MASTARS: Multi-Agent Sequential Trajectory Augmentation with Return-Conditioned Subgoals

Jiwon Jeon[1] **Myungsik Cho**[1] **Woojun Kim**[1] **Seongmin Kim**[1]
Woohyeon Byeon[1] **Seungyul Han**[2*] **Youngchul Sung**[1*]
[1]KAIST [2]UNIST
{jiwon.jeon, ms.cho, woojun.kim, seongmin.kim,
woohyeon.byeon, ycsung}@kaist.ac.kr, syhan@unist.ac.kr

Abstract

The performance of offline Multi-Agent Reinforcement Learning (MARL) is limited by the quality of the fixed offline dataset, motivating trajectory augmentation. However, augmenting multi-agent trajectories presents a fundamental challenge of the diversity–coordination trade-off: independent generation improves diversity but lacks coordination, while joint generation induces coordination but lacks diversity, reproducing the training distribution. To address this challenge in multi-agent data augmentation, we introduce MASTARS, a novel diffusion-based framework that generates multi-agent trajectories sequentially across agents while enforcing cross-agent consistent coordination via the method of inpainting. MASTARS produces trajectories that are diverse, coordinated and globally coherent. Experiments on various benchmarks such as MPE, SMAC, and SMAC-v2 demonstrate significant performance gains across multiple offline MARL methods. Code is available at https://github.com/Jiwonjeon9603/MASTARS.

1 Introduction

Offline reinforcement learning (RL) [18] aims to learn policies from pre-collected datasets without additional environment interaction, enabling safe and efficient policy development in domains where online exploration is costly or risky. While recent algorithmic advancements have yielded remarkable success [7, 16, 37, 26], the effectiveness of these approaches remains dependent on the quality of the underlying dataset [34]. To address this limitation, *data augmentation* has emerged as an important research direction in offline RL, and several methods have been proposed based on the strong generative capabilities of diffusion models [10] to produce high-quality synthetic data in single-agent RL [12, 22, 24, 8, 20, 17].

Extending offline single-agent data augmentation to multi-agent RL (MARL) introduces unique challenges due to the combinatorial growth of joint state–action spaces and the need for agent coordination. While prior work has proposed specialized offline MARL methods [41, 29, 38, 35, 15], their performance remains sensitive to the quality and diversity of the dataset. Unlike single-agent augmentation, MARL requires modeling coordinated joint behaviors across agents. However, widely-considered independent per-agent diffusion models neglect cross-agent dependencies, thereby hindering coordination. This motivates joint trajectory generation approaches [44, 27, 6], but existing joint approaches tend to produce limited diversity, frequently replicating the training distribution.

The key challenge. Multi-agent augmentation is not merely a generative modeling problem; it is a constrained synthesis problem. A useful augmented trajectory must introduce diverse behavior, preserve inter-agent coordination, and remain globally coherent with the environment. We refer to

*Correspondence to: Seungyul Han (syhan@unist.ac.kr) or Youngchul Sung (ycsung@kaist.ac.kr).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

this as the *diversity–coordination–coherence trade-off*. Existing approaches sacrifice at least one of these requirements: independent generation improves diversity but breaks coordination, joint generation preserves coordination but often collapses toward the dataset distribution, and naive sequential generation can introduce incoherent global states.

To address these challenges in multi-agent data augmentation, we propose **M**ulti-**A**gent **S**equential **T**rajectory **A**ugmentation with **R**eturn-conditioned **S**ubgoals **(MASTARS)**, a novel diffusion-based framework that generates cooperative multi-agent trajectories in an agent-wise sequential manner guided by high-value subgoals, while maintaining global coherence via a diffusion inpainting mechanism. MASTARS is built upon three core components: (i) an agent-wise sequential trajectory generation strategy, which generates coherent trajectories for each agent while coordinating with others via the key idea of *inpainting*. To reduce the search space for trajectory augmentation, rather than synthesizing trajectories from scratch, MASTARS leverages value-based subgoals through: (ii) partial generation based on subgoal identification, which preserves each agent's trajectory prefix up to its highest-value timestep and regenerates only the continuation, and (iii) subgoal-based generation ordering, which determines the order of sequential generation based on value-based subgoals.

Through these approaches, MASTARS ensures diverse, coordinated, and coherent data augmentation in the multi-agent setting, enhancing performance of major offline MARL methods when applied to the augmented dataset. Experiments on various MARL benchmarks, including the Multi-Agent Particle Environments (MPE) [23], the StarCraft Multi-Agent Challenge (SMAC) [32], and SMAC-v2 [4] demonstrate that datasets augmented by MASTARS significantly improve the performance of various offline MARL methods.

The main contributions of this work are as follows:

- We identify a diversity–coordination–coherence trade-off in multi-agent augmentation, where each of independent, joint and naive sequential strategies sacrifices at least one of these properties.
- We propose MASTARS, a plug-in diffusion-based framework resolving this tradeoff via inpainting-based sequential generation and reducing augmentation's search space with value-based subgoals.
- We demonstrate consistent performance gains of MASTARS across offline MARL algorithms on MPE, SMAC, and SMAC-v2, with ablations validating each component.

2 Preliminaries

2.1 Multi-Agent Offline Reinforcement Learning

We consider a fully cooperative Partially Observable Markov Game with N agents, formalized as a Dec-POMDP $\langle N, \mathcal{S}, \{\mathcal{A}^i\}_{i=1}^N, \Omega, \mathcal{O}, P, r, \rho, \gamma\rangle$ [28]. Here, $\mathcal{S}$ is the state space, $\mathcal{A}^i$ the action space of agent i, Ω the observation space, $\rho : \mathcal{S}\to[0,1]$ the initial state distribution, and $\gamma \in [0,1)$ the discount factor. At each timestep, agents share the environment state $s \in \mathcal{S}$ but receive private observations $\{o^i \in \Omega\}$ according to the observation function $\mathcal{O} : \mathcal{S}\times\mathcal{N}\to\Omega$. Each agent i selects an action $a^i \in \mathcal{A}^i$, forming the joint action $\mathbf{a} = (a^1, \ldots, a^N)$. The environment transitions via $P(s' \mid s, \mathbf{a}) : \mathcal{S}\times\mathcal{A}^1\times\cdots\times\mathcal{A}^N\to\mathbb{R}$ and yields a global reward $r(s, \mathbf{a})$. Bold symbols denote joint quantities, e.g., $\boldsymbol{o} = (o^1, \ldots, o^N)$.

In the offline setting, we are given a fixed dataset $\mathcal{D} = \{\tau_m^{\text{full}}\}_{m=1}^M$ of M trajectories $\tau^{\text{full}} = (\boldsymbol{o}_{0:T}, \boldsymbol{a}_{0:T-1}, r_{0:T-1})$ with maximum trajectory length T, collected by an unknown behavior policy. No further interaction with the environment is permitted. The goal is to learn a decentralized set of policies $\boldsymbol{\pi} = \{\pi^i(a^i \mid o^i; \theta^i)\}_{i=1}^N$ (or an equivalent centralized-training factorization) that maximizes the expected return $J(\boldsymbol{\pi}) = \mathbb{E}_{\tau\sim\boldsymbol{\pi},P}\Big[\sum_{t=0}^{T-1}\gamma^t r_t\Big]$.

2.2 Diffusion-Based Generative Models

Diffusion models [10, 36] consist of a forward process that gradually corrupts data with Gaussian noise and a learned reverse process that denoises the corrupted data. In the forward process, noise is injected into $x_0 \sim q(x_0)$ over K steps using a fixed variance schedule $\{\beta_k\}_{k=1}^K$: $q(x_k \mid x_{k-1}) = \mathcal{N}\big(\sqrt{1-\beta_k}x_{k-1}, \beta_k I\big)$, resulting in a Gauss–Markov process with $q(x_k \mid x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_k}x_0, (1-$

$\bar{\alpha}_k)I)$, where $\bar{\alpha}_k = \prod_{i=1}^{k}(1-\beta_i)$. The posterior $q(x_{k-1} \mid x_k, x_0)$ is also Gaussian. The reverse generative model p_θ starts from $x_K \sim \mathcal{N}(0, I)$ and generates samples via the reverse transitions: $p_\theta(x^{0:K}) = \mathcal{N}(x^K; 0, I) \prod_{k=1}^{K} p_\theta(x^{k-1} \mid x^k)$.

DDPM [10] converts learning p_θ into learning a noise model ϵ_θ, enabling reverse sampling as $x_{k-1} = \frac{1}{\sqrt{\alpha_k}}\left(x_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(x_k, k)\right) + \sqrt{\beta_k} z$ with $x_K, z \sim \mathcal{N}(0, I)$. A neural network for the noise model $\epsilon_\theta(x_k, k)$ is trained to predict the noise $\epsilon \sim \mathcal{N}(0, I)$ in the forward decomposition $x^k = \sqrt{\bar{\alpha}_k} x_0 + \sqrt{1-\bar{\alpha}_k}\epsilon$, using the simple mean-squared error (MSE) objective:

$$L_{\text{diff}}^{\text{DDPM}}(\theta) = \mathbb{E}_{x_0 \sim q(x_0),\, k \sim \mathcal{U}[1,K],\, \epsilon}\left[\|\epsilon_\theta(x_k, k) - \epsilon\|^2\right]. \tag{1}$$

Diffusion Models in Data Augmentation. To generate offline RL datasets, diffusion models [12, 22, 20, 17] learn the trajectory distribution $p_\theta(\tau)$ via ϵ_θ from an offline dataset using Eq. (1). The basic idea of RL dataset generation with diffusion models is described in Appendix C. In particular, Diffuser [1] proposes a return-conditioned diffusion model with the loss [9]:

$$\mathcal{L}(\theta) = \mathbb{E}_{k, \{\tau_0, R\} \in \mathcal{D}, \epsilon, \beta \sim Bern(p)} \quad \left[\left\|\epsilon - \epsilon_\theta\big(\tau_k, k, (1-\beta)R + \beta\emptyset\big)\right\|^2\right], \tag{2}$$

where $\emptyset$ is a null token for the unconditioned model. The reverse process uses the perturbed noise $(1+w)\epsilon_\theta(\tau_k, k, R) - w\epsilon_\theta(\tau_k, k, \emptyset)$, conditioned on return R with guidance weight w. MADiff [49] generates sequences of observations by a return-conditioned diffusion model with an inter-agent attention architecture for MARL. We adopt MADiff as our base diffusion model and further train a reward function $g_\xi(\boldsymbol{o}, \boldsymbol{a})$, a transition model $f_\varphi(o^i, a^i)$, and an inverse dynamics model $I_\phi(o^i, o^{i'})$ to predict rewards, observations, and actions from a generated observation sequence [1, 20]. Our training objective for the auxiliary models is defined as

$$\mathcal{L}(\xi, \phi, \varphi) = \mathbb{E}_{\boldsymbol{o}, \boldsymbol{a}, r, \boldsymbol{o}' \sim \mathcal{D}}\Big[\big(r - g_\xi(\boldsymbol{o}, \boldsymbol{a})\big)^2 + \sum_{i=1}^{N}\Big(\big(a^i - I_\phi(o^i, o^{i'})\big)^2 + \big(o^{i'} - f_\varphi(o^i, a^i)\big)^2\Big)\Big]. \tag{3}$$

3 Related Works

Offline MARL. In offline MARL, the rapidly growing joint action space creates challenges such as coordination under partial observability, out-of-distribution (OOD) errors, and convergence to suboptimal policies. Prior works address these via conservative value estimation (CFCQL [35]), hybrid optimization (OMAR [29]), and global-to-local regularization (OMIGA [38]). While these methods focus on value-based constraints, diffusion models have shown efficacy in modeling complex action distributions and reducing extrapolation errors in single-agent offline RL. Building on this, MADiff [49] extends diffusion to multi-agent settings with attention-based coordination, and DoF [19] generalizes the Individual-Global-Max (IGM) principle to the Individual-Global-Distributed (IGD) framework, aligning multi-agent outcomes with individual distributions.

Sequential Modeling in MARL. Recent work has explored treating MARL as a sequential generation problem [40, 11, 48, 2, 46, 47]. These approaches show that imposing an ordering over agents, communications, or latent variables can improve coordination and scalability. However, they target policy learning and message passing rather than trajectory augmentation. In contrast, MASTARS adopts a sequential paradigm for data augmentation and combines it with diffusion-based inpainting to refine trajectories conditioned on other agents, enabling coordinated multi-agent synthesis.

Data Augmentation. Data augmentation has been widely explored in vision to improve generalization, with inpainting methods (e.g., RePaint [25], PSM [21], CE [30]) using generative models to reconstruct missing regions without retraining. In offline RL, where performance is limited by the given dataset, augmentation strategies include model-based methods that learn uncertain dynamics [14, 42, 43, 45] and generative approaches that synthesize high-quality trajectories [24, 20, 17].

In multi-agent settings, EAQ [27] and INS [6] improve coordination through Q-guidance and sparse-attention-based synthesis. However, these joint generation methods often struggle with trajectory diversity. While MADiTS [44] attempts to mitigate spatial imbalances by identifying underperforming agents and performing joint resampling via partial noising, its one-shot refinement process often lacks consistency under complex coordination, potentially leading to noisy target states and misaligned behaviors. In contrast, MASTARS employs an agent-wise sequential framework with RePaint-style inpainting to ensure inter-agent harmonization, thereby synthesizing diverse, coherent, and coordinated trajectories that go beyond the limitations of the original data.

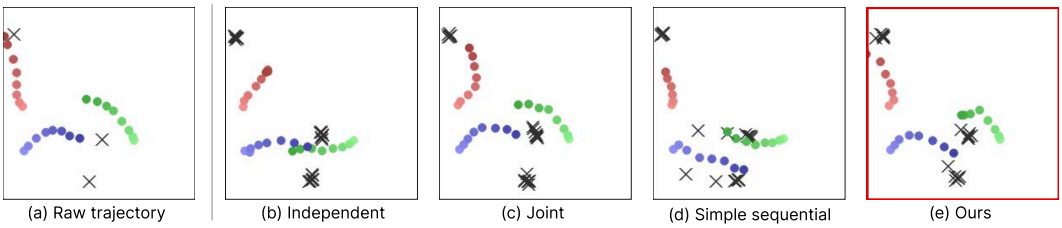


Figure 1: Comparison of trajectory generation strategies: Existing strategies struggle to jointly achieve coordination, diversity, and global coherence, while MASTARS satisfies all three. `X` marks are landmark positions reconstructed from each agent's observation.

4 Motivation

Generating high-quality augmented trajectories for offline MARL requires accurate modeling of inter-agent dependencies and sufficient diversity to meaningfully expand the dataset's support. However, existing diffusion-based data augmentation methods struggle to satisfy these requirements simultaneously. The limitation is illustrated in Fig. 1, which compares 1) agent-wise independent generation, 2) joint generation, 3) naive sequential generation and 4) our method, all implemented using MADiff [49]. Details of the generation processes are provided in Appendix D.

We consider the offline Spread medium-replay task from the Multi-agent Particle Environment (MPE) [23], where each of three agents (green, red, and blue) must reach a distinct landmark (marked as '`X`') while avoiding collisions. Following the standard MPE setup, each agent's observation is a concatenation of its `own velocity`, its `own position`, the `landmark positions` encoded relative to the agent, and the `relative positions of the other agents`. A trajectory thus corresponds to a sequence of these per-agent observations, so a diffusion-based augmentation method must generate the entire sequence, including the landmark coordinates (`X`) at every timestep. As shown in Fig. 1, none of the baseline strategies can consistently produce fully cooperative trajectories. The independent approach (Fig. 1(b)) frequently results in collisions because it neglects inter-agent dependencies. While the joint method mitigates collision risks, it lacks diversity by merely replicating the original data distribution, thereby inheriting the sub-optimality of the raw trajectories: Note that both blue and green agents are converging on the same landmark in Fig. 1(c). Finally, simple sequential generation (Fig. 1(d)) can yield diverse and collision-free trajectories through conditioning, but it fails to generate globally coherent trajectories; note that the generated rollouts exhibit inconsistent and fluctuating landmark positions although landmarks remain fixed throughout an episode. This inconsistency prevents agents from reliably reaching their designated goals.

Even this simple generation experiment reveals the challenge of generating multi-agent trajectories that are simultaneously coherent, diverse, and well-coordinated across agents. In particular, ensuring global consistency over sequential generation is difficult to achieve via naive agent-wise or sequential generation alone, motivating the need for complementary measures.

In the remainder of the paper, we propose a novel sequential generation framework that integrates subgoal-based partial generation with a diffusion-based inpainting mechanism to overcome the limitation of naive sequential generation. With these new ingredients, our approach enforces consistency throughout sequential generation, and yields harmonized trajectories with coordination and global coherence. In our generation (Fig. 1(e)), agents avoid collisions and reach distinct landmarks while maintaining consistent landmark positions, demonstrating that our method produces globally-coherent and well-coordinated trajectories that surpass the quality of the original offline dataset.

5 Proposed Method

To achieve desired multi-agent data augmentation, we combine two complementary mechanisms: i) resolving the diversity–coordination–coherence tradeoff via agent-wise sequential generation and ii) reducing the search space for trajectory augmentation via subgoal-based partial generation. Here, agent-wise sequential generation promotes diversity and coordination by regenerating each agent conditioned on previously generated agents, while diffusion-based inpainting additionally enforces global coherence by treating previously generated agents and preserved trajectory segments as known

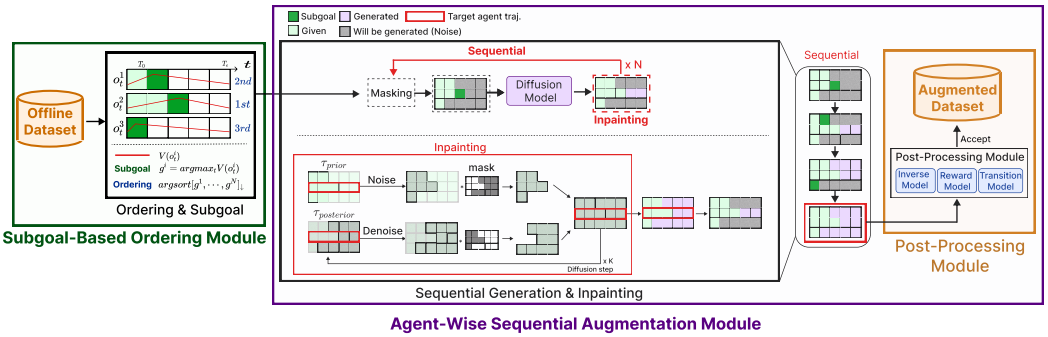


Figure 2: **MASTARS overview.** Value-based subgoals are identified, agents are ordered by subgoal timing, and the selected segments are regenerated sequentially via diffusion inpainting with previously generated agents as known context.

regions. Subgoal-based partial generation preserves each agent's trajectory prefix up to its highest-value timestep and regenerates only the continuation, and subgoal-based ordering determines the generation order based on value-based subgoals. The overall flow of MASTARS is shown in Fig. 2; the following subsections describe each component in detail.

5.1 Agent-Wise Sequential Trajectory Generation

First, we formulate the agent-wise generation step as an inpainting problem [3]: previously generated agents and preserved segments form the known regions, and the current agent is generated as the missing region conditioned on them, ensuring coordination. To further enforce global coherence, we adopt RePaint [25], a training-free diffusion inpainting method whose iterative resampling and re-noising harmonize the generated agents with the global trajectory.

Inpainting-Based Agent-Wise Sequential Generation. Formally, we define a MARL observation trajectory τ as an $N\times d\times H$ tensor, where N denotes the number of agents, d the observation dimension, and H the time horizon. Given an initial trajectory $\tau^0=\{\boldsymbol{o}_{t:t+H-1}\}\in\mathbb{R}^{N\times d\times H}$ and a generation order $(\kappa(1),\ldots,\kappa(N))$, we reconstruct each agent's trajectory using the RePaint algorithm, treating previously generated agent trajectories as known regions. RePaint is applied sequentially to each agent, resulting in N *outer loop* invocations, each consisting of an *inner loop* of K denoising steps following the diffusion process in Section 2.2. The initial trajectory τ^0 is sampled directly from the dataset and is progressively refined through this RePaint-based reconstruction process.

At the l-th RePaint generation step, we start with τ^{l-1}, the partially generated trajectory after the $l-1$ steps; that is, the trajectories for agents $\kappa(1),\ldots,\kappa(l-1)$ have already been generated. To identify the portion currently under generation, we define a binary mask m^l of size $N\times d\times H$.

$$m^l(j,d,t)=\begin{cases}1\;\forall t,d, & \text{if } j\in\{\kappa(l),\ldots,\kappa(N)\},\\ 0\;\forall t,d, & \text{otherwise,}\end{cases}\tag{4}$$

where the segments of the remaining agents $\kappa(l),\ldots,\kappa(N)$ after the $l-1$ generation steps are set to 1, and the remaining segments to 0.

With the binary mask m^l, the reverse-sampling inner loop starts with index $k=K$. For $k=K$, we initialize $\tilde{\tau}_K^{\text{posterior}}\sim\mathcal{N}(0,I)$ and $\tilde{\tau}_K^{\text{prior}}\sim\mathcal{N}\left(\sqrt{\bar{\alpha}_K}\tau^{l-1},(1-\bar{\alpha}_K)\mathbf{I}\right)$. Then, the denoising steps are applied according to the following sampling rules from $k=K$ to $k=1$ to obtain $\tilde{\tau}_0^l$:

$$\tilde{\tau}_k^l=m^l\odot\tilde{\tau}_k^{\text{posterior}}+(1-m^l)\odot\tilde{\tau}_k^{\text{prior}}\in\mathbb{R}^{N\times d\times H}\tag{5a}$$

$$\tilde{\tau}_{k-1}^{\text{posterior}}=\frac{1}{\sqrt{\alpha_k}}\left(\tilde{\tau}_k^l-\frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(\tilde{\tau}_k^l,k)\right)+\sqrt{\beta_k}z,\qquad z\sim\mathcal{N}(0,I)\tag{5b}$$

$$\tilde{\tau}_{k-1}^{\text{prior}}\sim\mathcal{N}\left(\sqrt{\bar{\alpha}_k}\tau^{l-1},(1-\bar{\alpha}_k)\,\mathbf{I}\right).\tag{5c}$$

Note that, for $k>1$, the prior input is a perturbed version of τ^{l-1} since $\bar{\alpha}_k<1$, which helps improve coherence between the prior and posterior. Throughout the reverse sampling loop for agents $\kappa(l),\cdots,\kappa(N)$, previously generated agents' data are appended to enable coordinated generation, as shown in Eq. (5a).

After the internal denoising loop, the output of the l-th generation step is obtained as $\tau^l = \tilde{\tau}_0^l$, where the trajectory of the current agent $\kappa(l)$ is updated while those of agents $\kappa(1), \ldots, \kappa(l-1)$ remain unchanged. Although agents $\kappa(l+1), \ldots, \kappa(N)$ are temporarily processed to maintain multi-agent coherence, their outputs are discarded and regenerated in later steps. This inpainting procedure proceeds sequentially following the order $(\kappa(1), \ldots, \kappa(N))$ until all agent trajectories are generated.

Post-Processing and Validation for Generated Dataset. After sequential generation, we obtain a synthesized observation trajectory $\tau = (\tilde{\boldsymbol{o}}_{t:t+H-1})$, where the tilde denotes generated quantities. To include this trajectory into an offline dataset, we infer the corresponding action and reward sequences using the pre-trained inverse dynamics model I_ϕ and reward model g_ξ, resulting in the following predictions: $\tilde{a}^i_{t'} = I_\phi(\tilde{o}^i_{t'}, \tilde{o}^i_{t'+1})$ and $\tilde{r}_{t'} = g_\xi(\tilde{\boldsymbol{o}}_{t'}, \tilde{\boldsymbol{a}}_{t'})$.

To ensure consistency with the environment dynamics and avoid out-of-distribution errors, we validate each generated trajectory using a pre-trained transition model f_φ. Trajectories are included in the augmented dataset only if their average prediction error satisfies the following condition:

$$\mathbb{E}_{i\in[1,N],\ t'\in[t,\,t+H-1]} \left\| \tilde{o}^i_{t'+1} - f_\varphi(\tilde{o}^i_{t'}, \tilde{a}^i_{t'}) \right\| < \epsilon, \tag{6}$$

where ϵ controls the acceptable level of deviation under the transition model [20].

5.2 Partial Generation via Subgoal Identification

While agent-wise sequential generation with inpainting addresses the diversity–coordination–coherence tradeoff, generating an entire trajectory from scratch requires the diffusion model to search over the full trajectory space. To reduce this search space, we further use *subgoal-based partial generation*. That is, each agent's trajectory prefix up to a high-value timestep is preserved as an anchor, and only the remaining continuation is regenerated. This turns full-trajectory synthesis into a tractable continuation problem from this anchor.

Given an original trajectory $\tau^0 = \{\boldsymbol{o}_{t:t+H-1}\} \in \mathbb{R}^{N\times d\times H}$, we define a *subgoal* g^i for each agent i as the timestep with the highest estimated value: $g^i = \arg\max_{t'\in[t,t+H-1]} V_\zeta(o^i_{t'})$.

The prefix up to g^i is treated as reliable behavior and kept fixed, while the suffix after g^i is regenerated. This selective regeneration preserves useful behavior from the original dataset while reducing the search space for trajectory augmentation.

The subgoal-conditioned generation is integrated into our agent-wise sequential generation process by modifying the mask $m^l(j,d,t)$ in Eq. (4) as follows:

$$m^l(j,d,t) = \begin{cases} 1\ \forall d, & \text{if } j \in \{\kappa(l), \ldots, \kappa(N)\},\ t > g^j, \\ 0\ \forall t, d, & \text{otherwise.} \end{cases} \tag{7}$$

5.3 Agent Ordering to Preserve Coordination Flexibility

The generation order affects how much flexibility remains for coordinating agents during sequential generation. If an agent that strongly constrains later trajectory evolution is generated too early, subsequent agents may have limited flexibility to coordinate with it. MASTARS therefore uses subgoal timesteps to choose the order $(\kappa(1), \ldots, \kappa(N))$. Since an agent with an earlier subgoal has a longer suffix to regenerate, generating it later allows the agent to better adapt to the trajectories generated before it.

Order of Generation. The subgoal g^i of agent i is the timestep at which its estimated value is maximized. We sort agents in descending order of their subgoal indices:

$$(\kappa(1), \ldots, \kappa(N)) = \mathrm{argsort}\left[g^1, \ldots, g^N\right]_\downarrow, \tag{8}$$

where $\downarrow$ denotes "sorting in descending order". This ordering generates agents with later subgoals first and defers agents with earlier subgoals to later stages, preserving flexibility for collision avoidance and multi-agent coordination.

Note that preserving the relative ordering of state values is important for subgoal identification and generation ordering in MASTARS. Fortunately, since values are needed only for observed states, SARSA-style learning in Eq. (12) in Appendix E.2 is sufficient, avoiding extrapolation and improving stability.

Table 1: Normalized return in the MPE environment (**bold**: best, underline: second best)

Alg	Dataset	Tag			Spread			World			Average
		Medium	MD-Replay	Random	Medium	MD-Replay	Random	Medium	MD-Replay	Random	
CFCQL	Original	44.3±16.5	43.3±13.7	7.4±2.1	16.6±9.8	31.2±14.0	18.8±7.6	82.0±25.5	49.1±8.5	7.9±2.3	33.4
	MADiff*	**57.8±16.4**	54.3±5.8	12.2±8.8	19.6±10.1	58.8±7.7	20.7±10.6	94.2±9.3	54.1±9.4	8.6±3.3	42.3
	DOF*	35.0±8.2	29.9±4.9	9.3±9.2	12.5±6.7	54.1±14.5	15.7±6.6	88.8±21.4	23.1±8.0	3.4±4.8	30.2
	MADiTS	52.8±14.9	24.5±6.0	6.7±3.0	**25.8±8.9**	46.7±12.3	21.6±10.5	90.8±22.3	**58.6±5.7**	3.7±1.2	36.8
	INS	46.6±7.8	45.8±8.0	8.1±5.4	18.5±8.9	45.1±14.4	**25.8±6.7**	103.8±19.1	51.9±10.4	8.1±2.8	39.3
	Ours	54.7±17.7	**54.4±6.0**	**13.3±3.5**	20.6±9.5	**59.7±16.6**	25.2±8.2	**111.4±10.8**	51.5±8.0	**10.0±3.0**	**44.5**
OMAR	Original	10.8±15.1	53.3±12.1	21.7±11.1	-1.1±6.4	32.8±7.3	21.1±7.0	63.5±34.8	55.3±8.9	7.1±2.3	29.4
	MADiff*	18.4±25.0	**62.8±8.1**	17.9±5.3	-1.0±6.9	**50.0±10.1**	18.8±7.6	58.0±39.5	59.1±18.0	**11.7±3.4**	32.9
	DoF*	7.1±7.7	41.1±7.0	24.0±7.2	-2.2±13.7	39.4±8.7	18.6±8.0	68.9±26.6	40.9±7.5	10.4±2.6	27.6
	MADiTS	23.9±9.5	41.5±11.3	**28.8±10.8**	3.3±7.6	43.7±12.6	23.5±11.3	53.1±19.3	56.1±3.8	6.3±2.3	31.2
	INS	4.5±5.6	50.9±8.5	18.1±5.5	0.6±8.9	45.4±10.5	21.8±5.1	75.8±29.3	54.4±11.5	7.8±3.6	31.0
	Ours	**24.3±21.7**	60.9±6.9	19.1±7.2	**9.7±4.8**	46.5±20.7	**27.0±10.2**	**109.7±12.4**	**59.5±12.3**	10.3±5.0	**40.8**
MATD3+BC	Original	23.4±11.3	32.2±9.0	3.7±3.7	8.5±7.3	28.1±10.6	20.1±7.2	84.4±31.3	50.6±8.9	4.4±2.3	28.4
	MADiff*	37.2±27.3	49.2±9.4	**13.2±9.2**	7.5±6.4	53.8±18.7	15.8±8.1	82.5±16.7	47.4±10.6	7.1±2.5	34.9
	DoF*	22.4±9.8	32.8±7.6	3.4±1.3	9.6±6.7	49.4±18.6	19.1±8.9	64.5±33.0	26.2±8.5	**9.8±6.0**	26.4
	MADiTS	**39.9±6.2**	22.0±4.6	2.5±1.2	**11.0±3.3**	45.3±10.5	17.4±9.0	75.0±12.3	**60.7±8.8**	1.5±2.5	30.6
	INS	15.7±9.7	43.9±7.0	4.6±3.7	5.6±15.9	45.9±20.7	21.6±8.2	66.5±34.6	49.0±12.4	8.0±1.8	29.0
	Ours	32.8±21.6	**50.8±8.0**	10.9±4.6	7.8±5.1	**54.2±17.9**	**24.7±10.1**	**100.2±8.8**	52.2±10.3	8.2±4.1	**38.0**

Table 2: Test win rate in the SMAC environment (**bold**: best, underline: second best)

Alg	Dataset	3m		2s3z		8m		Average
		Medium	Good	Medium	Good	Medium	Good	
CFCQL	Original	0.75±0.02	0.96±0.04	0.38±0.38	0.40±0.55	0.57±0.52	0.80±0.45	0.64
	MADiff*	0.78±0.27	0.84±0.36	0.31±0.18	0.68±0.23	0.80±0.45	0.87±0.13	0.71
	DoF*	0.89±0.14	0.86±0.09	0.16±0.49	0.76±0.33	0.78±0.25	0.73±0.19	0.70
	MADiTS	0.70±0.14	0.86±0.27	0.30±0.65	0.51±0.14	**0.90±0.08**	0.89±0.12	0.69
	INS	0.82±0.31	0.79±0.46	0.17±0.7	**0.83±0.21**	0.82±0.33	0.85±0.16	0.71
	Ours	**1.00±0.00**	**0.97±0.07**	**0.43±0.58**	0.77±0.09	0.84±0.06	**1.00±0.00**	**0.84**
OMAR	Original	0.71±0.07	0.41±0.24	**0.40±0.55**	0.60±0.55	0.75±0.43	0.75±0.43	0.60
	MADiff*	0.71±0.30	0.53±0.42	0.37±0.11	0.53±0.19	0.89±0.11	0.69±0.38	0.62
	DoF*	0.82±0.18	0.91±0.14	**0.40±0.42**	0.76±0.19	0.87±0.07	0.74±0.28	0.75
	MADiTS	0.72±0.12	0.44±0.40	0.36±0.60	0.40±0.29	0.72±0.26	0.89±0.03	0.59
	INS	0.84±0.20	0.88±0.14	0.36±0.37	0.78±0.35	0.85±0.14	0.81±0.15	0.75
	Ours	**1.00±0.00**	**0.99±0.02**	0.37±0.58	**0.94±0.00**	0.83±0.02	**1.00±0.00**	**0.86**

Practical Implementation. MASTARS leverages five pre-trained models: a diffusion model ϵ_θ for generating observation sequences (built on the MADiff [49] architecture), three dynamics models (a reward function g_ξ, an inverse dynamics model I_ϕ, and a transition model f_φ) that produce the corresponding actions and rewards from a generated observation sequence, and a value function V_ζ. Their training losses are given by Eqs. (2), (3), and (12), with full details in Appendix E.2. The overall pipeline of MASTARS is illustrated in Fig. 2.

In summary, MASTARS is a plug-in diffusion-based framework for multi-agent trajectory augmentation that integrates (i) agent-wise sequential generation with inpainting, (ii) subgoal-based partial generation, and (iii) subgoal-based ordering. By combining all three, MASTARS produces diverse, coordinated, and globally coherent trajectories. The complete algorithm is provided in Appendix E.1.

6 Experiments

Our evaluation focuses on five aspects: (i) performance gains with the augmented dataset, (ii) visual analysis of our generation strategy, (iii) quantitative analysis of the generated dataset in terms of diversity, coordination, and coherence, (iv) reliability of the auxiliary models, and (v) the contribution of key components through an ablation study.

6.1 Experimental Setup

We evaluated MASTARS using three widely adopted offline MARL algorithms: CFCQL [35], OMAR [29], and MA-TD3+BC [7]. These algorithms were tested on the MPE [23], SMAC [32], and SMAC-

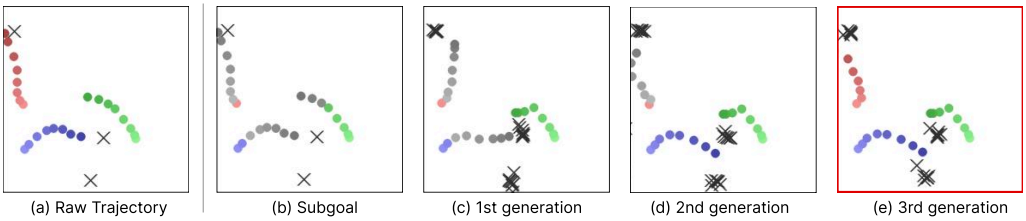


Figure 3: **Visualization of the full trajectory generation process.** Agents are generated sequentially via subgoal-conditioned inpainting, with each panel showing the trajectory after one agent is added, from the raw input (left) to the final result (right). `X` marks are landmark positions reconstructed from each agent's observation.

v2 [4] benchmarks. Detailed information regarding the benchmarks, algorithm configurations, and dataset compositions is provided in Appendix F.

Baselines We compared our method against other data augmentation approaches[1]: 1) **Original**: using only the given offline dataset. 2) **MADiff*** [49]: a diffusion-based trajectory generator with inter-agent attention. 3) **DoF*** [19]: a diffusion-based MARL policy extending the IGM principle to an IGD formulation. 4) **MADiTS** [44]: a stitching approach that constructs trajectories by combining segments. 5) **INS** [6]: a sparse-attention method that explicitly models inter-agent interactions.

6.2 Main Results on Offline MARL Benchmarks

Evaluation on MPE & SMAC benchmarks Tables 1 & 2 demonstrate strong performance across diverse benchmarks and offline MARL algorithms. In the MPE benchmark, MASTARS enhances the average performance of CFCQL, OMAR, and MATD3+BC by 33.2%, 38.8%, and 33.8%, respectively, compared to the original datasets. Notably, it achieves top-two results in 22 of 27 tasks.

Building on these results, the superiority of our framework extends to the SMAC benchmark, which presents more complex coordination challenges. Here, MASTARS improves CFCQL and OMAR by 31.3% and 43.3%, respectively, achieving top-two results in 10 out of 12 scenarios. Most notably, it achieves a perfect 100% win rate in demanding tasks such as 3m (Medium) and 8m (Good).

Extended evaluation on SMAC-v2 benchmark To further evaluate the robustness of our framework in more challenging settings, we conducted extended experiments on the SMAC-v2 benchmark, which is characterized by significantly higher stochasticity and complexity. As summarized in Table 3, MASTARS demonstrates remarkable adaptability to these environments, achieving a substantial average performance boost of 152.6% compared to the original dataset. It reliably ranks within the top-two in 18 out of 20 tasks. Detailed environment descriptions, dataset composition, and full SMAC-v2 results are provided in Appendix G.

Table 3: Average test win rate in SMAC-v2

Dataset	Protoss	Terran	Zerg	Average
Original	0.12	0.22	<u>0.24</u>	0.19
MADiff*	0.15	0.30	0.14	0.21
DoF*	<u>0.23</u>	0.29	0.19	<u>0.25</u>
MADiTS	0.18	0.18	0.19	0.18
INS	0.21	<u>0.33</u>	0.15	<u>0.25</u>
Ours	**0.50**	**0.50**	**0.40**	**0.48**

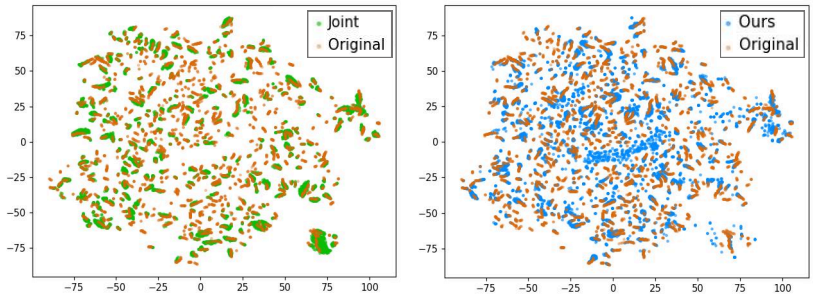
Summary of overall performance These results show that MASTARS consistently outperforms the evaluated augmentation baselines across environments, datasets, and offline MARL algorithms. Implementation details and computational cost analyses are provided in Appendices H and I.

6.3 Visualization Analysis of Our Generation Strategy

Sequential Trajectory Generation. We visualize a trajectory generated by MASTARS to illustrate its sequential construction process. As shown in Fig. 3, agents generate trajectories in the order of their subgoal timesteps in the original trajectory: green, blue, and then red. The green agent, with the latest subgoal, generates first, followed by the blue, and finally the red agent with the earliest subgoal.

By reusing trajectory segments up to the subgoal, MASTARS avoids full trajectory generation while leveraging expert behavior. Moreover, since agents with earlier subgoals offer greater flexibility,

[1]An asterisk (*) indicates methods that are not trajectory generators; we use only their generation components.



(a) Original vs Joint (b) Original vs Ours

Figure 4: t-SNE comparison: original (orange), jointly generated (green), and ours (blue)

generating them later enhances overall diversity. As a result, MASTARS produces realistic and cooperative trajectories that avoid collisions and reach targets (more examples in Appendix K).

Comparison with Multi-Agent Baselines. We visually compare MASTARS with MADiTS in Fig. 5; INS generates transitions rather than trajectories and is thus not visualized. MADiTS exhibits incoherent and uncoordinated movements due to its one-shot refinement, whereas MASTARS produces coherent and well-coordinated trajectories. Additional comparisons are provided in Appendix L.

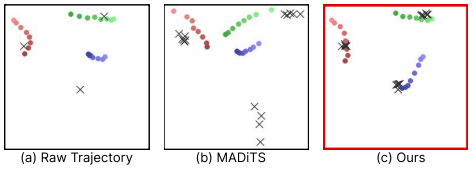


Figure 5: Comparison of MADiTS and Ours

6.4 Quantitative Analysis of Generated Dataset

Diversity. To verify the diversity of trajectories generated by MASTARS, we conduct a dataset analysis using DIRE and t-SNE. First, we adopt the DIRE metric [39], which was proposed to distinguish between original and generated samples. This metric is defined as: $DIRE(x_0) = \|x_0 - R(I(x_0))\|$, where $I(\cdot)$ and $R(\cdot)$ denote the forward and deterministic reverse processes, respectively. Since $R(I(x_0))$ represents the reconstruction of x_0, DIRE measures the reconstruction error with respect to the training distribution. Lower scores indicate closer alignment with the training data, and higher scores reflect greater distributional diversity of the generated trajectories relative to the training set.

As shown in Table 4, MASTARS yields higher DIRE scores than joint generation methods, such as MADiff*, indicating the synthesis of more novel behaviors. This observation is further corroborated by the t-SNE visualization in Fig. 4. While MADiff* overlaps significantly with the original data manifold, MASTARS produces a broader distribution that extends outside the original data manifold. Note that all analyzed samples have passed return-conditioned guidance (Eq. (2)) and transition validation (Eq. (6)), so this broader coverage reflects guided, filtered samples rather than raw generations.

Table 4: Reconstruction error (DIRE)

	Original	MADiff*	Ours
DIRE	0.045	0.086	**0.101**

Coordination and Coherence. To quantify the visual analyses in Figs. 1 and 5, we measure three metrics over all generated MPE trajectories: *collision rate*, the fraction of transitions with an agent collision (coordination); *landmark error*, the within-episode dispersion of each fixed landmark's position reconstructed from every agent's generated observation, i.e., the mean ℓ_2 distance of the reconstructions to their centroid, which is zero for real data (coherence); and *dynamics violation rate*, the fraction of generated transitions that deviate from the true environment dynamics by more than the threshold ϵ in Eq. (6). As shown in Table 5, MASTARS achieves the best result on all three metrics: sequential inpainting improves coordination and coherence, while transition validation removes inconsistent candidates. Together, these results show that MASTARS synthesizes diverse, coordinated, and globally coherent trajectories.

Table 5: Coordination and coherence metrics of generated MPE datasets. INS generates transitions, so its landmark error is undefined. Baselines use looser acceptance thresholds to retain sufficient data.

Method	Collision rate ↓	Landmark error ↓	Dynamics violation ↓
MADiff*	13.3%	0.088	14.2%
DoF*	42.0%	0.276	85.9%
MADiTS	33.2%	0.078	19.0%
INS	16.7%	N/A	93.0%
MASTARS (Ours)	**8.4%**	**0.039**	**2.0%**

Table 6: Prediction errors of auxiliary models on offline and generated data. N/A: no ground-truth actions or values exist for generated data.

Model	Offline	Generated	Model	Offline	Generated
Reward g_ξ	2.3% (1.7%)	4.0% (2.3%)	Transition f_φ	0.9% (0.6%)	0.8% (0.6%)
Inverse dynamics I_ϕ	5.9% (4.3%)	N/A	Value V_ζ (TD error)	0.01	N/A

6.5 Reliability of Auxiliary Models

MASTARS relies on four auxiliary models (g_ξ, I_ϕ, f_φ, V_ζ), raising the concern that their errors may compound. However, none of them is used for recursive rollouts: V_ζ only selects subgoals and the order, f_φ only validates candidates, and g_ξ and I_ϕ only annotate generated observations, so their errors do not accumulate across generation steps.

Table 6 shows that all prediction errors remain within a few percent of the data range, even on generated data. MASTARS is also robust to these errors: V_ζ uses only the relative ranking of values, so value noise of up to 20% still outperforms Original (Appendix J.4); f_φ affects MASTARS only through ϵ, to which performance is robust (Fig. 7a); and reward and action noise of up to 30% keeps the score at 37.4–40.3 (Appendix J.5). Thus, MASTARS is robust to auxiliary-model errors.

6.6 Ablation Studies

We conducted ablation studies to assess the contributions of four key components of MASTARS: (1) partial generation via subgoal conditioning, (2) sequential generation, (3) inpainting, and (4) subgoal-based ordering. All ablation results are averaged over CFCQL, OMAR, and MATD3+BC on the Medium, MD-Replay, and Random datasets of each MPE task.

Since the ordering mechanism is foundational, we tested its necessity by replacing our subgoal-based strategy with reverse and random ordering schemes. As shown in Table 7, removing individual modules ("w/o") or using alternative ordering schemes leads to significant performance degradation.

Table 7: Ablation performance

	Tag	Spread	World	Total
w/o Subgoal	28.11	26.06	45.84	33.34
w/o Sequence	27.35	23.88	52.58	34.60
w/o Inpainting	29.00	27.96	54.86	37.27
Random ordering	28.03	24.11	46.46	32.87
Reverse ordering	27.60	24.31	53.73	35.21
Ours	**35.69**	**30.60**	**57.00**	**41.10**

Additional analyses of hyperparameter sensitivity, synthesis fraction, scalability with respect to agent number and horizon length, and value-estimation sensitivity are provided in Appendix J, and generation cost in Appendix I.

7 Conclusion

In this paper, we showed that effective offline MARL augmentation requires more than realistic joint trajectory modeling. To overcome the limitation of conventional data augmentation methods, we introduced MASTARS, a novel data augmentation method for offline MARL. By decomposing generation agent-wise while using inpainting to preserve joint coherence, and by preserving high-value subgoal prefixes, MASTARS expands the offline dataset in directions that remain coordinated and globally feasible. This combination leads to consistent improvements across multiple offline MARL algorithms and benchmarks, suggesting that structured generative augmentation is a promising direction for improving data-limited multi-agent learning.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2025-00557589, Generative Model Based Efficient Reinforcement Learning Algorithms for Multi-modal Expansion in Generalized Environments), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2026-25510069), and the "Advanced GPU Utilization Support Program" funded by the Government of the Republic of Korea (Ministry of Science and ICT) (No. 02-26-01-0110). The authors declare no competing interests.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=sP1fo2K9DFG`.

[2] Gang Ding, Zeyuan Liu, Zhirui Fang, Kefan Su, Liwen Zhu, and Zongqing Lu. Multi-agent coordination via multi-level communication. *Advances in Neural Information Processing Systems*, 37:118513–118539, 2024.

[3] Omar Elharrouss, Noor Almaadeed, Somaya Al-Máadeed, and Younes Akbari. Image inpainting: A review. *CoRR*, abs/1909.06399, 2019.

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

[5] Claude Formanek, Asad Jeewa, Jonathan P. Shock, and Arnu Pretorius. Off-the-grid MARL: a framework for dataset generation with baselines for cooperative offline multi-agent reinforcement learning. *CoRR*, abs/2302.00521, 2023. doi: 10.48550/ARXIV.2302.00521. URL `https://doi.org/10.48550/arXiv.2302.00521`.

[6] Yuqian Fu, Yuanheng Zhu, Jian Zhao, Jiajun Chai, and Dongbin Zhao. Ins: Interaction-aware synthesis to enhance offline multi-agent reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[7] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 20132–20145, 2021.

[8] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[9] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *CoRR*, abs/2207.12598, 2022. doi: 10.48550/ARXIV.2207.12598.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[11] Kun Hu, Muning Wen, Xihuai Wang, Shao Zhang, Yiwei Shi, Minne Li, Minglong Li, and Ying Wen. Pmat: Optimizing action generation order in multi-agent reinforcement learning. *arXiv preprint arXiv:2502.16496*, 2025.

[12] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9902–9915. PMLR, 2022.

[13] Jeewon Jeon, Woojun Kim, Whiyoung Jung, and Youngchul Sung. Maser: Multi-agent reinforcement learning with subgoals generated from experience replay buffer. In *International conference on machine learning*, pages 10041–10052. PMLR, 2022.

[14] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

[15] Woojun Kim and Katia Sycara. B3c: A minimalist approach to offline multi-agent reinforcement learning. *arXiv preprint arXiv:2501.18138*, 2025.

[16] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[17] Jaewoo Lee, Sujin Yun, Taeyoung Yun, and Jinkyoo Park. GTA: generative trajectory augmentation with guidance for offline reinforcement learning. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[18] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

[19] Chao Li, Ziwei Deng, Chenxing Lin, Wenqi Chen, Yongquan Fu, Weiquan Liu, Chenglu Wen, Cheng Wang, and Siqi Shen. Dof: A diffusion factorization framework for offline multi-agent reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. Diffstitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=phGHQOKmaU`.

[21] Wenbo Li, Xin Yu, Kun Zhou, Yibing Song, Zhe Lin, and Jiaya Jia. Image inpainting via iteratively decoupled probabilistic modeling, 2023. URL `https://arxiv.org/abs/2212.02963`.

[22] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 20725–20745. PMLR, 2023.

[23] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6379–6390, 2017.

[24] Cong Lu, Philip J. Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[25] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[26] Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported value regularization for offline reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[27] Jihwan Oh, Sungnyun Kim, Gahee Kim, Sunghwan Kim, and Se-Young Yun. Diffusion-based episodes augmentation for offline multi-agent reinforcement learning. *CoRR*, abs/2408.13092, 2024. doi: 10.48550/ARXIV.2408.13092. URL `https://doi.org/10.48550/arXiv.2408.13092`.

[28] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer Briefs in Intelligent Systems. Springer, 2016. ISBN 978-3-319-28927-4. doi: 10.1007/978-3-319-28929-8.

[29] Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 17221–17237. PMLR, 2022.

[30] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[31] 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.

[32] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor, editors, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pages 2186–2188. International Foundation for Autonomous Agents and Multiagent Systems, 2019. URL `http://dl.acm.org/citation.cfm?id=3332052`.

[33] 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.

[34] Kajetan Schweighofer, Marius-Constantin Dinu, Andreas Radler, Markus Hofmarcher, Vihang Prakash Patil, Angela Bitto-Nemling, Hamid Eghbal-zadeh, and Sepp Hochreiter. A dataset perspective on offline reinforcement learning. In Sarath Chandar, Razvan Pascanu, and Doina Precup, editors, *Conference on Lifelong Learning Agents, CoLLAs 2022, 22-24 August 2022, McGill University, Montréal, Québec, Canada*, volume 199 of *Proceedings of Machine Learning Research*, pages 470–517. PMLR, 2022.

[35] Jianzhun Shao, Yun Qu, Chen Chen, Hongchang Zhang, and Xiangyang Ji. Counterfactual conservative Q learning for offline multi-agent reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[36] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[37] Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/26cce1e512793f2072fd27c391e04652-Abstract-Conference.html.

[38] Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[39] Zhendong Wang, Jianmin Bao, Wengang Zhou, Weilun Wang, Hezhen Hu, Hong Chen, and Houqiang Li. Dire for diffusion-generated image detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22445–22455, 2023.

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

[41] Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 10299–10312, 2021.

[42] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[43] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[44] Lei Yuan, Yuqi Bian, Lihe Li, Ziqian Zhang, Cong Guan, and Yang Yu. Efficient multi-agent offline coordination via diffusion-based trajectory stitching. In *The Thirteenth International Conference on Learning Representations*, 2025.

[45] Junjie Zhang, Jiafei Lyu, Xiaoteng Ma, Jiangpeng Yan, Jun Yang, Le Wan, and Xiu Li. Uncertainty-driven trajectory truncation for data augmentation in offline reinforcement learning. In *ECAI 2023*, pages 3018–3025. IOS Press, 2023.

[46] Yang Zhang, Xinran Li, Jianing Ye, Delin Qu, Shuang Qiu, Chongjie Zhang, Xiu Li, and Chenjia Bai. Revisiting multi-agent world modeling from a diffusion-inspired perspective. *arXiv preprint arXiv:2505.20922*, 2025.

[47] Zijie Zhao, Honglei Guo, Shengqian Chen, Kaixuan Xu, Bo Jiang, Yuanheng Zhu, and Dongbin Zhao. Empowering multi-robot cooperation via sequential world models. *arXiv preprint arXiv:2509.13095*, 2025.

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

[49] 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. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

A Broader Impact

Our algorithm can generate cooperative trajectories from small datasets, reducing the need for expensive data collection in real-world applications. This has significant potential in domains where reinforcement learning is applicable but data acquisition is costly, such as autonomous driving, warehouse robotics, and surgical training simulators. By lowering the data requirements, our method enables faster prototyping and broader accessibility of RL-based systems. However, careful validation is necessary to ensure that the generated data does not introduce unrealistic behaviors or biases when deployed in safety-critical environments.

B Limitations and Future Works

The sequential generation process in MASTARS performs one inpainting pass per agent, so its sampling cost grows with the number of agents. Although this cost is incurred only once when constructing the augmented dataset and is offset by a higher acceptance rate under strict validation, it may become a practical burden in settings with very many agents; reducing it, e.g., by generating weakly interacting agents in parallel, is a promising direction. In addition, it would be beneficial to explore alternative approaches to subgoal generation, such as learning subgoal distributions or leveraging a combination of joint and individual value functions [13].

C Trajectory Generation in Single-Agent RL

In offline RL, diffusion models are trained to generate entire trajectory segments by treating each segment as if it were a 2D "image." Concretely, they learn the distribution $p_\theta\big(\tau_{t:t+H-1}\big)$, by minimizing the DDPM loss in Eq. (1). To do so, each trajectory segment is reshaped into a matrix of size $(d_s + d_a) \times H$ by stacking state and action vectors as rows, where d_s is the state dimension, d_a is the action dimension, and H is the segment length to be generated. The diffusion model then generates this matrix just as it would generate an image:

$$\tau = \begin{bmatrix} \mathbf{s}_t & \mathbf{s}_{t+1} & \cdots & \mathbf{s}_{t+H-1} \\ \mathbf{a}_t & \mathbf{a}_{t+1} & \cdots & \mathbf{a}_{t+H-1} \end{bmatrix}. \tag{9}$$

By framing trajectories in this way, the model naturally outputs new trajectory "images" whose statistical properties match those of the offline dataset.

D Trajectory Generation in MADiff

Given a sequence of observations $\tau = (\boldsymbol{o}_{t:t+H-1})$, MADiff [49] proposes two trajectory generation strategies: Independent Trajectory Generation and Joint Trajectory Generation.

Independent Trajectory Generation. In the independent scheme, we adopt a decentralized execution scheme in MADiff. Each agent i generates its trajectory based solely on its current local observation, without access to the observations or plans of other agents. The diffusion process begins with a noisy trajectory conditioned on agent i's current observation, represented as $\tilde{\boldsymbol{x}}_K = [[\tilde{x}^1_{K,t}, \ldots, \tilde{x}^1_{K,t_e}], \ldots, [o^i_{t_s}, \ldots, \tilde{x}^i_{K,t_e}], \ldots, [\tilde{x}^N_{K,t}, \ldots, \tilde{x}^N_{K,t_e}]]$, with $\tilde{x}^j_{K,t'} \sim \mathcal{N}(0,1)$, and $t_e = t_s + H - 1$.

Joint Trajectory Generation. The joint generation scheme adopts a centralized control setup as in MADiff, where the trajectories of all agents are generated simultaneously, conditioned on the current joint observation $\boldsymbol{o}_t$. The initial noisy trajectory is constructed as $\tilde{\boldsymbol{x}}_K = [[o^1_{t_s}, \ldots, \tilde{x}^1_{K,t_e}], \ldots, [o^i_{t_s}, \ldots, \tilde{x}^i_{K,t_e}], \ldots, [o^N_{t_s}, \ldots, \tilde{x}^N_{K,t_e}]]$, again with $\tilde{x}^j_{K,t'} \sim \mathcal{N}(0,1)$, and $t_e = t_s + H - 1$.

Simple Sequence Trajectory Generation. Since MADiff does not propose a sequential generation scheme, we propose a simple, naive sequential trajectory generation scheme as follows. In the simple sequence generation scheme, trajectories are generated sequentially following a predefined order of agents (we use the same order as our subgoal-based ordering to visualize our motivation). For each agent $i \in \{1, \ldots, N\}$, the diffusion process is performed to generate its specific trajectory while keeping others fixed. The initial noisy trajectory $\tilde{\boldsymbol{x}}_K$ at step i is constructed as follows: $\tilde{\boldsymbol{x}}_K = [[\hat{x}^1_t, \ldots, \hat{x}^1_{t_e}], \ldots, [o^i_t, \ldots, \tilde{x}^i_{K,t_e}], \ldots, [\tilde{x}^N_{K,t}, \ldots, \tilde{x}^N_{K,t_e}]]$, where $\hat{x}^j$ for $j < i$ represents the fixed trajectory already generated in previous steps, and $\tilde{x}^j_{K,t'} \sim \mathcal{N}(0,1)$ for $j \geq i$ denotes Gaussian noise for the current and subsequent agents. This procedure is repeated until the entire sequence of agents is finalized, resulting in a complete joint trajectory.

E Pseudocode and Loss functions

This section presents the algorithm in detail using pseudocode and outlines the loss functions of the pretrained models used for generation.

E.1 Pseudocode

Algorithm 1 Agent-Wise Sequential Trajectory Generation

Input: Original offline dataset D_{org}, Pretrained diffusion model ϵ_θ, Inverse dynamics model I_ϕ, Transition model f_φ, Reward model g_ξ, Value function V_ζ, Number of episodes to generate N_{episode}, Number of denoising steps K, Pre-defined schedule $\bar{\alpha}_k$
Output: Augmented dataset D_{aug}
Initialize $D_{\text{aug}} \leftarrow \emptyset$
while number of episodes in $D_{\text{aug}} < N_{\text{episode}}$ **do**
  Sample joint sequence $\tau^0 = (\mathbf{o}_t, \ldots, \mathbf{o}_{t+H-1}) \sim D_{\text{org}}$
  // Subgoal-Based Ordering
  Calculate Order$(\kappa(1), \ldots, \kappa(N))$ from τ^0 via Eq. (8)
  `Input_traj` $\leftarrow \tau^0$
  // Agent-Wise Sequential Generation
  for each agent i in Order$(\kappa(1), \ldots, \kappa(N))$ **do**
    Select g^i as the observation in `Input_traj` with the highest value V_ζ.
    Generate mask m with g^i via Eq. (7)
    // Initialize at Step K
    $\tilde{\tau}_K^{\text{posterior}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\tilde{\tau}_K^{\text{prior}} \sim \mathcal{N}(\sqrt{\bar{\alpha}_K}\texttt{Input_traj}, (1-\bar{\alpha}_K)\mathbf{I})$
    $\tilde{\tau}_K \leftarrow m \odot \tilde{\tau}_K^{\text{posterior}} + (1-m) \odot \tilde{\tau}_K^{\text{prior}}$
    for $k = K$ **down to** 1 **do**
      // Repaint with Alg. 2
      $\tilde{\tau}_{k-1} \leftarrow \textsc{Repaint}(\tilde{\tau}_k, m, \bar{\alpha}_k, \epsilon_\theta)$
    end for
    `Input_traj` $\leftarrow \tilde{\tau}_0$
  end for
  `Seq_traj` $\leftarrow \tilde{\tau}_0$
  // Post-Processing
  Compute actions/rewards for `Seq_traj` using I_ϕ, g_ξ
  if transition validity (Eq. (6)) is satisfied **then**
    $D_{\text{aug}} \leftarrow D_{\text{aug}} \cup \{\texttt{Seq_traj}\}$
  end if
end while
return D_{aug}

Algorithm 2 Repaint

Input: Trajectory from previous denoised step $\tilde{\tau}_k$, Mask m, Gaussian model ϵ_θ, pre-defined scheduler $\bar{\alpha}_k$
Output: $\tilde{\tau}_{k-1}$
$\tilde{\tau}_{k-1}^{\text{prior}} \leftarrow$ Add noise on $\tilde{\tau}_k$ {Eq. (5c)}
$\tilde{\tau}_{k-1}^{\text{posterior}} \leftarrow$ Denoise from $\tilde{\tau}_k$ {Eq. (5b)}
$\tilde{\tau}_{k-1} \leftarrow m \odot \tilde{\tau}_{k-1}^{\text{posterior}} + (1-m) \odot \tilde{\tau}_{k-1}^{\text{prior}}$ {Eq. (5a)}
return $\tilde{\tau}_{k-1}$

E.2 Loss functions

Diffusion model Training. The model is conditioned on the normalized return $R(\tau)$ to generate observation sequences of length H, using the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{k,\tau_0\in\mathcal{D},\epsilon,R(\tau),\beta}\Big[\big\|\epsilon - \epsilon_\theta\big(\tau_k, k,\ (1-\beta)R(\tau_0) + \beta\emptyset\big)\big\|^2\Big]. \tag{10}$$

where $\tau := (\boldsymbol{o}_{t:t+H-1})$ is the observation sequence, $\beta\sim\mathrm{Bern}(\lambda)$ is a Bernoulli mask, and $\epsilon\sim\mathcal{N}(0, I)$

Dynamic models Training. To reconstruct full trajectories including actions and rewards, we train three auxiliary models: an inverse dynamics model $I_\phi(o^i, o^{i'})$ for each agent i, a reward model $g_\xi(\boldsymbol{o}, \boldsymbol{a})$, and a transition model $f_\varphi(o^i, a^i)$. These models are optimized jointly to ensure consistency with environment dynamics, using the following objective:

$$\mathcal{L}(\xi,\phi) = \mathbb{E}_{\boldsymbol{o},\boldsymbol{a},r,\boldsymbol{o}'\sim D}\Big[\big(r - g_\xi(\boldsymbol{o},\boldsymbol{a})\big)^2 + \sum_{i=1}^{N}\Big(\big(a^i - I_\phi(o^i, o^{i'})\big)^2 + (o^{i'} - f_\varphi(o^i, a^i))^2\Big)\Big]. \tag{11}$$

Value function Training. To mitigate potential estimation bias in the value function, which is critical for determining subgoals, we employ a ***SARSA***-style objective focused on observed transitions. This approach stabilizes value estimation by reducing extrapolation errors. Furthermore, the generation process remains robust to absolute value bias as it relies on relative ranking for prioritization. Finally, to ensure subgoals reflect the global team return, individual value functions are trained using the collective return, as defined below:

$$L(\zeta) = \mathbb{E}_{\boldsymbol{o},r,\boldsymbol{o}'\sim D}\Big[\sum_{i=1}^{N}\big(V_\zeta(o^i) - (r + \gamma V_\zeta(o^{i'}))\big)^2\Big]. \tag{12}$$

F Experimental Details

F.1 Benchmarks

MPE. *Multi-Agent Particle Environment (MPE [23])* is a standard benchmark for continuous multi-agent coordination tasks, featuring simple physical dynamics and inter-agent interactions in a 2D space. We evaluated our method on three representative tasks: `Tag`, `Spread`, and `World`, each requiring distinct coordination. Fig. 6 illustrates the three environments, with detailed descriptions provided below.

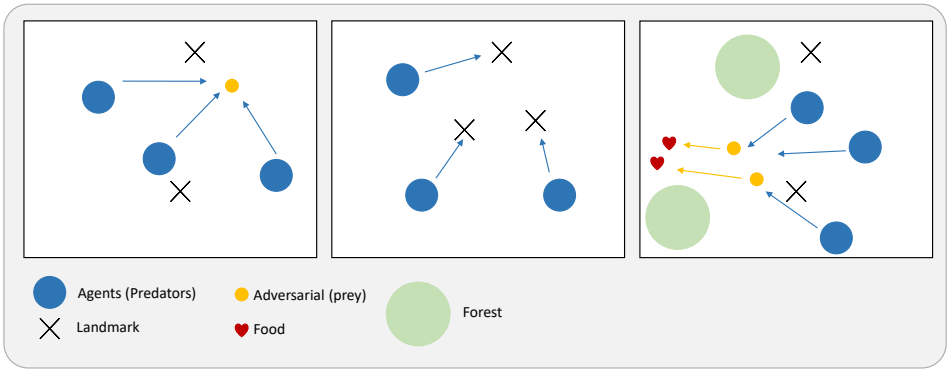


Figure 6: Visualization of MPE tasks. From left to right: Simple Tag, Simple Spread, Simple World.

- `Simple Tag`: This task involves three slower agents (predators) attempting to capture a faster adversarial agent (prey). Due to their speed disadvantage, predators must coordinate their movements to surround and capture the prey, as individual pursuit is ineffective.
- `Simple Spread`: This environment consists of three agents and three fixed landmarks. Agents are rewarded based on their proximity to the landmarks and penalized for collisions with other agents. The objective is to spread out and cover all landmarks while avoiding collisions, requiring spatial coordination.
- `Simple World`: This task involves three cooperative agents and two adversaries. Cooperative agents aim to capture the adversaries, while the adversaries simultaneously try to collect food items scattered throughout the environment and avoid being captured.

SMAC. *StarCraft Multi-Agent Challenge (SMAC)* [33] is a high-dimensional, discrete-action cooperative benchmark built on the StarCraft II engine. In this benchmark, agents must coordinate to defeat enemies using partial observations and task-specific strategies. We evaluated our method on three commonly used SMAC scenarios: `3m`, `2s3z`, and `8m`.

Table 8: Agent and enemy compositions for SMAC tasks

Task	Agents	vs	Enemy
3m	3 marines	vs	3 marines
2s3z	2 stalkers and 3 zealots	vs	2 stalkers and 3 zealots
8m	8 marines	vs	8 marines

SMAC-v2. Compared to the original SMAC, which utilizes fixed unit types and deterministic starting configurations, SMAC-v2 [4] introduces significantly higher variability. Specifically, both ally and enemy unit compositions are sampled from probability distributions, and initial agent placements are randomized via reflections and spatial perturbations of the map layout. These features increase stochasticity and diversity across episodes, providing a more rigorous testbed for assessing robustness and generalization in multi-agent coordination. A detailed explanation is provided in Appendix G.

F.2 Evaluation Algorithms

This section outlines the evaluation algorithms used for comparison on the original and augmented datasets.

Counterfactual Conservative Q-Learning (CFCQL) [35]. CFCQL applies conservative regularization to each agent's Q-function individually in a counterfactual manner. That is, it marginalizes over the actions of other agents while fixing the agent's own action. This design enables each agent to estimate a conservative Q-value for its own actions while still accounting for the behaviors of others through the environment dynamics. The training objective is formulated as:

$$\hat{Q}_{k+1} \leftarrow \arg\min_{Q} \ \alpha \left[\sum_{i=1}^{n} \mathbb{E}_{s\sim\mathcal{D},\, a^{-i}\sim\beta^{-i}} \mathbb{E}_{a^i\sim\mu^i} \left[Q(s,a) \right] - \mathbb{E}_{s\sim\mathcal{D},\, a\sim\beta} \left[Q(s,a) \right] \right] + \hat{\mathcal{E}}_{\mathcal{D}}(\pi, Q, k). \tag{13}$$

where $\hat{\mathcal{E}}_{\mathcal{D}}(\pi, Q, k)$ denotes the temporal-difference (TD) error used to fit the Q-function.

Offline Multi-Agent RL with Actor Rectification (OMAR) and MATD3+BC [29]. OMAR introduces a hybrid optimization strategy that combines first-order policy gradients with zeroth-order optimization through population-based search. The actor loss is defined as:

$$\mathbb{E}_{\mathcal{D}_i} \left[(1-\lambda) Q_i(o_i, \pi_i(o_i)) - \lambda \left\| \pi_i(o_i) - \hat{a}_i \right\|^2 \right]. \tag{14}$$

where the candidate actions $\hat{a}_i$ are sampled from a population distribution that is adaptively shaped based on the critic Q_i. If the candidate actions $\hat{a}_i$ are instead sampled directly from the offline dataset, the algorithm reduces to MATD3+BC.

F.3 Dataset Composition

We utilize a combination of established offline MARL repositories and custom-generated datasets to evaluate MASTARS across varying levels of environmental complexity.

MPE and SMAC. For the Multi-Agent Particle Environment (MPE) and the original StarCraft Multi-Agent Challenge (SMAC), we adopted the standardized datasets provided by off-the-grid MARL (og-marl) [5]. To effectively investigate the effects of the augmented data, we used a smaller dataset compared to the original offline dataset. The dataset size for each task is as follows: each MPE task contains 2,000 episodes with a trajectory length of 25, and each SMAC task contains approximately 250,000 transition samples (s, a, r, s').

SMAC-v2. Unlike the original SMAC, for which standardized datasets are readily available, standardized offline datasets for SMAC-v2 are not yet widely available in existing repositories like og-marl. To address this, we generated custom trajectory datasets for various tasks, each consisting of 2,000 episodes with a fixed horizon of 200 steps. These trajectories were collected using the QMIX algorithm [31] implemented within the PyMARL framework. For a comprehensive description of the generation process and hyperparameters, please refer to Appendix G.

G SMAC-v2

This appendix provides additional details on the environment, the composition of the dataset, and extended experimental results on the more challenging SMAC-v2 [4] benchmark to further evaluate the generalizability of MASTARS beyond the settings considered in the main experiments.

Compared to SMAC, which uses fixed unit types and deterministic starting configurations, SMAC-v2 introduces substantially greater variability: both ally and enemy unit compositions are sampled from probability distributions, and initial agent placements are randomized through reflections and spatial perturbations of the map layout. These features make SMAC-v2 more stochastic and diverse across episodes, providing a stronger testbed for assessing robustness and generalization in multi-agent coordination.

Table 9 summarizes the unit-type sampling distributions used to generate our datasets for each faction (Protoss, Terran, and Zerg). These stochastic unit compositions, combined with randomized starting positions, provide diverse combat scenarios that challenge the robustness and generalization ability of multi-agent reinforcement learning methods.

Table 9: Unit-type sampling probabilities and starting-position distributions for SMAC-v2 datasets.

Faction	Unit Type	Probability	Start Position (p=0.5)
Protoss	Stalker	0.45	Surrounded / Reflected
	Zealot	0.45	
	Colossus	0.10	
Terran	Marine	0.45	Surrounded / Reflected
	Marauder	0.45	
	Medivac	0.10	
Zerg	Zergling	0.45	Surrounded / Reflected
	Baneling	0.45	
	Hydralisk	0.10	

To construct the SMAC-v2 datasets, we collected 2,000 episodes for each task, with all episodes having a fixed horizon of 200 steps. To obtain these trajectories, we used the QMIX algorithm [31], a widely adopted baseline in multi-agent reinforcement learning, implemented in the PyMARL framework, which is a widely adopted baseline in multi-agent reinforcement learning.

Due to the increased stochasticity and difficulty of SMAC-v2, the collected datasets do not achieve near-expert performance (i.e., a test win rate close to 1.0). Instead, we provide two dataset qualities, medium and good, reflecting different levels of policy competence under these challenging settings. Detailed statistics for each task, including the number of trajectories, average return, and test win rate, are reported in Table 10.

Table 10: Dataset statistics for SMAC-v2 tasks.

Task	Quality	# Trajectories	Average return	Average win rate
protoss 3_vs_3	medium	2000	16.44	0.42
	good	2000	18.16	0.58
protoss 5_vs_5	medium	2000	17.98	0.41
	good	2000	19.38	0.64
terran 3_vs_3	medium	2000	11.6	0.44
	good	2000	16.2	0.63
terran 5_vs_5	medium	2000	13.09	0.4
	good	2000	12.26	0.8
zerg 3_vs_3	medium	2000	9.75	0.43
	good	2000	13.33	0.57

Table 11 summarizes the performance of all methods on the SMAC-v2 tasks. MASTARS consistently outperforms existing baselines across both CFCQL and OMAR. Notably, the synthetic data generated

by MASTARS leads to substantial performance improvements, achieving a 143.7% increase under CFCQL and a 189.1% increase under OMAR. Furthermore, MASTARS ranks within the top-two methods on 8 out of 10 tasks under CFCQL, and achieves a top-two ranking on all tasks under OMAR, reflecting its robustness across diverse scenarios and evaluation settings.

These results suggest that MASTARS effectively captures the underlying distribution of complex multi-agent interactions, even in highly stochastic environments where standard offline methods often struggle with distributional shift. The consistent performance gain across different base algorithms (CFCQL and OMAR) further demonstrates that our framework provides a universally beneficial data augmentation strategy that is not tied to a specific policy optimization objective.

Table 11: Performance on SMAC-v2 tasks (**bold**: best, underline: second best). Mean ± std.

		Protoss 3_vs_3		Protoss 5_vs_5		Terran 3_vs_3		Terran 5_vs_5		Zerg 3_vs_3		
Alg	**Dataset**	Medium	Good	Medium	Good	Medium	Good	Medium	Good	Medium	Good	Average
CFCQL	Original	0±0	0.31±0.05	0.14±0.07	0.18±0.07	0.23±0.16	0.40±0.55	0.15±0.25	0.15±0.21	0.20±0.45	0.30±0.17	0.21
	MADiff*	0.26±0.16	0.27±0.15	0.06±0.09	0.10±0.24	0.39±0.34	0.56±0.44	0.23±0.04	0.14±0.19	0.10±0.10	0.25±0.18	0.24
	DoF*	**0.59±0.38**	0.46±0.51	0.29±0.42	0.14±0.09	**0.52±0.46**	0.39±0.15	0.18±0.17	0.32±0.41	0.25±0.43	0.25±0.43	0.34
	MADiTS	0.34±0.39	0.33±0.41	0.09±0.07	0.05±0.06	0.20±0.45	0.36±0.07	0.28±0.41	0.02±0.04	**0.26±0.43**	0.28±0.12	0.22
	INS	0.16±0.13	0.40±0.52	0.42±0.52	0.20±0.45	0.44±0.25	0.45±0.36	**0.40±0.55**	0.45±0.51	0.23±0.09	0.20±0.45	0.34
	Ours	0.50±0.52	**0.48±0.71**	**0.50±0.71**	**0.28±0.45**	0.21±0.07	**0.91±0.00**	0.18±0.09	**1.00±0.00**	**0.26±0.04**	**0.70±0.50**	**0.50**
OMAR	Original	0±0	0.13±0.07	0.10±0.06	0.06±0.04	0.12±0.07	0.28±0.22	0.31±0.40	0.13±0.12	0.22±0.43	**0.21±0.19**	0.16
	MADiff*	0.15±0.08	0.16±0.13	0.21±0.44	0.01±0.01	0.48±0.48	**0.37±0.36**	0.11±0.07	0.08±0.08	0.06±0.06	0.12±0.09	0.18
	DoF*	0.12±0.1	0.06±0.1	0.08±0.09	0.06±0.09	0.05±0.1	0.32±0.39	0.30±0.40	0.24±0.43	0.20±0.45	0.03±0.05	0.15
	MADiTS	0.25±0.43	0.06±0.07	0.12±0.07	0.21±0.44	0.20±0.45	0.10±0.08	0.20±0.45	0.03±0.05	0.13±0.03	0.07±0.03	0.14
	INS	0.08±0.07	0.20±0.45	0.20±0.45	0±0	0.21±0.04	0.18±0.05	0±0	**0.45±0.48**	0.14±0.23	0±0	0.15
	Ours	**0.55±0.64**	**0.65±0.51**	**0.39±0.53**	**0.61±0.45**	**0.52±0.41**	0.31±0.55	**0.55±0.36**	0.29±0.48	**0.46±0.06**	0.18±0.12	**0.45**

H Implementation Details and Hyperparameters

We present the implementation details and hyperparameter settings that affect the performance and stability of our models during offline training.

Architectural and Experimental Setup. For the diffusion model, we adopt the same hyperparameter configuration as MADiff [49]. All auxiliary networks, including the value function, inverse dynamics model, reward model, and transition model, are implemented as two-layer multilayer perceptrons (MLPs) with ReLU activations and 256 hidden units per layer. In addition, the models, such as the diffusion model, value function, and dynamic models, were trained over 100,000 time steps. All experiments were conducted with five random seeds to ensure statistical reliability with the main results reported as mean $\pm$ standard deviation.

Fairness of Comparison. All augmentation methods are compared under matched budgets. Each method generates 1,000 episodes; INS is matched by the total number of transitions, since it synthesizes transitions rather than trajectories. MADiff*, DoF*, MADiTS, and MASTARS share the same post-processing filter (Eq. (6)), the same auxiliary models, and the same number of diffusion steps, whereas INS retains its native synthesis and filtering protocol. All methods use the same training and tuning budget, including the same search space for the acceptance threshold ϵ, and we report the best validation result of each method. Because the amount of accepted augmented data is fixed, a looser threshold does not provide more training data to a baseline. All other settings follow the recommended hyperparameters in the original papers.

Hyperparameters on Sequential Inpainting. Across all experiments, including MPE, SMAC, and SMAC-v2, we generate 1,000 augmented trajectories, each consisting of 8 timesteps (i.e., $H = 8$). The diffusion model performs 200 denoising steps during sampling for both MADiff* and MASTARS. For MADiTS, we follow the original hyperparameter settings from the paper for training and sampling, except for the adaptive threshold and the number of generated trajectories, which are adjusted to ensure a fair comparison.

Hyperparameters for the Post-Processing. This section describes the threshold values used in the post-processing module to filter out invalid transitions based on Eq. (6).

The main results on the MPE environment (Table 1) are obtained using task-specific adaptive thresholds for all baselines, including our method, as specified in Table 12. Since INS does not employ thresholds for trajectory generation, we exclude it from this setting.

For the ablation studies, we use the same thresholds as those reported in Table 12, except for the w/o subgoal variant. When generating trajectories without subgoals, applying low ϵ values for certain tasks results in overly strict acceptance criteria, leading to no trajectories being generated. To address this issue, we adopt slightly higher ϵ values for Tag Random, World Medium, World MD-Replay, and World Random, setting them to 0.03, 0.004, 0.02, and 0.02, respectively.

For the SMAC and SMAC-v2 benchmarks, MADiff*, DoF*, and MADiTS, use higher thresholds than MASTARS, as detailed in Tables 13 and 14, since they generate entire trajectories, resulting in samples that deviate more significantly from the original dataset distribution. In contrast, our method only generates trajectory segments after the subgoal timestep, enabling the use of lower thresholds and stricter acceptance criteria.

Table 12: MPE thresholds in all algorithms

Task \ Algo	**MADiff***	**DoF***	**MADiTS**	**Ours**
Tag-Medium	0.03	0.02	0.03	0.008
Tag-MD-Replay	0.03	0.02	0.03	0.006
Tag-Random	0.04	0.03	0.03	0.007
Spread-Medium	0.02	0.02	0.03	0.01
Spread-MD-Replay	0.02	0.02	0.03	0.004
Spread-Random	0.03	0.02	0.03	0.001
World-Medium	0.04	0.04	0.01	0.001
World-MD-Replay	0.02	0.04	0.03	0.003
World-Random	0.04	0.04	0.03	0.009

Table 13: SMAC thresholds in all algorithms

Task \ Algo	**MADiff***	**DoF***	**MADiTS**	**Ours**
3m-Medium	0.3	0.05	1.0	0.3
3m-Good	0.3	0.1	1.0	0.002
2s3z-Medium	0.7	0.3	2.0	0.09
2s3z-Good	0.7	0.4	2.0	0.009
8m-Medium	1.0	0.4	2.0	0.2
8m-Good	1.0	0.8	2.0	0.3

Table 14: SMAC-v2 thresholds in all algorithms

Task \ Algo	**MADiff***	**DoF***	**MADiTS**	**Ours**
Protoss 3_vs_3-Medium	1.0	0.5	1.0	1.0
Protoss 3_vs_3-Good	1.0	0.8	1.2	0.02
Protoss 5_vs_5-Medium	1.0	0.5	1.0	0.9
Protoss 5_vs_5-Good	1.0	0.8	1.2	0.9
Terran 3_vs_3-Medium	1.0	0.5	0.9	0.02
Terran 3_vs_3-Good	1.0	0.7	1.2	1.0
Terran 5_vs_5-Medium	1.0	0.5	0.9	0.6
Terran 5_vs_5-Good	1.0	0.8	1.2	0.1
Zerg 3_vs_3-Medium	1.0	0.5	0.9	0.7
Zerg 3_vs_3-Good	1.0	0.6	0.8	0.01

I Computational Cost

All experiments, including model training and data generation, were conducted on a single machine with 8 NVIDIA GeForce RTX 3090 GPUs, 96 logical CPU cores, and Linux (kernel 5.15), using Python 3.8 with CUDA acceleration. Only the wall-clock timings reported in Tables 15–18 were measured separately on a single RTX 4090 with batch size 256 and $K = 200$ denoising steps.

I.1 Training Cost

To evaluate computational efficiency, we compared the training time of MASTARS against the MADiTS baseline on the MPE Simple Spread (MD-Replay) task on a single RTX 4090. Both methods were trained for 100K steps. MADiTS required approximately 3 hours and 7 minutes (187 minutes), while MASTARS completed training in 2 hours and 20 minutes (140 minutes), achieving a $\sim 25\%$ reduction in training time under identical conditions. Since MASTARS and MADiff have the same training architecture, MADiff incurs the same training cost as MASTARS.

The four auxiliary models (reward, inverse dynamics, transition, and value) are lightweight two-layer MLPs (Appendix H) and are trained jointly with the diffusion model in the same training loop rather than in separate runs. As summarized in Table 15, they add only 0.3M parameters, about 0.5% of the 64.3M-parameter diffusion model, so their training overhead is negligible.

Table 15: Training cost of MASTARS on a single RTX 4090

Training block	Parameters	Measured training
Diffusion model	64.3M	100K steps: 2.33 h (0.084 s/step)
Four auxiliary models	0.3M (0.5%)	Same training loop

I.2 Generation Cost

MASTARS performs N agent-wise inpainting stages of K denoising steps, so its per-candidate sampling cost grows with the number of agents. Since rejected candidates never enter the dataset, however, the practical cost is the *time per accepted trajectory* (candidate latency divided by the acceptance rate). Table 16 reports it for joint, simple sequential, and MASTARS generation (all built on MADiff; Appendix D) under three acceptance thresholds ϵ.

Table 16: Time per accepted trajectory in MPE under different acceptance thresholds ϵ on a single RTX 4090. ∞: no candidate accepted. **Bold**: fastest per setting.

Env	Method	$\epsilon = 0.001$	$\epsilon = 0.005$	$\epsilon = 0.01$
Spread	Joint	0.50s	**0.05s**	**0.04s**
	Simple-seq	1.52s	0.13s	0.12s
	MASTARS	**0.26s**	0.13s	0.12s
Tag	Joint	∞	0.55s	**0.12s**
	Simple-seq	∞	1.36s	0.32s
	MASTARS	**0.51s**	**0.26s**	0.19s
World	Joint	∞	0.43s	**0.15s**
	Simple-seq	∞	1.18s	0.48s
	MASTARS	**0.53s**	**0.29s**	0.21s

Joint generation is fastest at the loose threshold $\epsilon = 0.01$, but its acceptance rate collapses as ϵ tightens: at $\epsilon = 0.001$ it accepts no candidate on Tag and World. MASTARS matches or outperforms simple sequential generation at every threshold owing to its higher acceptance rate, and is thus the most efficient option whenever validation is strict.

I.3 Scaling with the Number of Agents and Horizon

Table 17 reports the per-candidate sampling latency in SMAC as the number of agents grows (3m, 5m_vs_6m, 8m, and 25m with 3, 5, 8, and 25 agents) and as the horizon H grows on 3m. Latency varies only modestly with the horizon, since one diffusion pass covers all timesteps, but increases sharply with the number of agents, which require sequential inpainting passes. This reveals a computation–performance trade-off: MASTARS incurs a higher generation cost as the agent count grows, but it also delivers the strongest downstream performance in the largest-agent and longest-horizon settings (Table 20).

Table 17: Per-candidate sampling latency of MASTARS in SMAC on a single RTX 4090

Map	3m	5m_vs_6m	8m	25m
#agents	3	5	8	25
Time (s)	0.12	0.30	0.75	8.3

(a) Number of agents

Map	3m	3m	3m	3m
Horizon H	8	16	24	32
Time (s)	0.29	0.25	0.29	0.31

(b) Horizon length

I.4 Cost Reduction from Partial Generation

Partial generation preserves each trajectory up to its value-based subgoal and synthesizes only the suffix; on average only 44% of a trajectory is synthesized (Appendix J.2). To verify that this reduced search space translates into lower cost, we measure the end-to-end cost of obtaining 1,000 trajectories accepted by Eq. (6) with and without subgoals, using the same source data, threshold, hardware, and seeds, averaged over the three MPE environments. Both variants take 0.12s per candidate, so the difference comes entirely from the acceptance rate. As shown in Table 18, partial generation reduces the number of failed attempts and thereby the wall-clock time by 40%.

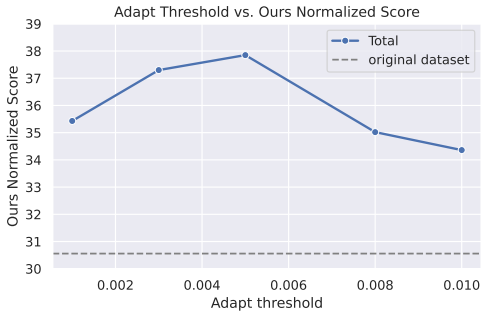
Table 18: Cost of collecting 1,000 accepted trajectories with and without subgoal-based partial generation on a single RTX 4090

Variant	Synthesized fraction	Attempts for 1,000 accepted	Rejection rate	Wall-clock time
w/o Subgoal	1.00	2,732	63%	328s
MASTARS	0.44	1,623	38%	**195s**

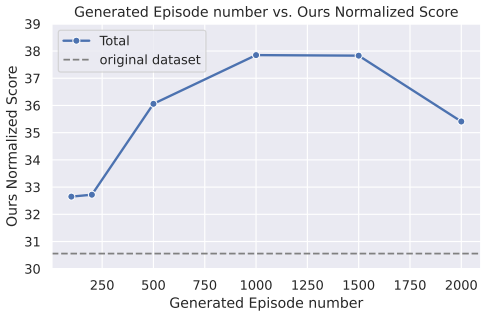
J Additional Experimental Analyses

J.1 Effect of Hyperparameters

We analyze the sensitivity of MASTARS to two key hyperparameters: the acceptance threshold ϵ and the number of generated trajectories, both of which influence the diversity of augmented data. For the analysis of the number of generated trajectories, we fix ϵ to 0.005. As shown in Fig. 7, MASTARS is robust to variations in ϵ, while the number of generated trajectories has a more significant impact. Generating too few samples yields limited improvement, whereas excessive augmentation can introduce distributional shift and degrade performance. Optimal results are achieved with a moderate amount of augmentation, highlighting the importance of balancing quantity and quality. Notably, MASTARS consistently outperforms the original dataset across all settings.



(a) Performance across acceptance threshold ϵ.



(b) Performance with different amounts of generated trajectories.

Figure 7: Analysis of generation sensitivity. (a) shows that our method is robust to threshold selection. (b) shows that moderate augmentation leads to better performance.

J.2 Synthesis Fraction and Efficiency

To verify that MASTARS avoids full-sequence regeneration, we measured the proportion of trajectories synthesized via value-based subgoals. By bypassing the excessively large search space of full regeneration and focusing exclusively on necessary refinements, MASTARS synthesizes only 44% of each trajectory on average, as reported in Table 19. This demonstrates an effective balance between leveraging high-value existing segments and generating new segments to ensure team-wide coordination.

Table 19: Post-subgoal ratio

	Medium	MD-Replay	Random
Tag	0.42	0.52	0.63
Spread	0.58	0.26	0.62
World	0.44	0.17	0.35

J.3 Scalability Analysis w.r.t. Agent Number and Horizon Length

To verify scalability in terms of agent number and horizon length, we conducted additional experiments in SMAC: the 25m scenario (25 agents) for agent scalability, and the 2c_vs_64zg scenario (horizon length 400) for temporal scalability. For both settings, we report results averaged over CFCQL and OMAR. As shown in Table 20, MADiTS showed performance drops in these challenging scenarios, while MASTARS consistently outperformed the original offline datasets. These results highlight the effective scalability of MASTARS, demonstrating strong performance even in scenarios with many agents and extended horizons.

Table 20: Test win rate on 25m and 2c_vs_64zg in the SMAC environment

Task \Algorithms	Original	MADiff*	MADiTS	MASTARS (Ours)
25m	0.7	0.75	0.6	**0.87**
2c_vs_64zg	0.52	0.7	0.05	**0.87**

J.4 Sensitivity Analysis of Value Estimation

We tested the stability of our SARSA-style value learning. Across MPE benchmarks, normalized TD error steadily converged to near zero (within $\sim$1%), as value learning progressed, confirming reliable estimation (Table 21). To test sensitivity, we intentionally injected random noise into values during subgoal generation, perturbing each value within $[(1-\sigma)\text{value}, (1+\sigma)\text{value}]$. As Fig. 8 shows, performance dropped with noise, underscoring the need for order-preserving estimates. Still, for all tested noise levels up to $\sigma = 0.2$, the score remained within 34.50–35.40 and MASTARS consistently outperformed the original dataset (30.56).

Table 21: TD error for value estimation

Time	0	30K	60K	90K
Value Loss	86.3	0.02	0.01	0.01

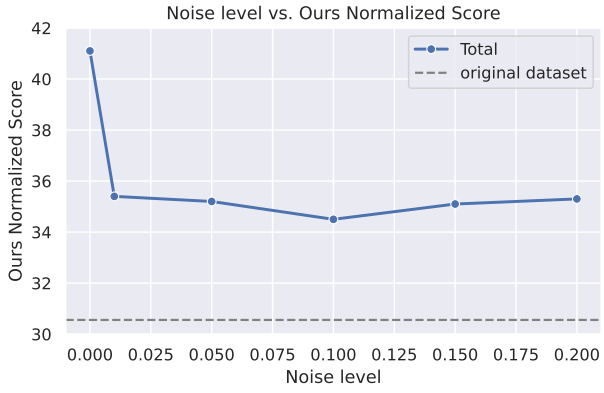


Figure 8: Average performance across different value noise levels

J.5 Robustness to Reward and Inverse-Dynamics Errors

The reward model g_ξ and inverse dynamics model I_ϕ only annotate generated observation sequences. Following the value-noise protocol in Appendix J.4, we perturb every predicted reward or action within $[(1-\sigma)x, (1+\sigma)x]$ before adding it to the augmented dataset. Scores are averaged over all MPE environments, dataset levels, and offline MARL algorithms; $\sigma = 0$ denotes unmodified MASTARS and Original (30.56) denotes training on the unaugmented dataset. As shown in Fig. 9, reward noise gradually lowers the score to 37.43, while inverse-dynamics noise causes an initial drop but stays at 38.06–38.40. All noisy settings outperform Original, showing graceful degradation.

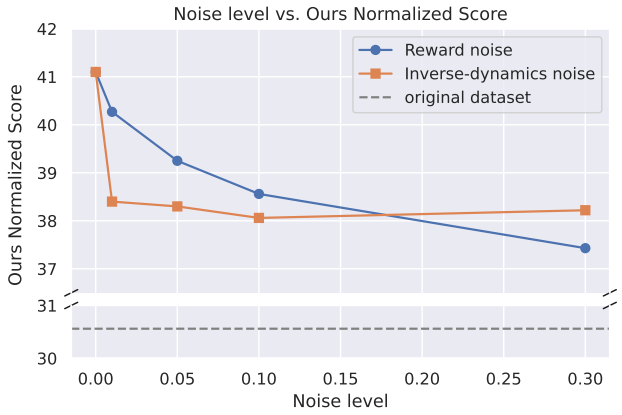


Figure 9: Average performance across different reward and inverse-dynamics noise levels

K Visualization of Generated Trajectories

We provide additional visualization examples across various tasks (Spread and Tag). Each row corresponds to one of eight rollout samples. The leftmost column shows the original trajectory from the dataset, while the three middle columns illustrate intermediate steps of agent-wise sequential generation. The rightmost column (highlighted in red) presents the final trajectory obtained after all agents have been sequentially generated using subgoal-conditioned inpainting. These examples demonstrate that MASTARS produces cooperative, diverse, and globally coherent trajectories, contributing to improved performance through effective augmentation.

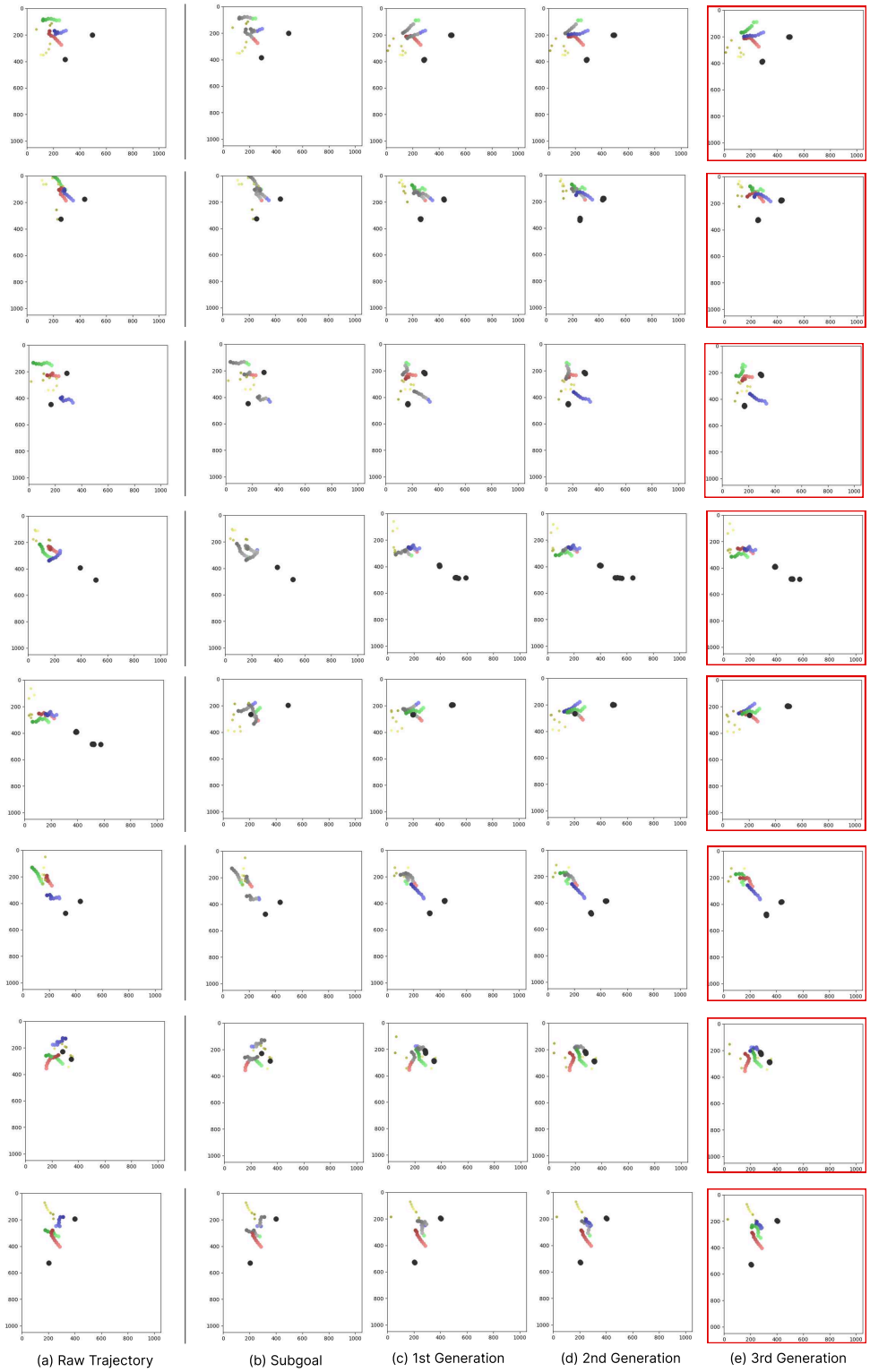


Figure 10: Visualization of the full trajectory generation process using our MASTARS framework in the `Tag-Medium` task. `X` marks are landmark positions reconstructed from each agent's observation.

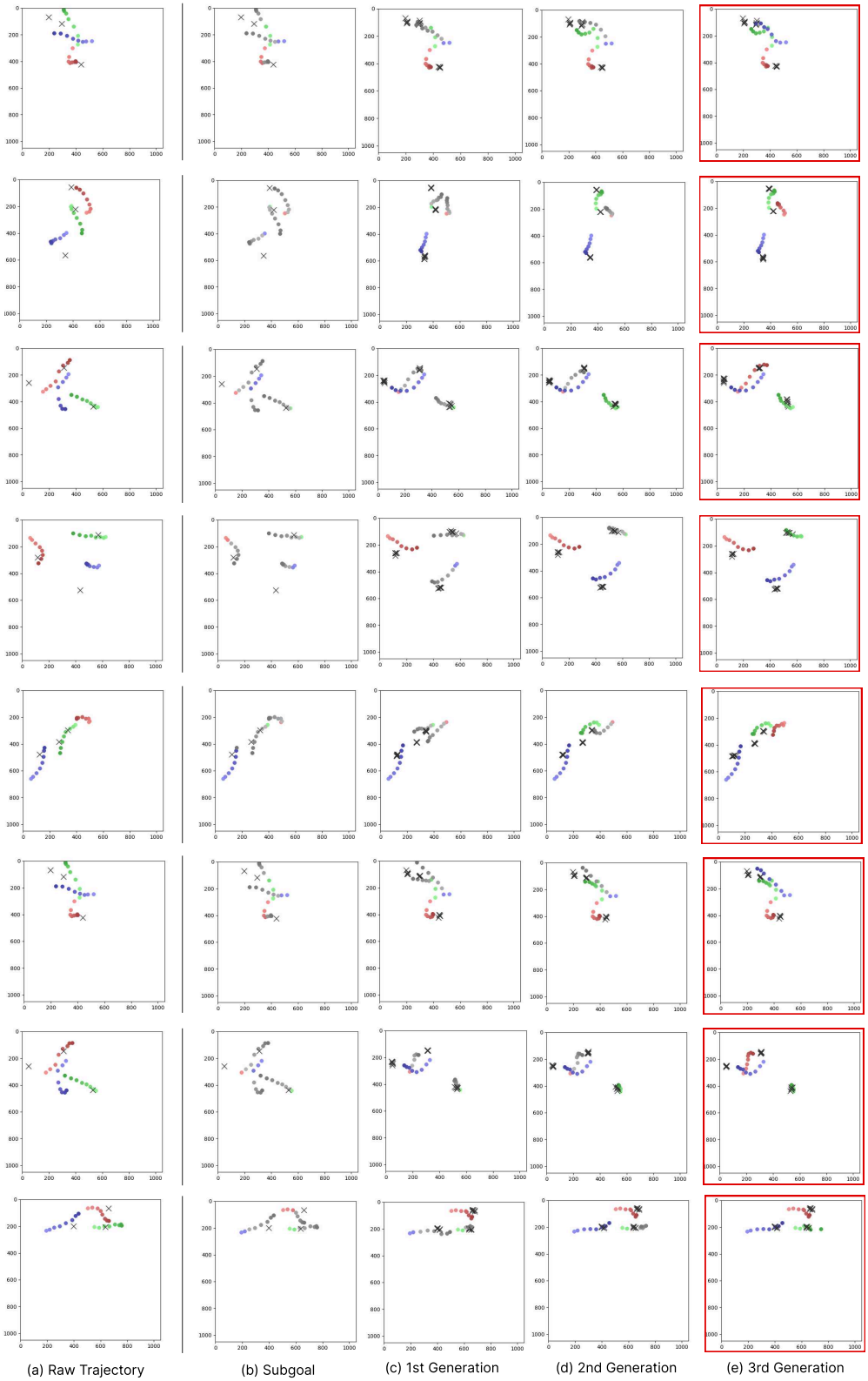


Figure 11: Visualization of the full trajectory generation process using our MASTARS framework in the `Spread-Medium` task. `X` marks are landmark positions reconstructed from each agent's observation.

L Visual Comparison between MADiTS and Our Method

To further analyze the differences between our method and other joint generation approaches, we provide a visual comparison with MADiTS. Although the main paper includes comparisons against MADiff*, it is important to clarify that MADiff is not a trajectory generation method. To the best of our knowledge, among diffusion-based approaches applicable to multi-agent offline reinforcement learning, only two methods are designed to synthesize new data: MADiTS and INS. We therefore compare to MADiTS, which is the only prior work that performs genuine joint trajectory generation. We do not compare against INS, since INS produces transitions rather than trajectories.

As shown in Fig. 12, MADiTS frequently fails to generate valid destination points (marked with X in the figure) and in some cases even fails to produce a feasible trajectory. This discrepancy stems from the fundamental differences in generation mechanisms. MADiTS relies on joint conditional diffusion to produce an entire trajectory in a single pass; as a result, its one-shot refinement process often lacks consistency, making the generated destinations noisy and poorly aligned with the surrounding agent behaviors.

In contrast, MASTARS adopts a RePaint-style inpainting strategy. By iteratively resampling masked segments conditioned on neighboring agents and previous denoising steps, our approach progressively corrects local inconsistencies and produces coherent, well-structured trajectories. This iterative refinement allows MASTARS to generate reliable and harmonized trajectories that MADiTS frequently fails to achieve.

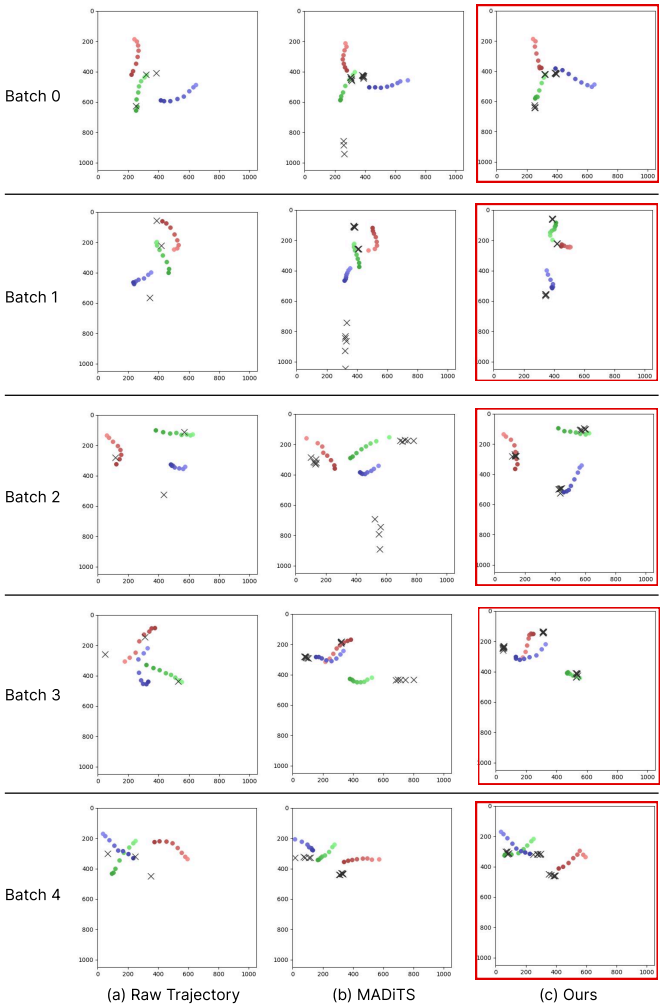


Figure 12: (a) Raw trajectory, (b) MADiTS, and (c) Ours

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 Section 1 clearly state the paper's main contributions, which are supported by the experimental results presented in Section 6.

   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: We discuss the limitations and future directions of MASTARS in Appendix B, including the scalability of sequential generation with respect to the number of agents and potential extensions to subgoal generation.

   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: [N/A]

Justification: The paper does not include theoretical results; all contributions are empirical.

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 full algorithm of MASTARS is described in Section 5 and Appendix E.1 (pseudocode), with training losses in Appendix E.2. Detailed experimental settings, dataset compositions, and all hyperparameters (including post-processing thresholds) are provided in Appendices F and H.

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: The source code, along with instructions to reproduce the main experimental results, is publicly available at `https://github.com/Jiwonjeon9603/MASTARS`. The benchmark datasets used in this paper (MPE, SMAC, og-marl) are publicly available, as cited in Appendix F.

   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: All training and evaluation details, including dataset compositions, algorithm configurations, network architectures, training schedules, and task-specific hyperparameters such as post-processing thresholds, are provided in Appendices F and H.

   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: All main results are reported as mean $\pm$ standard deviation across five random seeds, as described in Appendix H.

   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: The hardware specifications (8 NVIDIA GeForce RTX 3090 GPUs, 96 logical CPU cores) and runtime comparisons between MASTARS and baselines are reported in Appendix I.

   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: We have reviewed the NeurIPS Code of Ethics, and our research conforms to it in every respect.

   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: We discuss the broader impacts of MASTARS in Appendix A, including its potential to improve sample efficiency in safety-critical applications such as autonomous driving and robotics, as well as the need for careful validation when deploying in such environments.

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: This work does not release any pre-trained models, datasets, or other artifacts with a high risk of misuse. The augmented data is generated for standard simulated benchmarks (MPE, SMAC, SMAC-v2) and poses no such risks.

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: All existing assets used in this paper, including the MPE, SMAC, SMAC-v2 benchmarks and the og-marl, PyMARL, and QMIX implementations, are properly cited and used in accordance with their respective licenses.

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: [Yes]

Justification: The MASTARS code is publicly released with documentation at `https://github.com/Jiwonjeon9603/MASTARS`. The data generation procedure for our custom SMAC-v2 datasets, including unit-type sampling distributions and dataset statistics, is fully described in Appendix G.

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: This work does not involve crowdsourcing or research with human subjects.

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: This work does not involve research with human subjects, and IRB approval is therefore 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: LLMs are not used as any component of the core methodology of this research, and were only used as assistive tools for writing and grammar checking, which does not require declaration per the NeurIPS LLM policy.

    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.