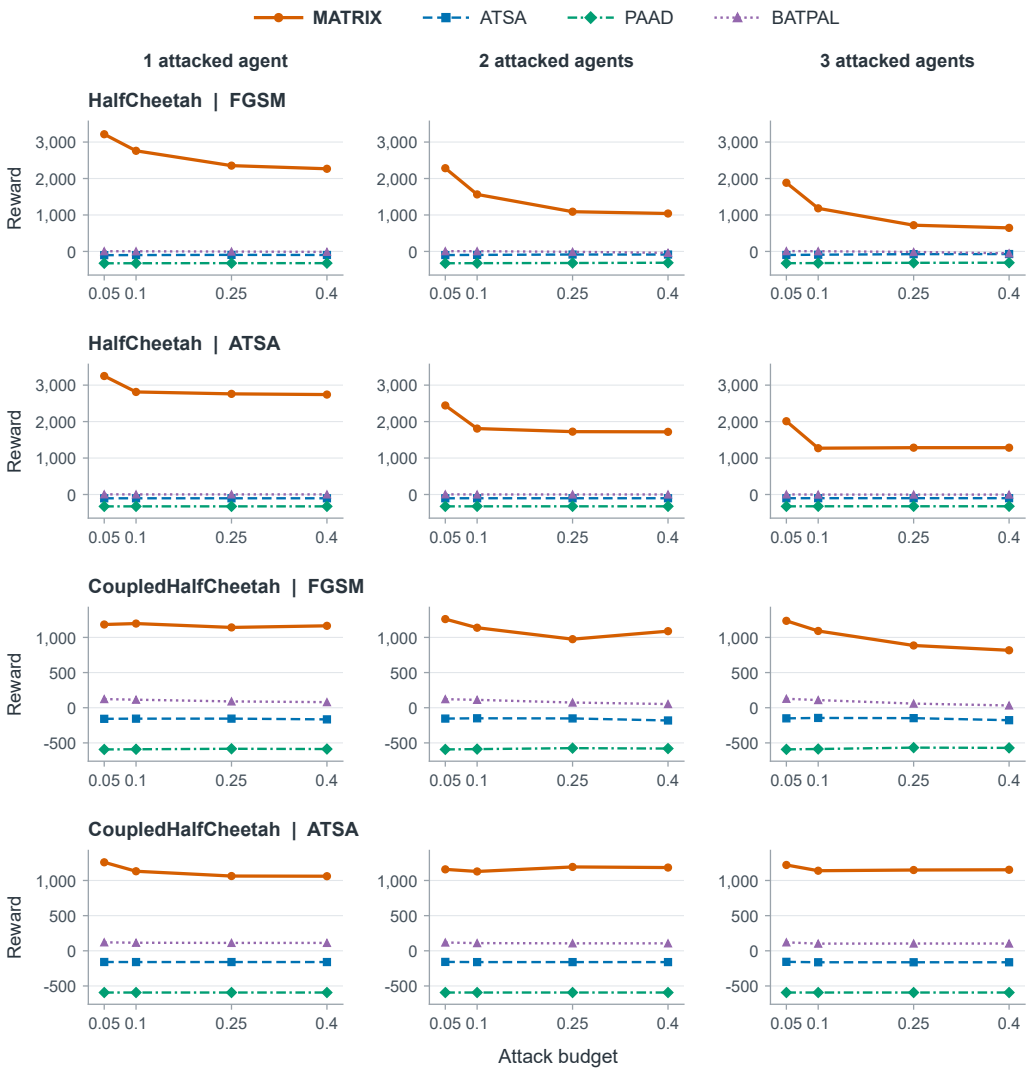


Figure 9: Additional robustness evaluations on HalfCheetah and CoupledHalfCheetah. Rows identify the scenario and attack; columns correspond to one, two, and three attacked agents.