INS: INTERACTION-AWARE SYNTHESIS TO ENHANCE OFFLINE MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Data scarcity in offline multi-agent reinforcement learning (MARL) is a key challenge for real-world applications. Recent advances in offline single-agent reinforcement learning (RL) demonstrate the potential of data synthesis to mitigate this issue. However, in multi-agent systems, interactions between agents introduce additional challenges. These interactions complicate the synthesis of multi-agent datasets, leading to data distortion when inter-agent interactions are neglected. Furthermore, the quality of the synthetic dataset is often constrained by the original dataset. To address these challenges, we propose **INteraction-aware Synthesis (INS)**, which synthesizes high-quality multi-agent datasets using diffusion models. Recognizing the sparsity of inter-agent interactions, INS employs a sparse attention mechanism to capture these interactions, ensuring that the synthetic dataset reflects the underlying agent dynamics. To overcome the limitation of diffusion models requiring continuous variables, INS implements a bit action module, enabling compatibility with both discrete and continuous action spaces. Additionally, we incorporate a select mechanism to prioritize transitions with higher estimated values, further enhancing the dataset quality. Experimental results across multiple datasets in MPE and SMAC environments demonstrate that INS consistently outperforms existing methods, resulting in improved downstream policy performance and superior dataset metrics. Notably, INS can synthesize high-quality data using only 10% of the original dataset, highlighting its efficiency in data-limited scenarios.

1 INTRODUCTION

Recent advances in offline multi-agent reinforcement learning make it possible to train policies for real-world applications, such as autonomous driving (Zhang et al., 2024), robotics (Hu et al., 2023), and network control (Ma et al., 2024), entirely from previously collected multi-agent datasets. This approach eliminates the need for online interactions with environments, mitigating the high costs associated with trial-and-error in online MARL. However, offline MARL faces challenges due to the fixed dataset with incomplete coverage of the state-action space (Fujimoto et al., 2019), leading to the distribution shift problem and suboptimal policy performance. Furthermore, data collection in real-world multi-agent scenarios is expensive and potentially dangerous (Canese et al., 2021), restricting the scalability of datasets and resulting in policy overfitting (Agarwal et al., 2020). Consequently, learning effective MARL policies from limited datasets remains a critical challenge.

The data scarcity challenge also exists in offline single-agent reinforcement learning, where using generative models for data synthesis is a prevalent approach to overcome this issue. Despite the promising results in single-agent settings, extending this approach to multi-agent scenarios presents unique challenges. Firstly, multi-agent systems involve complex inter-agent interactions (Georgeff, 1988), such as collisions and information exchange, which influence agent dynamics. Therefore, merely applying existing data synthesis methods by treating each agent independently often results in lower-quality synthetic datasets, as these methods fail to capture the true agent dynamics. In Appendix A, we present a motivating example to illustrate this phenomenon. Moreover, an agent typically interacts with only a subset of other agents (DeWeese & Qu, 2024; Liu et al., 2024), and considering interactions with irrelevant agents can introduce distractions into the synthesis process. Secondly, research shows that both the diversity and quality of datasets are crucial for effective policy training (Corrado & Hanna, 2024). While generative models aim to preserve diversity by approximating the original dataset distribution, their effectiveness is constrained by the dataset quality.

Low-quality original datasets often lack sufficient high-value transitions, leading to imbalanced synthetic datasets and ultimately hindering effective policy learning. Finally, since most diffusion models are limited to generating continuous data, existing synthesis methods primarily focus on continuous action spaces, such as D4RL (Fu et al., 2020) and MPE (Lowe et al., 2017). This restricts their applicability to multi-agent environments with discrete action spaces, such as SMAC (Samvelyan et al., 2019). In summary, there is a critical need for a method that can *synthesize high-quality multi-agent datasets applicable to both continuous and discrete action spaces in the offline MARL domain*.

To meet this need, we propose **INteraction-aware Synthesis (INS)**, which leverages a powerful generative model, diffusion model (Ho et al., 2020), to synthesize multi-agent datasets. Diffusion models excel at ensuring broad coverage of the data distribution while maintaining high quality, making them ideal for modeling complex multi-agent transitions. To model the inter-agent interaction during the synthesis process, INS integrates a sparse attention mechanism (Martins & Astudillo, 2016). This mechanism assigns different weights to agents based on their interactions and filters out irrelevant ones. To further improve dataset quality, we design a value-based select mechanism that estimates transition values and prioritizes higher-value transitions for the synthetic dataset. To ensure compatibility with discrete action environments, INS implements a bit action module, bridging the gap between continuous diffusion models and transitions with discrete actions. We conduct a preliminary evaluation of INS with existing synthesis methods on various multi-agent datasets. As illustrated in Figure 1, our method outperforms baseline methods in both policy performance and dataset metrics. In contrast, baseline methods that ignore inter-agent interactions yield performance comparable to, or even worse than, that of the original datasets. More details of the preliminary evaluation are provided in Appendix B.

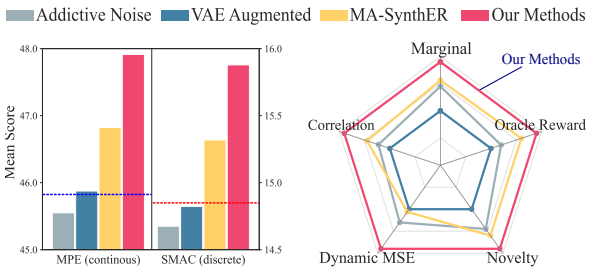


Figure 1: **Preliminary evaluation.** Our INS consistently outperforms existing data synthesis methods across policy performance and dataset metrics.

Our contributions are three-fold: (1) We propose an interaction-aware synthesis method, INS, which leverages diffusion models to synthesize high-quality multi-agent datasets, applicable to both continuous and discrete action spaces. INS is compatible with any offline MARL methods, as it synthesizes data without modifying the policy learning algorithms. To the best of our knowledge, INS is the first data synthesis approach for offline MARL. (2) We introduce a sparse attention mechanism to capture inter-agent interactions in the synthesis process, effectively reducing the irrelevant interaction modeling. Additionally, we design a select mechanism to improve dataset quality by increasing the proportion of high-value transitions. (3) We conduct extensive evaluations on MPE and SMAC datasets, showing that our method enhances both dataset metrics and policy performance across various offline MARL methods. Moreover, INS can synthesize high-quality data with only 10% of the original dataset, underscoring its effectiveness in data-limited scenarios.

2 Related Works

Data Synthesis in Reinforcement Learning. Data synthesis is an effective technique that is widely applied across diverse domains, including computer vision (He et al., 2023), natural language processing (Puri et al., 2020), and large language models (Li et al., 2023b). This approach effectively improves performance in data-limited scenarios. Offline reinforcement learning frequently encounters challenges stemming from dataset scarcity (Wang et al., 2022). Although data synthesis shows potential to improve sample efficiency and the approximation of the Q function in reinforcement learning, its application in the offline domain remains underexplored. Previous works (Cho et al., 2022; Ma et al., 2022) attempt to use generative models, such as VAEs or GANs, in offline reinforcement learning for data synthesis yield suboptimal results, primarily because of the intrinsic limitations of these models. Recent advances leverage diffusion models for data synthesis, e.g., MTDiff-S (He et al., 2024) and SynthER (Lu et al., 2024). MTDiff-S generates trajectory-level data for multi-task settings, producing trajectories for unseen tasks. SynthER, closely related to our method, synthesizes data at the transition level for subsequent policy training. In offline MARL, data synthesis has not

yet achieved the same level of performance as that observed in single-agent RL. This disparity is pronounced due to the complex dynamics in multi-agent systems (Tian et al., 2023; Chai et al., 2024), where multiple agents interact within a shared environment. Our method diverges from existing approaches by explicitly considering inter-agent interactions and improving dataset quality.

Diffusion Models in Offline MARL. Diffusion models (Ho et al., 2020; Karras et al., 2022), as a powerful class of generative models, have recently been adopted in offline reinforcement learning as planners and policies (Zhu et al., 2023b). The application of diffusion models in the field of MARL remains underexplored. Building on the foundational work of Janner et al. (2022) and Wang et al. (2023), recent studies extend diffusion models to the MARL domain, applying them to trajectory generation (Zhu et al., 2023a) and policy estimation (Li et al., 2023a). In contrast to these methods, which focus on the policy level, we aim to enhance offline MARL performance from a data perspective by using diffusion models to synthesize multi-agent datasets. Further details regarding the differences between our method and related works can be found in Appendix D.

3 Preliminaries

3.1 Offline Multi-agent Reinforcement Learning

We focus on the fully cooperative MARL, which can be modeled as a *Decentralized-Partially Observable Markov Decision Process* (Dec-POMDP) (Oliehoek et al., 2016), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \Omega, R, n, \gamma \rangle$. Here, $s \in \mathcal{S}$ denotes the global state of the environment and $\mathcal{A}$ represents the action set for each of the n agents. At each time step, each agent $i \in \mathcal{N} \coloneqq \{1, 2, \ldots, n\}$ receives a local observation $o_i \in \mathcal{O}$, generated by the observation function $\Omega(s, i) : \mathcal{S} \times \mathcal{N} \to \mathcal{O}$. Each agent i selects and executes its own action $a_i \in \mathcal{A}$, resulting in a joint action $\boldsymbol{a} \in \mathcal{A}^n$ that induces the next state according to the state transition function $\mathcal{P}(s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A}^n \to \mathcal{S}$. The environment provides a global reward r shared by agents, determined by the reward function $R(s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A}^n \to \mathbb{R}$. The objective is to find the optimal policies $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ that maximize the discounted return $G = \sum_{t=0}^{T} \gamma^t r_t$, where T denotes the horizon and $\gamma \in [0, 1)$ represents the discount factor.

In the offline MARL setting, access to each agent's transition is restricted to a fixed multi-agent dataset $\mathcal{D}_{\text{ori.}} \coloneqq \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{|\mathcal{D}_{\text{ori.}}|}\}$. Each joint transition is defined as $\boldsymbol{x} \coloneqq (\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}', r)$, and the transition of agent i is given by $x_i \coloneqq (o_i, a_i, o_i', r) \in \mathbb{R}^{d_x}$. The dataset is typically collected by arbitrary policies like human experts or random policies. However, these policies fail to guarantee sufficient coverage of the whole state-action space, which grows exponentially with the number of agents. As a result, learning effective decision-making policies that surpass the behavior policy from scarce and potentially suboptimal multi-agent datasets poses a significant challenge. Existing offline MARL methods (Yang et al., 2021; Pan et al., 2022) attempt to address these issues by incorporating policy constraints and conservative estimation techniques. Moving beyond these methods, another intuitive yet underexplored solution to dataset scarcity is data synthesis (Levine et al., 2020).

3.2 Diffusion Models

Diffusion models (DM) (Ho et al., 2020) represent a class of generative models that generate data $\boldsymbol{x}_0$ by incrementally removing noise from a Gaussian distribution. These models comprise two primary processes: a forward noising process and a reverse denoising process. Given the data distribution $p(\boldsymbol{x})$ and the noise standard deviation σ, the data distribution with added noise is denoted by $p(\boldsymbol{x}; \sigma)$. The forward process is characterized by a Markov chain, starting from the data $\boldsymbol{x}_0 \sim p(\boldsymbol{x})$ and gradually increasing the noise level to $\sigma_{\max}$. The reverse process involves K times sequential denoising, starting with noise $\boldsymbol{x}_K$ sampled from a Gaussian distribution $\mathcal{N}(0, \sigma_{\max}^2 I)$, and following a decreasing fixed sequence of noise levels $\sigma_{\max} = \sigma_0 > \sigma_1 > \cdots > \sigma_K = 0$. As a result, the endpoint $\boldsymbol{x}_0$ of this process aligns with the original data distribution $p(\boldsymbol{x})$.

In this work, we adopt the Elucidated Diffusion Model (EDM) architecture (Karras et al., 2022), which implements both forward and reverse processes through the continuous schedule of noise levels in a probability flow ordinary differential equation (ODE):

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(k)\sigma(k)\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}; \sigma(k))\mathrm{d}k, \tag{1}$$

where the dot notation represents the time derivative, and $\sigma(k)$ denotes the noise level at time k. $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}; \sigma(k))$ is the score function, which can be computed via the connection between score

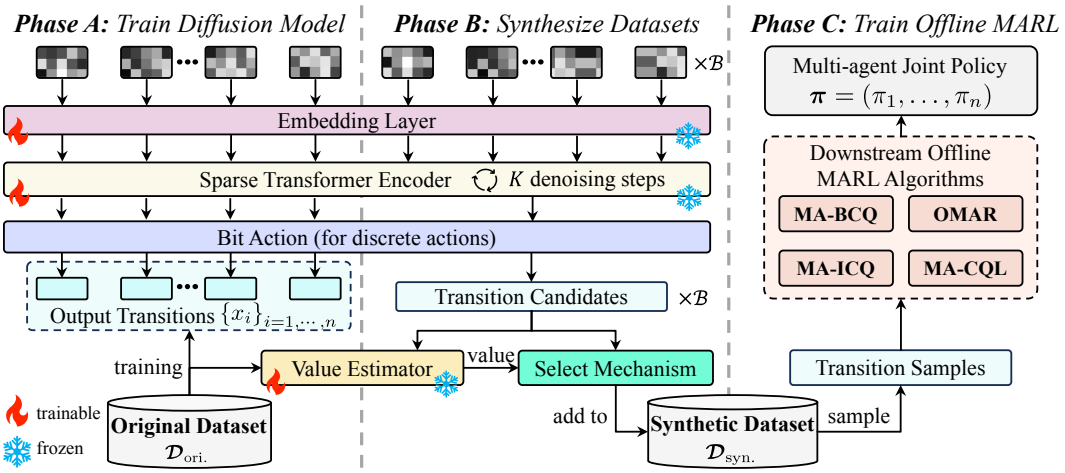


Figure 2: **Overview of INS framework.** The framework consists of three phases: In the first phase, we train a diffusion model. Following this, we synthesize and select transition data. Lastly, we employ the synthetic transitions to train offline MARL policy.

matching and denoising: $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}; \sigma) = (D_\theta(\boldsymbol{x}; \sigma) - \boldsymbol{x})/\sigma^2$. $D_\theta(\boldsymbol{x}; \sigma)$ denotes the denoiser with parameters θ. The L_2 loss objective for denoising, applied to the denoiser $D_\theta(\boldsymbol{x}; \sigma)$ is defined as:

$$\mathcal{L}_{\text{dm}}(\theta) = \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}), \epsilon \sim \mathcal{N}(0, \sigma^2 I)} \left\| D_\theta(\boldsymbol{x} + \epsilon, \sigma) - \boldsymbol{x} \right\|^2 . \tag{2}$$

Subsequently, we can sample transitions via solving Eq. (1) with the trained denoising network $D_\theta(\boldsymbol{x}; \sigma)$ and score function $\nabla_{\boldsymbol{x}} \log p\left(\boldsymbol{x}; \sigma(k)\right)$.

4 Method

In this section, we present interaction-aware synthesis (INS), which synthesizes high-quality multi-agent transitions and is compatible with any offline MARL methods. As illustrated in Figure 2, our method consists of three phases: **(i)** train the diffusion model, **(ii)** synthesize and select transition data, and **(iii)** train offline MARL policy.

4.1 Phase A: Train Diffusion Model

Diffusion Model Architecture. In order to synthesize multi-agent transitions, we employ a diffusion model to learn the distribution of the original dataset $\mathcal{D}_{\text{ori.}}$. While existing methods typically use a U-Net (Ronneberger et al., 2015) denoising network, INS adopts a Transformer encoder (Vaswani et al., 2017) as its basic architecture, incorporating a sparse attention mechanism along the agent dimension. This design offers two key advantages: First, in multi-agent systems, interaction relationships are often sparse (Liu et al., 2024; Li et al., 2019), with agents typically interacting only with specific nearby agents. The conventional dense attention mechanism in the Transformer encoder assigns non-zero weights to all agent pairs, implicitly assuming universal interaction. This mechanism is inefficient and complicates learning the true interaction distribution. Moreover, modeling irrelevant-agent interaction can distract diffusion models, potentially hindering the denoising process. In contrast, the sparse attention focuses on critical inter-agent interactions, thereby simplifying the learning task for models. Second, prior research (Dong et al., 2024) indicates that Transformer-based diffusion models exhibit improved training stability and achieve superior performance in reinforcement learning contexts. The diffusion model architecture of INS is detailed in Appendix K.1.

Sparse Attention for Inter-agent Interaction. In the denoising process, the transition sample $\boldsymbol{x}_k = [x_{1,k}, \dots, x_{n,k}] \in \mathbb{R}^{n \times d_x}$ at denoising step k is initially embedded into $\boldsymbol{x}_{k,e} \in \mathbb{R}^{n \times d_e}$ via an embedding layer. For notational simplicity, we denote the embedding matrix as $\boldsymbol{X}$ instead of $\boldsymbol{x}_{k,e}$. The embedding features of each agent are then projected into query $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q$, key $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K$, and value $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V$ representations, where $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$ are learnable parameters. The

product of query and keys is denoted as $\boldsymbol{Z} = \boldsymbol{Q}\boldsymbol{K}^\intercal \in \mathbb{R}^{n\times n}$, comprising n rows $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$. To focus on the critical interactions, we replace the dense Softmax in conventional dense attention with Sparsemax (Martins & Astudillo, 2016), which tends to produce sparse attention weights:

$$\text{Sparsemax}(\boldsymbol{z}) = \underset{\boldsymbol{p}\in\Delta^{n-1}}{\arg\min} \|\boldsymbol{p} - \boldsymbol{z}\|_2 , \tag{3}$$

where $\boldsymbol{z} \in \mathbb{R}^n$ is the input, $\boldsymbol{p} \in \mathbb{R}^n$ is the output, and $\Delta^{n-1} := \{\boldsymbol{p} \in \mathbb{R}^n | \boldsymbol{p} \geq 0, \|\boldsymbol{p}\|_1 = 1\}$ is the $(n-1)$-dimensional simplex. The solution to Eq. (3) is given as follows:

$$\boldsymbol{p} = [\boldsymbol{z} - \sigma(\boldsymbol{z})\mathbf{1}]_+, \tag{4}$$

where $\sigma(\boldsymbol{z})$ is the Sparsemax threshold, $\mathbf{1} \in \mathbb{R}^n$ is the all-one vector, and $[\cdot]_+$ denotes the ReLU function. In order to get the solution $\boldsymbol{p}$, we calculate the threshold $\sigma(\boldsymbol{z})$ as follows:

$$\sigma(\boldsymbol{z}) = \frac{\left(\sum_{i\leq m(\hat{\boldsymbol{z}})} \hat{\boldsymbol{z}}_i - 1\right)}{m(\hat{\boldsymbol{z}})}, \tag{5}$$

where $\hat{\boldsymbol{z}}$ is $\boldsymbol{z}$ sorted in descending order, and $m(\hat{\boldsymbol{z}})$ is the number of non-zero elements in $\boldsymbol{p}$:

$$m(\hat{\boldsymbol{z}}) = \underset{m\in\{1,\ldots,n\}}{\arg\max} \left\{ m\hat{\boldsymbol{z}}_m > \sum_{i\leq m} \hat{\boldsymbol{z}}_i - 1 \right\}. \tag{6}$$

Thus, the resulting sparse attention weight is given by $\boldsymbol{P} = \text{Sparsemax}(\boldsymbol{Z}) \in \mathbb{R}^{n\times n}$. The output of sparse attention is $\left(\boldsymbol{P}/\sqrt{d_e}\right)\boldsymbol{V}$, which preserves key properties of Softmax while assigning zero probability to irrelevant agents. This approach allows the diffusion model to concentrate on critical inter-agent interactions, effectively mitigating distractions during the denoising process. A detailed discussion of Sparsemax and its comparison with related work can be found in Appendix E.

Bit Action. While the current diffusion model inherently generates continuous data, many multi-agent environments, such as SMAC (Samvelyan et al., 2019), feature discrete action spaces. To bridge this gap, we introduce a bit action module inspired by Bit Diffusion in Chen et al. (2023). The bit action module maps a discrete action space containing M actions to a real-valued bit vector of length $L = \lceil \log_2(M) \rceil$, as $\{0.0, 1.0\}^L$. During the denoising process, this bit vector is concatenated with other continuous variables in the transition. After that, the output vector corresponding to the action is thresholded into a binary bit vector via a quantization mechanism. Then this vector is mapped back to the original discrete action space. The bit action design offers two key advantages. Firstly, it enables INS to be universally applicable to multi-agent environments with either discrete or continuous action spaces without requiring additional modifications. Secondly, compared to alternative discretization techniques such as one-hot encoding, the bit action module effectively reduces the action space dimensionality from M to $\lceil \log_2(M) \rceil$, thereby decreasing computational complexity.

Diffusion Model Training. Given the original multi-agent dataset $\mathcal{D}_{\text{ori.}}$ containing transitions $\boldsymbol{x}$ and the denoising network $D_\theta(\boldsymbol{x}; \sigma)$, we train a diffusion model $p_{\text{dm}}(\boldsymbol{x})$ to approximate the joint distribution $p(\boldsymbol{x})$ of multi-agent transitions. The parameters θ of the denoising network are optimized by minimizing the L_2 loss for denoising score matching, as described in Eq. (2). With the trained model, we can generate numerous transitions from the original dataset, as shown in the next phase.

4.2 Phase B: Synthesize Datasets

The synthesis process begins by sampling an initial noisy transition $\boldsymbol{x}_K$, which is considered the result of multiple noise additions to the original transition. Following the diffusion steps described in Eq. (1), the final joint transition is obtained as:

$$\underbrace{(\hat{\boldsymbol{o}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{o}}', \hat{r})}_{\boldsymbol{x}_K} \xrightarrow{K \text{ steps}} \underbrace{(\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}', r)}_{\boldsymbol{x}_0}. \tag{7}$$

In offline MARL, both dataset diversity and high-value transitions are crucial, implying that the relative importance of each transition varies (Schweighofer et al., 2022). Previous approaches primarily focused on diversity, overlooking the role of high-value transitions, which led to an

imbalance in transition quality within the synthetic dataset $\mathcal{D}_{\text{syn.}}$. To mitigate this imbalance, we propose a transition **select mechanism** that prioritizes transitions based on their estimated values. Specifically, a state **value estimator** is trained simultaneously with the diffusion model during *Phase A* to estimate transition values:

$$\mathcal{L}_\phi^V(\boldsymbol{o}) = \mathbb{E}_{(\boldsymbol{o},r,\boldsymbol{o}')\sim\mathcal{D}_{\text{ori.}}}[r + \gamma V_{\bar{\phi}}(\boldsymbol{o}') - V_\phi(\boldsymbol{o})]^2, \tag{8}$$

where ϕ and $\bar{\phi}$ are parameters of the value estimator, and γ is the discount factor. During the synthesis process, $\mathcal{B}$ transitions are generated as candidates. The value estimator evaluates each transition, and candidates within the proportion η are selected for inclusion in the final output synthetic dataset. In this paper, we propose two select mechanisms:

- **Top-K**: All candidate transitions are sorted by their estimated values, and the top η proportion of them are selected. While this approach intuitively selects the most valuable transitions for training, it may reduce dataset diversity and degrade policy performance.
- **Softmax**: A Softmax probability is calculated based on the estimated values of transitions:

$$p(\boldsymbol{x}_0^i) = \frac{\exp(V_\phi(\boldsymbol{o}_0^i))}{\sum_{j=1}^{\mathcal{B}} \exp(V_\phi(\boldsymbol{o}_0^j))}. \tag{9}$$

  Then the η proportion of candidates are sampled according to these probabilities. Despite the additional computational cost, empirical results show that it enhances downstream policy performance. Therefore, the Softmax select mechanism is employed in the primary experiments.

The selection proportion η represents a trade-off between dataset diversity and quality. Increasing this proportion tends to concentrate more high-value transitions but at the cost of reduced diversity. This parameter can be fine-tuned according to the specific requirements of the synthetic data.

4.3 Phase C: Train Offline MARL

In the final phase, the high-quality dataset synthesized in the previous stages is used for offline multi-agent policy training. As previously mentioned, our approach is orthogonal to existing offline MARL methods. Therefore, rather than designing new policy training algorithms, we directly integrate INS with these methods to assess its effectiveness. The overall algorithm is given in Appendix C.

5 Experiments

In this section, we evaluate INS across different multi-agent settings, including both discrete and continuous action spaces. We initially evaluate our method on multiple datasets, replacing the original datasets with synthetic datasets to isolate and highlight the impact of synthesis methods. We also investigate how the quality and size of synthetic data affect downstream policy performance in offline MARL. Additionally, we conduct an extensive ablation study to empirically validate the effectiveness of key components and hyperparameters within our method. We provide visualizations of data coverage, attention weights, reward distributions, and value distributions to illustrate the effects of our method. Finally, we demonstrate the advantage of INS when trained on small original datasets.

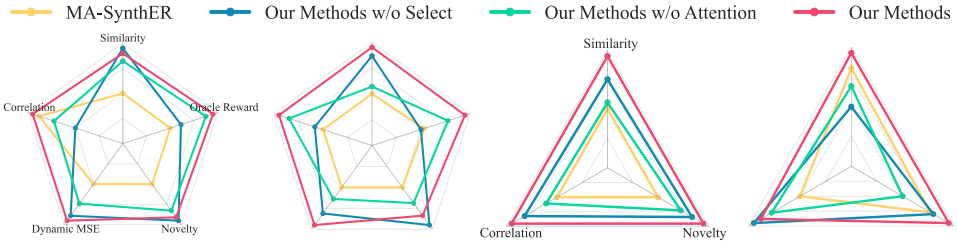
5.1 Experimental Setup

We conduct experiments on two widely used offline MARL environments: Multi-Agent Particle Environment (MPE) (Lowe et al., 2017) for continuous action space tasks and StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) for discrete action space tasks. We use a diverse range of original datasets, including 12 from MPE and 9 from SMAC. All datasets are categorized based on the quality of the policies used for collection. In MPE, datasets are categorized as expert, medium, medium replay (md-replay), or random, while in SMAC, they are classified as good, medium, or random. In our primary experiments, the size of the synthetic dataset is fixed at $5M$. More information about datasets is provided in Appendix H.

We compare our method against SynthER (Lu et al., 2024), a recently proposed synthetic method for offline single-agent RL. To ensure a fair comparison, we implement a multi-agent version of SynthER

Table 1: **The normalized average scores of INS and baselines in MPE and SMAC.** Original means training policies on the original datasets. We show the mean and standard deviation of scores averaged over 8 seeds. The best performance is highlighted in **bold**.

Datasets			MA-ICQ			MA-CQL			OMAR/MA-BCQ		
			Original	MA-SynthER	INS (ours)	Original	MA-SynthER	INS (ours)	Original	MA-SynthER	INS (ours)
MPE (continuous)	Spread	Expert	102.8 ± 3.3	104.7 ± 5.2	**107.0 ± 4.4**	97.9 ± 5.3	96.8 ± 4.6	**98.5 ± 5.0**	114.1 ± 3.3	111.3 ± 1.9	**115.6 ± 3.1**
		Medium	26.8 ± 4.8	27.9 ± 3.8	**30.2 ± 2.5**	34.1 ± 7.1	35.6 ± 8.2	**36.6 ± 6.5**	47.4 ± 17.3	45.6 ± 11.2	**48.0 ± 11.3**
		Md-replay	12.9 ± 6.1	14.4 ± 5.6	**15.2 ± 5.4**	20.3 ± 6.4	18.1 ± 7.1	**21.1 ± 8.8**	37.8 ± 10.5	38.1 ± 12.1	**38.8 ± 13.0**
		Random	5.1 ± 2.3	4.2 ± 0.6	**5.2 ± 3.5**	**23.9 ± 5.9**	23.0 ± 10.5	23.6 ± 7.3	34.2 ± 6.2	33.8 ± 7.5	**36.7 ± 4.6**
	Tag	Expert	111.7 ± 15.1	114.1 ± 15.6	**114.2 ± 17.2**	93.8 ± 12.2	92.5 ± 15.7	**93.9 ± 16.6**	117.2 ± 18.8	114.4 ± 20.3	**119.1 ± 19.6**
		Medium	62.3 ± 19.6	61.6 ± 18.6	**64.7 ± 20.3**	60.8 ± 22.7	60.2 ± 24.3	**61.4 ± 21.3**	66.8 ± 20.9	64.1 ± 23.4	**67.5 ± 27.3**
		Md-replay	33.9 ± 25.9	36.2 ± 28.0	**36.4 ± 30.3**	24.8 ± 17.5	25.2 ± 16.1	**26.1 ± 18.1**	46.1 ± 13.6	44.9 ± 16.3	**48.6 ± 14.9**
		Random	1.8 ± 2.3	1.6 ± 2.4	**1.9 ± 4.1**	4.7 ± 8.8	3.8 ± 9.4	**4.8 ± 8.0**	**11.0 ± 2.4**	9.3 ± 1.1	10.8 ± 2.7
	World	Expert	108.5 ± 20.7	110.7 ± 24.3	**111.4 ± 22.4**	72.4 ± 31.1	72.8 ± 28.7	**73.8 ± 28.0**	110.4 ± 25.7	108.9 ± 22.3	**112.7 ± 25.8**
		Medium	71.2 ± 19.2	**73.6 ± 19.4**	72.1 ± 17.9	57.9 ± 11.9	55.0 ± 10.6	**58.8 ± 10.1**	74.3 ± 14.5	72.1 ± 19.5	**75.0 ± 16.3**
		Md-replay	12.2 ± 9.8	**12.6 ± 6.4**	11.9 ± 7.0	29.3 ± 12.8	27.1 ± 16.2	**30.4 ± 14.2**	43.0 ± 20.5	42.6 ± 21.2	**46.2 ± 14.5**
		Random	0.8 ± 3.4	0.3 ± 2.9	**0.8 ± 1.3**	**0.7 ± 1.5**	0.2 ± 3.3	0.4 ± 2.7	5.7 ± 2.2	4.2 ± 3.3	**6.9 ± 3.1**
Total Score			550.0	561.9	**573.0**	520.6	510.3	**529.1**	708.0	689.3	**725.9**
SMAC (discrete)	3m	Good	18.1 ± 0.6	18.6 ± 1.5	**19.9 ± 1.4**	18.5 ± 0.5	18.2 ± 0.9	**20.1 ± 0.3**	3.4 ± 0.8	3.4 ± 0.8	**4.1 ± 0.5**
		Medium	16.9 ± 0.8	17.9 ± 1.1	**18.5 ± 1.8**	16.3 ± 0.8	16.2 ± 1.1	**17.9 ± 1.0**	4.1 ± 1.5	**4.5 ± 0.9**	4.4 ± 1.3
		Poor	13.0 ± 1.2	13.5 ± 2.5	**13.8 ± 0.9**	**5.6 ± 0.6**	4.0 ± 2.1	5.0 ± 1.6	3.3 ± 1.0	**3.6 ± 1.6**	3.1 ± 1.1
	5m_vs_6m	Good	16.3 ± 0.8	16.4 ± 1.2	**17.2 ± 1.7**	12.8 ± 2.1	13.6 ± 4.1	**14.3 ± 2.9**	2.1 ± 0.4	2.8 ± 0.7	**3.0 ± 0.4**
		Medium	12.8 ± 0.6	12.7 ± 1.0	**13.9 ± 1.1**	14.8 ± 1.7	14.1 ± 1.9	**16.1 ± 2.1**	3.9 ± 0.7	4.2 ± 0.8	**4.3 ± 0.7**
		Poor	9.4 ± 0.5	9.5 ± 0.7	**9.9 ± 0.4**	10.5 ± 0.7	10.8 ± 1.0	**12.3 ± 2.7**	3.2 ± 0.6	2.9 ± 0.2	**3.3 ± 0.2**
	8m	Good	19.3 ± 0.9	20.2 ± 2.0	**20.7 ± 1.4**	11.1 ± 5.1	10.1 ± 5.9	**12.5 ± 4.2**	4.5 ± 0.8	4.9 ± 1.0	**5.0 ± 1.2**
		Medium	18.0 ± 1.2	18.8 ± 0.5	**19.3 ± 0.5**	15.4 ± 3.3	15.1 ± 4.0	**16.4 ± 2.5**	5.6 ± 0.5	5.3 ± 0.3	**5.7 ± 1.1**
		Poor	9.8 ± 0.6	**10.2 ± 1.4**	10.1 ± 1.1	4.4 ± 2.0	3.8 ± 3.3	**5.4 ± 1.9**	**3.4 ± 0.8**	2.7 ± 1.2	2.9 ± 0.6
Total Score			133.6	137.8	**143.3**	109.4	105.9	**120.0**	33.5	34.3	**35.8**



(a) Spread_expert (b) Spread_medium (c) 3m_good (d) 3m_medium

Figure 3: **Dataset metrics of the synthetic dataset.** We compare datasets synthesized by INS and MA-SynthER across different datasets. Details of metrics are provided in Appendix F.

(MA-SynthER) and evaluate it in the same settings. The detailed implementation of MA-SynthER is provided in Appendix J. The data synthesis methods used in the preliminary evaluation (Additive Noise and VAE Augmented) are excluded from the main experiments due to their poor performance. In order to demonstrate the broad applicability of our method, we select four widely used offline MARL algorithms for downstream policy training: MA-ICQ (Yang et al., 2021), MA-CQL (Kumar et al., 2020), OMAR (Pan et al., 2022), and MA-BCQ (Wu et al., 2019). More information on these algorithms is provided in Appendix I.

5.2 Comparative Evaluation

We begin our evaluation by comparing the performance of INS combined with different offline MARL algorithms across multiple environments. Due to constraints in the official code implementations, we evaluate OMAR only in MPE and MA-BCQ only in SMAC. The main results are presented in Table 1, which demonstrate that offline MARL algorithms using INS synthetic datasets outperform all other baseline datasets across all algorithms on the total score. Among different types of original datasets (expert/good, medium, md-replay, and random), synthetic datasets generated from expert/good and md-replay datasets lead to more substantial improvements in downstream policy performance compared with those generated from medium and random datasets. This discrepancy can be attributed to two key factors. First, the narrow coverage of medium datasets constrains the potential improvements in data coverage achievable by the diffusion model of INS, thereby limiting

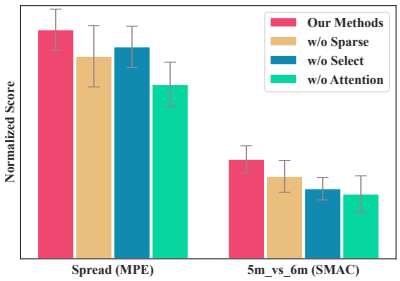
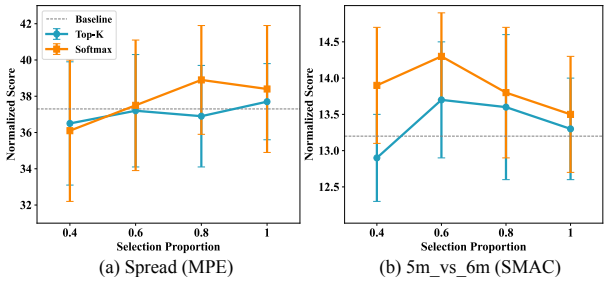


Figure 4: **Module ablation.** The normalized average score of the downstream policy with ablation variants.



(a) Spread (MPE) (b) 5m_vs_6m (SMAC)

Figure 5: **Parameter ablation.** Ablation studies on the selection proportion and the select mechanism type across different datasets.

the synthesis fidelity. Second, the inherently low quality of random datasets hinders the capacity of synthetic datasets to enhance downstream policy performance. These findings underscore the critical role of initial dataset quality in synthesis methods. Additionally, the limited performance gains of MA-SynthER, and in some cases, performance degradation (such as when combined with OMAR in the MPE environment), can be attributed to its failure to consider the inter-agent interaction. To summarize, these results underline the effectiveness of INS across various offline MARL algorithms.

5.3 Dataset Quality Analysis

In order to analyze the dataset quality of synthetic data, we employ multiple data metrics to evaluate the synthetic data synthesized by INS and MA-SynthER. Inspired by prior works (Lu et al., 2024; Lee et al., 2024), we conduct a comprehensive analysis using five metrics: Similarity measures the distance between the synthetic and original data distributions, while Correlation evaluates their statistical relationships, whether casual or not; Oracle Reward represents the true reward of the synthetic data, which indicates optimality; Dynamic MSE evaluates how well the synthetic data aligns with the environment dynamics; Novelty quantifies the uniqueness of the synthetic transition, reflecting the extent of data coverage. Detailed definitions and calculation methods for these metrics are provided in Appendix F. We exclude Oracle Reward and Dynamic MSE for the SMAC environment due to the unavailability of true environment rewards and environment dynamics. Figure 3 illustrates the metrics of synthetic datasets generated from original datasets of different quality. The results demonstrate that our method consistently outperforms MA-SynthER across all metrics and datasets. This superior performance indicates that INS not only enhances the true reward and data coverage of the synthetic data but also preserves its similarity and correlation with the original dataset. Furthermore, it demonstrates INS's ability to capture the dynamic information inherent in the original dataset.

5.4 Ablation Study

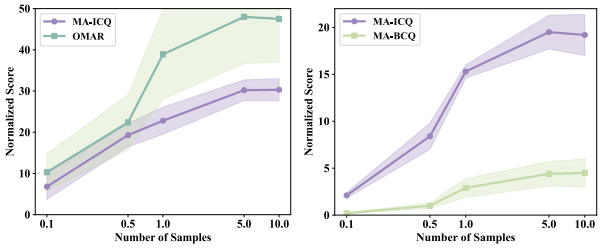
To verify the effectiveness of each component in our method and to analyze the impact of hyperparameters and dataset size, we conduct a series of ablation experiments. All experiments are performed in both MPE and SMAC environments, with MA-ICQ serving as the downstream policy algorithm.

Module Ablation. We evaluate the contribution of key modules through several ablations. These include replacing sparse attention with dense attention (w/o sparse), removing the select mechanism retaining all synthetic data (w/o select), and replacing the attention module with an MLP, allowing each agent to independently synthesize transitions (w/o attention). Figure 4 illustrates the impact of each module on dataset quality and policy performance. The results demonstrate that the attention mechanism significantly enhances the ability of INS to learn inter-agent interactions, improving synthetic data accuracy and policy performance. The select mechanism effectively prioritizes high-value transitions, which proves crucial in complex environments such as SMAC. Additionally, sparse attention reduces redundant information in agent interactions, mitigating distraction impact in the denoising process. We also analyze the metrics of synthetic data generated by different ablation variants. As shown in Figure 3, the select mechanism enhances the true reward of synthetic data,

while sparse attention improves the accuracy and state-action coverage of the method. Details of the ablation variants and more results are provided in Appendix G.3.

Parameter Ablation. We investigate the effect of selection proportion η and select mechanism type within INS. As mentioned in previous sections, the selection proportion represents a trade-off between dataset diversity and quality, with higher proportions favoring high-value transitions at the cost of diversity. As shown in Figure 5, the relationship between policy performance and the selection proportion varies across different environments. In the Spread environment, the offline MARL algorithm benefits from high-value transitions in the dataset, while in 5m_vs_6m, agents require a more diverse offline dataset to achieve optimal performance. We find that the selection proportion of 0.8 provides sufficient performance on different datasets, so we fix it in our primary experiments. Additionally, our experiments on select mechanisms show that the Softmax approach used in primary experiments balances high-value distribution and dataset diversity. Across various selection proportions in different environments, the Softmax select mechanism achieves superior performance compared to the Top-K select mechanism.

Synthetic Dataset Size. In order to investigate the relationship between synthetic dataset size and downstream policy performance, we conduct experiments with varying dataset sizes. We select five different dataset sizes spanning two orders of magnitude: $\{0.1M, 0.5M, 1.0M, 5.0M, 10.0M\}$ transitions. As shown in Figure 6, as the dataset size increases, policy performance improves. However, the improvement does not scale linearly with the dataset size. Similar to findings reported by He et al. (2023) and Lu et al. (2024), the performance tends to saturate as the dataset size increases beyond a certain threshold. Based on these observations, we conclude that a dataset size of $5M$ offers a reasonable balance between performance and computational cost.



(a) Spread (MPE) (b) 5m_vs_6m (SMAC)

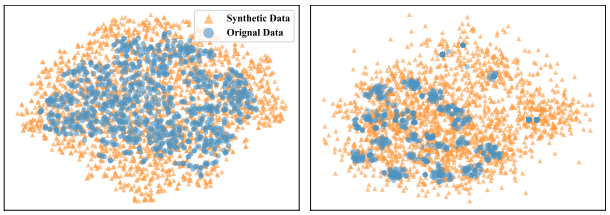
Figure 6: **Impact of synthetic dataset size.** Each curve displays the mean and standard deviation of the normalized score across 5 random seeds.

5.5 VISUALIZATION

To provide intuitive insights into the effectiveness of INS in synthesizing multi-agent interaction datasets, we present four types of visualizations: data coverage, attention weights, value distributions, and reward distributions. These visualizations aim to demonstrate the improvements in dataset quality, the impact of the sparse attention mechanism, and the effect on value and reward distributions.

Data Coverage. To assess the improvement in dataset coverage, we employ t-SNE (van der Maaten & Hinton, 2008) to project both the generated data and the original (o, a) pairs onto a 2D space for both MPE and SMAC, as illustrated in Figure 7. Our analysis reveals that INS not only fills in parts of the data distribution that are not well explored by the real interaction but also augments the empty spaces around the existing transition distribution. In other words, it demonstrates an interpolation effect, potentially filling gaps in the original dataset distribution.

Attention Weights. To elucidate the impact of our attention mechanism on agent interactions, we visualize the attention weights during the synthesis process across multiple multi-agent scenarios. Specifically, we use the real transitions as input for the final diffusion step in INS, then synthesize transition using sparse and dense attention mechanisms, visualizing their respective attention weights. Additionally, we compute the inverse distance $1/d(i, j)$ between each agent pair (i, j) to reflect the true interaction relationships between agents. As shown in Figure 8, sparse attention effectively focuses on agents that are more likely to interact based on their proximity. In contrast, dense attention distributes focus more broadly, disregarding agents' relative distances, which introduces irrelevant interactions and reduces the fidelity of the synthetic dataset. Overall, sparse attention offers two key advantages: lower computational cost during synthesis and less distraction from irrelevant agents. These advantages contribute to more efficient and accurate multi-agent interaction synthesis.



(a) Spread (MPE) (b) 5m_vs_6m (SMAC)

Figure 7: **Data coverage visualization.** Visualization of a t-SNE projection on the observation-action space of the original dataset and INS synthetic dataset.

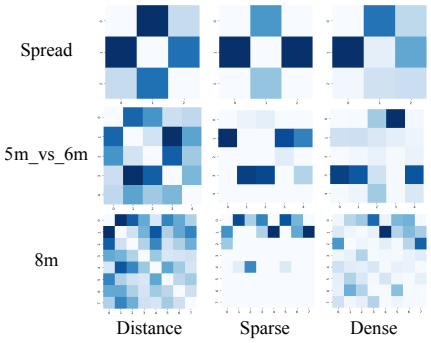


Figure 8: **Attention visualization.** Visualization of attention weights in the synthesis process.

Oracle Reward and Value Distribution. To further evaluate the impact of our method on the synthetic dataset, we visualize the oracle reward distribution (in Appendix G.1) and the actual value distribution (in Appendix G.2) of the original dataset and synthetic datasets. The results indicate that the select mechanism introduced in INS's *Phase B* guides the synthetic dataset toward regions with higher rewards, effectively improving the actual value distribution of the offline MARL policy.

5.6 Synthesize from Small Datasets

We evaluate the ability of INS to synthesize transitions from limited-size datasets by subsampling each dataset in proportion to its original size. Specifically, we select $\{10\%, 50\%, 100\%\}$ of the original dataset as training data for INS, and then assess the policy performance on the corresponding synthetic data. The denoising network uses the same hyperparameters as in the main evaluation. The experimental results in Table 2 show that our method can synthesize effective transitions even when using only 10% of the original data, with the performance comparable to that of the original dataset. These results validate the capability of INS to synthesize high-quality datasets from small-size multi-agent datasets, taking a step towards addressing the data scarcity in real-world offline MARL tasks.

Table 2: **Synthesize from small datasets.** We only present two expert/good dataset results for brevity, with additional results provided in Appendix G.4.

Dataset	***Original***	**10%**	**50%**	**100%**
Spread	*102.8*	102.3	106.3	107.0
5m_vs_6m	*16.3*	16.3	16.8	17.2

6 Conclusion

In this paper, we propose INteraction-aware Synthesis (INS), a method that leverages diffusion models to synthesize high-quality datasets for offline MARL. INS employs a sparse attention mechanism for inter-agent interactions and selects transitions based on a value estimator. The method is orthogonal to existing offline MARL algorithms and adaptable to datasets with both discrete and continuous action spaces. Our experiments demonstrate that datasets synthesized by INS lead to superior policy performance and improved dataset metrics compared to the original datasets and those synthesized by baseline methods. Additionally, INS can synthesize datasets using as little as 10% of the original data while maintaining comparable policy performance. INS enables MARL agents to benefit from synthetic datasets, taking a step towards effective training on scarce real-world datasets.

Limitations. Real-world multi-agent systems often involve a variable number of agents, making dataset synthesis for varying agent numbers a promising research direction. Additionally, given the high computational demands of diffusion models, acceleration techniques will be employed to scale our method to large-scale multi-agent systems. Finally, to balance diversity and fidelity, we did not apply guidance in INS. However, applying guided diffusion models to specific domains (e.g., safety MARL) presents an interesting future direction.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In International Conference on Machine Learning, pp. 104–114. PMLR, 2020.

Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. Applied Sciences, 11(11):4948, 2021.

Jiajun Chai, Yuqian Fu, Dongbin Zhao, and Yuanheng Zhu. Aligning credit for multi-agent cooperation via model-based counterfactual imagination. In Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, pp. 281–289, 2024.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog Bits: Generating discrete data using diffusion models with self-conditioning. In International Conference on Learning Representations, 2023.

Daesol Cho, Dongseok Shim, and H Jin Kim. S2P: State-conditioned image synthesis for data augmentation in offline reinforcement learning. Advances in Neural Information Processing Systems, 35:11534–11546, 2022.

Nicholas Corrado and Josiah P. Hanna. Understanding when dynamics-invariant data augmentations benefit model-free reinforcement learning updates. In International Conference on Learning Representations, 2024.

Alex DeWeese and Guannan Qu. Locally interdependent multi-agent mdp: Theoretical framework for decentralized agents with dynamic dependencies. In International Conference on Machine Learning, 2024.

Zibin Dong, Yifu Yuan, Jianye Hao, Fei Ni, Yi Ma, Pengyi Li, and Yan Zheng. CleanDiffuser: An easy-to-use modularized library for diffusion models in decision making. arXiv preprint arXiv:2406.09509, 2024.

Edgar C Fieller, Herman O Hartley, and Egon S Pearson. Tests for rank correlation coefficients. i. Biometrika, 44(3/4):470–481, 1957.

Claude Formanek, Asad Jeewa, Jonathan Shock, and Arnu Pretorius. Off-the-Grid MARL: Datasets with baselines for offline multi-agent reinforcement learning. arXiv preprint arXiv:2302.00521, 2023.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv:2004.07219, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In International Conference on Machine Learning, pp. 2052–2062. PMLR, 2019.

Michael Georgeff. Communication and interaction in multi-agent planning. In Readings in Distributed Artificial Intelligence, pp. 200–204. Elsevier, 1988.

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. Advances in Neural Information Processing Systems, 36:64896–64917, 2024.

Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? In International Conference on Learning Representations, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33:6840–6851, 2020.

Guangzheng Hu, Haoran Li, Shasha Liu, Yuanheng Zhu, and Dongbin Zhao. Neuronsmae: a novel multi-agent reinforcement learning environment for cooperative and competitive multi-robot tasks. In International Joint Conference on Neural Networks, pp. 1–8, 2023.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In International Conference on Machine Learning, pp. 9902–9915. PMLR, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. Advances in Neural Information Processing Systems, 35:26565–26577, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. Advances in Neural Information Processing Systems, 33:1179–1191, 2020.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. Advances in Neural Information Processing Systems, 33:19884–19895, 2020.

Jaewoo Lee, Sujin Yun, Taeyoung Yun, and Jinkyoo Park. Gta: Generative trajectory augmentation with guidance for offline reinforcement learning. arXiv preprint arXiv:2405.16907, 2024.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020.

Wenhao Li, Bo Jin, and Xiangfeng Wang. Sparsemaac: Sparse attention for multi-agent reinforcement learning. In Database Systems for Advanced Applications: DASFAA 2019 International Workshops: BDMS, BDQM, and GDMA, Chiang Mai, Thailand, April 22–25, 2019, Proceedings 24, pp. 96–110. Springer, 2019.

Zhuoran Li, Ling Pan, and Longbo Huang. Beyond conservatism: Diffusion policies in offline multi-agent reinforcement learning. arXiv preprint arXiv:2307.01472, 2023a.

Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. Synthetic data generation with large language models for text classification: Potential and limitations. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 10443–10461, 2023b.

Shunyu Liu, Jie Song, Yihe Zhou, Na Yu, Kaixuan Chen, Zunlei Feng, and Mingli Song. Interaction pattern disentangling for multi-agent reinforcement learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–15, 2024.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6382–6393, 2017.

Cong Lu, Philip Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. Advances in Neural Information Processing Systems, 36:46323–46344, 2024.

Chengdong Ma, Aming Li, Yali Du, Hao Dong, and Yaodong Yang. Efficient and scalable reinforcement learning for large-scale network control. Nature Machine Intelligence, 9 2024. ISSN 2522-5839.

Guozheng Ma, Zhen Wang, Zhecheng Yuan, Xueqian Wang, Bo Yuan, and Dacheng Tao. A comprehensive survey of data augmentation in visual reinforcement learning. arXiv preprint arXiv:2210.04561, 2022.

Andre Martins and Ramon Astudillo. From Softmax to Sparsemax: A sparse model of attention and multi-label classification. In International Conference on Machine Learning, pp. 1614–1623. PMLR, 2016.

Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. Journal of the American Statistical Association, 46(253):68–78, 1951.

Frans A Oliehoek, Christopher Amato, et al. A concise introduction to decentralized POMDPs, volume 1. Springer, 1st edition, 2016.

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

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. Training question answering models from synthetic data. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 5811–5826, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In Proceedings of the 20th International Conference on Neural Information Processing Systems, pp. 1177–1184, 2007.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-assisted Intervention, pp. 234–241. Springer, 2015.

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. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, pp. 2186–2188, 2019.

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 Conference on Lifelong Learning Agents, pp. 470–517. PMLR, 2022.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Machine learning proceedings 1990, pp. 216–224. Elsevier, 1990.

Qi Tian, Kun Kuang, Furui Liu, and Baoxiang Wang. Learning from good trajectories in offline multi-agent reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pp. 11672–11680, 2023.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. MLP-Mixer: An all-MLP architecture for vision. Advances in Neural Information Processing Systems, 34: 24261–24272, 2021.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. Journal of Machine Learning Research, 9(86):2579–2605, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6000–6010, 2017.

Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. Bootstrapped transformer for offline reinforcement learning. Advances in Neural Information Processing Systems, 35:34748–34761, 2022.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In International Conference on Learning Representations, 2023.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361, 2019.

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. Advances in Neural Information Processing Systems, 34:10299–10312, 2021.

Ruiqi Zhang, Jing Hou, Florian Walter, Shangding Gu, Jiayi Guan, Florian Röhrbein, Yali Du, Panpan Cai, Guang Chen, and Alois Knoll. Multi-agent reinforcement learning for autonomous driving: A survey. arXiv preprint arXiv:2408.09675, 2024.

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. arXiv preprint arXiv:2305.17330, 2023a.

Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Yong Yu, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. arXiv preprint arXiv:2311.01223, 2023b.

Appendix

Table of Contents

A Motivating Example: Fireboy and Watergirl

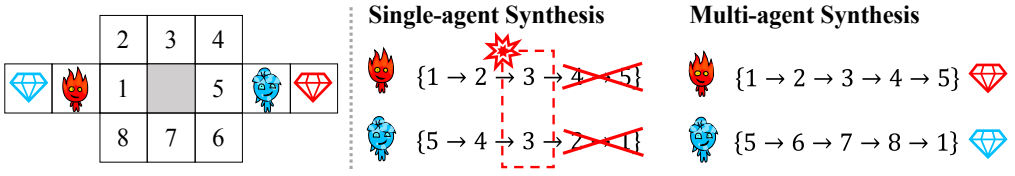


Figure A1: **Motivating example.** Fireboy and watergirl are situated in a grid world where they must collaborate to retrieve the diamonds located behind each other. In this environment, fireboy and watergirl cannot occupy the same grid cell, requiring them to follow different paths to reach their respective diamonds.

In this section, we present a simple motivating example to illustrate the importance of inter-agent interactions during data synthesis. As shown in Figure A1, two agents must select different paths to reach opposite ends of the grid, as they cannot occupy the same grid cell. The original dataset contains transitions where each agent follows either the upper or lower path. However, applying single-agent data synthesis methods independently, without considering inter-agent interactions, leads to the scenario depicted in Figure A1 (mid), where both agents (fireboy and watergirl) are synthesized to follow the same upper path. This transition is infeasible in the real environment, resulting in a distorted synthetic dataset. By considering inter-agent interactions, multi-agent data synthesis can effectively avoid this issue and synthesize high-quality datasets, as shown in Figure A1 (right).

B Details of the Preliminary Study

In this section, we provide more details of the experimental setting of our preliminary study, as illustrated in Figure 1. We compare our method against three baseline approaches: Additive Noise, VAE Augmented, and MA-SynthER. The Additive Noise method (Laskin et al., 2020) augments the original dataset by introducing Gaussian noise. The VAE Augmented approach employs a variational autoencoder to learn from the original dataset and subsequently generate new data through sampling. MA-SynthER, our multi-agent implementation of SynthER (Lu et al., 2024) utilizes a diffusion model based on the MLP-Mixer architecture (Tolstikhin et al., 2021). These models synthesize transitions for each agent independently, without considering interactions between multiple agents. To adapt the Additive Noise, VAE Augmented method, and MA-SynthER for discrete action environments, we incorporated our proposed bit action module into these methods.

Specifically, Figure 1 (left) compares the performance of MA-ICQ using various purely synthetic datasets in MPE and SMAC environments. The dashed line indicates the policy performance trained on the original dataset. Our method demonstrates superior performance compared to both the original dataset and the baseline synthetic dataset, indicating the efficacy of INS synthetic dataset in enhancing downstream policy training. Additive Noise and VAE Augmented methods yield results comparable to or even worse than the original dataset. MA-SynthER shows a modest improvement over the original dataset but underperforms relative to our proposed method, potentially due to its limited consideration of inter-agent interactions and dataset quality enhancement. Figure 1 (right) presents a comparison of various dataset metrics across different synthetic datasets. INS demonstrates consistently high performance across all dataset metrics, suggesting the effectiveness of our method in synthesizing high-quality synthetic datasets.

C Pseudocode of Interaction-aware Synthesis

We describe the training and synthesis processes of INS, as shown in Algorithm 1.

Algorithm 1 Interaction-aware Synthesis

Input: Offline MARL dataset $\mathcal{D}_{\text{ori.}}$, number of candidates $\mathcal{B}$ and selection proportion η

Output: Synthetic dataset $\mathcal{D}_{\text{syn.}}$ and trained joint policy $\boldsymbol{\pi}$

// Diffusion model training process

Transform $\mathcal{D}_{\text{ori.}}$ into $(|\mathcal{D}_{\text{ori.}}|, n, d_x)$ shaped tensor

Update diffusion model $p_{\text{dm}}(\boldsymbol{x})$ and state value estimator $V_\phi(\boldsymbol{o})$ with samples from $\mathcal{D}_{\text{ori.}}$ using Eq. (2) and Eq. (8), respectively

// Dataset synthesis process

Initialize synthetic dataset $\mathcal{D}_{\text{syn.}} = \emptyset$ and transition candidate set $\mathcal{D}_{\text{can.}} = \emptyset$

for $i = 1, \ldots, \mathcal{B}$ **do**

  Sample the i-th new joint transition $\boldsymbol{x}^i = (\boldsymbol{o}^i, \boldsymbol{a}^i, \boldsymbol{o'}^i, r^i)$ using diffusion model $p_{\text{dm}}(\boldsymbol{x})$

  Add transitions into the candidate set $\mathcal{D}_{\text{can.}}$

  Calculate the value of each candidate transition using state value estimator $V_\phi(\boldsymbol{o}^i)$

end for

Select a proportion η of the candidate set by their values

Add all transitions of selected candidates to $\mathcal{D}_{\text{syn.}}$

// Policy training process

Initialize and train the offline MARL policy $\boldsymbol{\pi}$ using $\mathcal{D}_{\text{syn.}}$

return $\mathcal{D}_{\text{syn.}}$ and $\boldsymbol{\pi}$

D Main Differences between INS and Related Works

Diffusion Models in Offline MARL. In this section, we discuss related works, MTDiff-S (He et al., 2024), SynthER (Lu et al., 2024) and MADiff (Zhu et al., 2023a), to clarify the differences between our approach and existing methods. Motivationally, both MTDiff-S and SynthER are designed for single-agent settings. MTDiff-S generates data at the trajectory level for unknown tasks, focusing on multi-task scenarios, which is why it is not included as a baseline in our work. SynthER, on the other hand, generates data at the transition level and is applicable to both online and offline reinforcement learning. Unlike these methods, INS is specifically designed for multi-agent environments, incorporating inter-agent interactions during synthesis to produce enhanced datasets for improving offline MARL performance. Technically, MTDiff-S adopts a Transformer architecture, treating trajectory generation as a sequence modeling problem, while SynthER employs an MLP-Mixer architecture. In contrast, INS uses a Sparse Transformer Encoder architecture, treating each agent as an individual token input. Additionally, neither MTDiff-S nor SynthER can handle discrete action spaces, whereas INS achieves this compatibility action spaces through bit action. MADiff is an offline MARL method, which models a planner as a return-conditional diffusion model to maximize the cumulative reward. At each time step, MADiff generates a trajectory of length $H - 1$ based on the agent's current or historical observations and then uses only the observations (o_t, o_{t+1}) along with an inverse dynamics model to determine the action a_t.

- Theoretically, data synthesis in reinforcement learning often refers to the ability to generate data that can be used to train policies, rather than directly engaging in decision-making (as a planner or policy) (Dong et al., 2024; Cho et al., 2022). In two overviews on diffusion models for RL (Dong et al., 2024; Zhu et al., 2023b), MADiff is consistently categorized as a planner (notably, the first

Table A1: A comparison of INS and the related works.

Method	Key Feature	Input	Output	Requires Real State	Requires Policy	Action Space
SynthER	Data Synthesizer (single-agent)	None	$\{o, a, o', r\}$	No	No	Continuous
MTDiff-S	Multi-task Data Synthesizer (single-agent)	h_t	$\{o_{t+i}, a_{t+i}, o_{t+i+1}, r_{t+i}\}_{i=1,\dots,H-1}$	Yes	No	Continuous
MADiff	Planner	o_t, r	$\{\boldsymbol{o}_{t+i}\}_{i=1,\dots,H-1}$	Yes	No	None
MOMA-PPO	Model-based MARL (Dyna)	h_t, π	$\{(h_j, a_j, r_j)\}_{j=t,\dots,t+k-1}$	Yes	Yes	None
INS (ours)	Data Synthesizer	None	$\{\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}', r\}$	No	No	Continuous/Discrete

author of Zhu et al. (2023b) is also the first author of MADiff). In contrast, INS and SynthER generate synthetic data without maximizing rewards or selecting actions, and should therefore be categorized under Data Synthesizers. Thus, we believe that MADiff and our method are fundamentally different. Furthermore, MADiff generates observation sequences based on the agent's current state and only uses the next observation (o_t, o_{t+1}) to produce an action, meaning it cannot generate a dataset from scratch that can be used for policy training. In contrast, our method requires no input and can generate a complete dataset of transitions $\{(o_t, a_t, o_{t+1}, r_t)\}$ suitable for subsequent offline multi-agent reinforcement learning.

- Technically, while both MADiff and our approach leverage the generative capabilities of diffusion models, we believe this similarity may cause confusion. In terms of the learned targets, MADiff trains on observation sequences o_t, while INS learns from transitions (o_t, a_t, o_{t+1}, r_t). When it comes to the generated output, MADiff produces observation sequences for inverse decision-making, while INS generates multiple transitions (o_t, a_t, o_{t+1}, r_t) that form a dataset. In terms of training, MADiff uses classifier-free guidance, requiring the agent's current observation and team rewards as inputs, whereas our method needs no input and can generate multi-agent datasets from scratch.

Model-based MARL. We note that our method shares some similarities with model-based MARL, particularly with Dyna-like algorithms (Sutton, 1990), which involve unrolling trajectories "in the imagination" of the world model to optimize a policy. In contrast, INS is a data synthesis method that generates new experiences without *starting from a real state or the current policy*, and the generated experiences follow the distribution of the dataset. Moreover, INS is an orthogonal method that can be combined with forward dynamics models by generating initial states via diffusion, potentially enhancing diversity. A summary of the differences between INS and the related works mentioned above is provided in Table A1.

E Comparison of the Sparsemax in Related Works

In OPT (Liu et al., 2024) and SparseMAAC (Li et al., 2019), sparsemax is used to extract relationships between the real observations of multiple agents, enabling agents to focus on specific targets during inference, which helps in effective decision-making. Therefore, in these works, the input to sparsemax is the encoding of agent observations or observation-action pairs, and the output is used for subsequent action generation or Q-value computation. In contrast, our work is a generative task, where sparsemax is applied during the denoising process of transitions (which are initially sampled entirely from noise). During training, sparsemax learns the interaction patterns present in the dataset, and during synthesis, it progressively incorporates agent interactions into the noise, rather than extracting pre-existing relationships. Therefore, we believe the role of sparsemax in INS is fundamentally different from its role in the aforementioned works.

F Dataset Quality Metrics

In this section, we define the metrics employed to assess the quality of the synthetic dataset. Our evaluation framework consists of five distinct metrics: Similarity, Correlation, Oracle Reward, Dynamic Mean Squared Error (MSE), and Novelty. The Similarity and Correlation metrics quantify the statistical fidelity and relational consistency between the synthetic and original datasets, while the Oracle Reward metric evaluates the true reward magnitudes associated with the synthetic data. The Dynamic MSE metric assesses the fidelity of the synthetic data to the environment dynamics,

quantifying how accurately it represents the transitions. The Novelty metric quantifies the uniqueness and diversity of the synthetic dataset, providing insights into its coverage of the observation-action space. All metrics are computed from $200K$ samples from each synthetic dataset. The formal mathematical definitions and computational procedures of these metrics are provided below.

F.1 SIMILARITY

This metric means Kolmogorov-Smirnov (KS) distance (Massey Jr, 1951), which measures the maximum distance between the cumulative distribution functions of the original and synthetic datasets. The Similarity is defined as:

$$\text{Similarity}(\mathcal{D}_{\text{ori.}}, \mathcal{D}_{\text{syn.}}) = \sup_{\boldsymbol{x}} |F_{\text{ori.}}(\boldsymbol{x}) - F_{\text{syn.}}(\boldsymbol{x})|, \tag{A1}$$

where $F_{\text{ori.}}(\boldsymbol{x})$ and $F_{\text{syn.}}(\boldsymbol{x})$ are the cumulative distribution functions of the original and synthetic datasets, respectively.

F.2 CORRELATION

This metric is pairwise rank Pearson correlation (Fieller et al., 1957), which measures the statistical relationship between the original and synthetic datasets. The Correlation is defined as:

$$\text{Correlation}(\mathcal{D}_{\text{ori.}}, \mathcal{D}_{\text{syn.}}) = \frac{\sum \left(\text{rank}_{\text{ori.}}(\boldsymbol{x}_i) - \bar{r}_{\text{ori.}}\right)\left(\text{rank}_{\text{syn.}}(\boldsymbol{x}_i) - \bar{r}_{\text{syn.}}\right)}{\sqrt{\sum \left(\text{rank}_{\text{ori.}}(\boldsymbol{x}_i) - \bar{r}_{\text{ori.}}\right)^2 \sum \left(\text{rank}_{\text{syn.}}(\boldsymbol{x}_i) - \bar{r}_{\text{syn.}}\right)^2}}, \tag{A2}$$

where $\text{rank}(\boldsymbol{x}_i)$ is the rank of joint transition $\boldsymbol{x}_i$, and $\bar{r}$ is the mean rank of the dataset.

F.3 ORACLE REWARD

This metric is the average true reward of the synthetic dataset, which measures the optimality of synthetic data. The Oracle Reward is defined as:

$$\text{OracleReward}(\mathcal{D}_{\text{syn.}}) = \frac{1}{|\mathcal{D}_{\text{syn.}}|} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{syn.}}} r_{\text{true}}, \tag{A3}$$

where the true reward r_{true} is obtained by querying the environment with the generated joint observations and actions $(\boldsymbol{o}, \boldsymbol{a})$.

F.4 DYNAMIC MSE

This metric is the mean squared error (MSE) between the predicted next observation and the true next observation, which evaluates how well synthetic data fits the environment dynamics. Prior to error computation, we normalize each observation. The Dynamic MSE is defined as:

$$\text{DynamicMSE}(\mathcal{D}_{\text{syn.}}) = \frac{1}{|\mathcal{D}_{\text{syn.}}|} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{syn.}}} (\mathcal{P}(\boldsymbol{o}, \boldsymbol{a}) - \boldsymbol{o}')^2, \tag{A4}$$

where $\mathcal{P}(\boldsymbol{o}, \boldsymbol{a})$ is the true next state by the environment dynamics.

F.5 NOVELTY

This metric is the average distance between synthetic data and the closest original dataset, which quantifies the uniqueness of datasets. The Novelty is defined as:

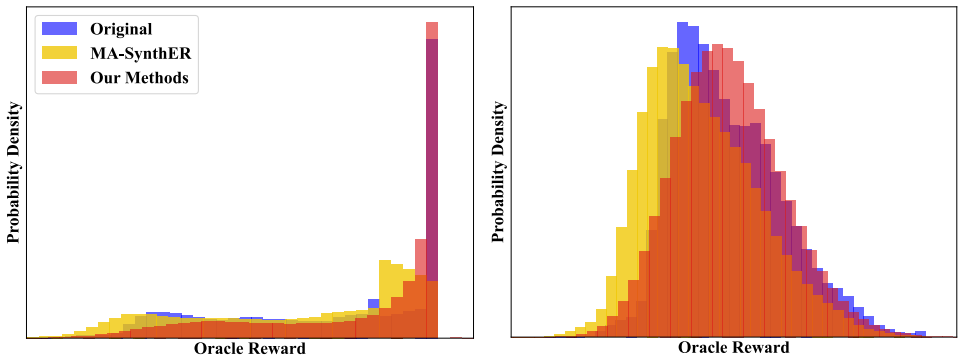
$$\text{Novelty}(\mathcal{D}_{\text{syn.}}, \mathcal{D}_{\text{ori.}}) = \frac{1}{|\mathcal{D}_{\text{syn}}|} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{syn.}}} \min_{\bar{\boldsymbol{x}} \in \mathcal{D}_{\text{ori.}}} \left((\boldsymbol{o}, \boldsymbol{a}) - (\bar{\boldsymbol{o}}, \bar{\boldsymbol{a}})\right)^2, \tag{A5}$$

where $\bar{\boldsymbol{x}}$ is the nearest neighbor of $\boldsymbol{x}$ in the original dataset.

G Additional Experimental Results

G.1 Oracle Reward Distribution Visualization

We visualize the reward of the original dataset and real reward computed from the environment using observation and action generated from MA-SynthER and INS to visualize that INS can improve the data distribution to high reward region while preserving environmental dynamics. Figure A2 presents these distributions. The results demonstrate that our method effectively enhances the proportion of high-reward transitions in the dataset while simultaneously broadening the overall range of rewards compared to the original dataset. This dual improvement is particularly beneficial for subsequent policy learning, as it provides a richer and more diverse reward landscape for agents to learn from.

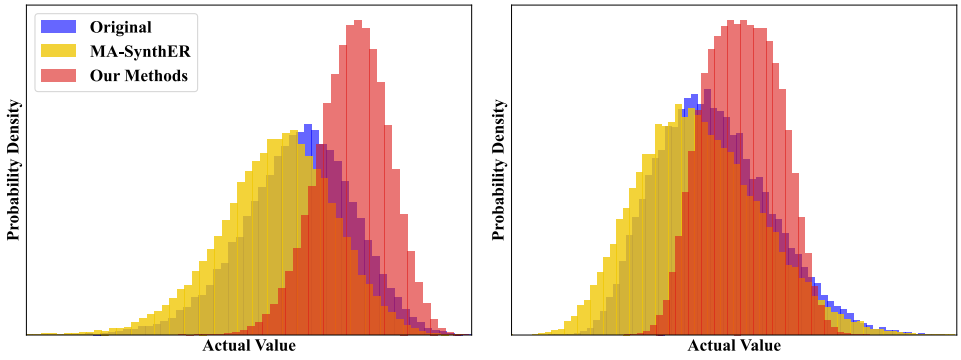


(a) Spread-Expert (b) Spread-Medium

Figure A2: **Oracle reward distribution visualization.** Comparison of the oracle reward distributions among original, MA-SynthER, and INS datasets.

G.2 Actual Value Distribution Visualization

To further evaluate the impact of our method on dataset quality, we visualize the actual value distributions of the original, MA-SynthER, and INS datasets, as shown in Figure A3. We first train a policy on the dataset and then obtain the actual value for the trained policy by running it in the real environment. The results demonstrate that our method effectively enhances the value distribution of transitions in the dataset under the policy, while the value distribution of MA-SynthER synthetic data exhibits a decrease compared to the original dataset.



(a) Spread-Expert (b) Spread-Medium

Figure A3: **Actual value distribution visualization.** Comparison of the actual value distributions among original, MA-SynthER, and INS datasets.

G.3 Module Ablation

We conduct more ablation studies on the components of INS across 14 datasets from 4 different scenarios. These include replacing the sparse attention mechanism in the Transformer Encoder with a dense attention mechanism (w/o sparse), removing the select mechanism retaining all synthetic data (w/o select), and replacing the attention module with an MLP, allowing each agent to independently synthesize transitions (w/o attention). In the w/o attention variant, we concatenate the transitions of individual agents and replace the attention module in the Transformer Encoder with multiple MLPs, removing the dot-product and Sparsemax components. This modification results in each agent synthesizing transitions independently, disregarding interactions between agents. The results are shown in the Figure A4. The results demonstrate that the attention mechanism significantly enhances agents' ability to learn inter-agent interactions, improving synthetic data accuracy and policy performance. The select mechanism effectively screens high-value transitions, which is particularly crucial in complex environments like SMAC. Additionally, the sparse attention mechanism reduces redundant information in agent interactions, mitigating distraction impact.

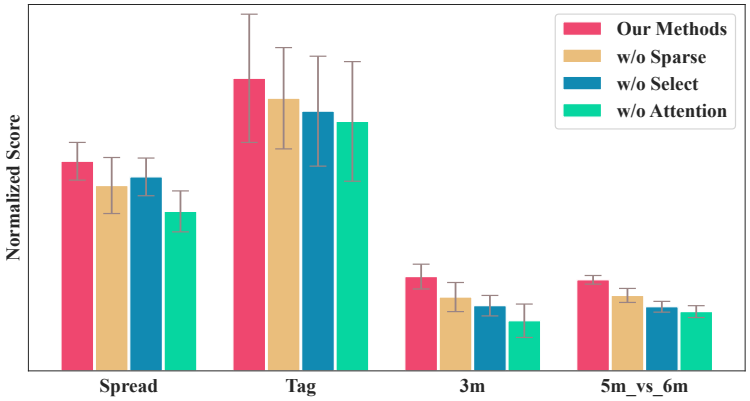


Figure A4: **Module ablation on different datasets.** The normalized average score of the downstream policy with INS ablation variants.

G.4 Synthesize from Small Datasets

We evaluate the scalability of our method across 8 datasets from 4 different scenarios. The results show that, when using only 10% of the original dataset, our method often achieves performance comparable to that of the original dataset. With 50% of the dataset, the policy consistently outperforms the original dataset. The quality of the original dataset also impacts scalability. On expert-level datasets, where a higher proportion of high-quality samples remain even after downscaling, our method demonstrates better scalability, particularly when using just 10% of the data.

Table A2: **Synthesize transitions from a small dataset.** We use four different scenarios for exploration, each scenario includes two different quality datasets.

Dataset		*Original*	10%	50%	100%
Spread	Expert	*102.8*	101.9	106.8	107.0
	Medium	*26.8*	24.5	28.4	30.2
Tag	Expert	*111.7*	110.9	113.1	114.2
	Medium	*62.3*	61.2	63.5	64.7
3m	Good	*18.1*	18.0	19.3	19.9
	Medium	*16.9*	15.6	17.8	18.5
5m_vs_6m	Good	*16.3*	16.3	16.8	17.2
	Medium	*12.8*	11.9	12.9	13.9

G.5 Computational Efficiency of Dataset Synthesis

To evaluate the computational efficiency of INS, we measured the time taken by INS to generate $1M$ transitions and compared it with the time required to collect $1M$ transitions through environment interactions using trained policies. The results in Table A3 show that INS is significantly more time-efficient than collecting data via direct environment interactions, while still maintaining high dataset quality. We believe this demonstrates that dataset synthesis is a promising avenue for MARL, where generative methods like INS can be employed to synthesize multi-agent datasets, reducing the time and challenges associated with data collection in real-world environments.

Table A3: **Computational efficiency of INS.**

Scenario	INS	Collecting by agents
spread	1540s	2728s (MAPPO)
3m	2396s	9243s (QMIX)

H Information on Offline MARL Datasets

(a) MPE (b) SMAC

Figure A5: **MPE and SMAC environment examples.**

H.1 MPE Datasets

Multi-Agent Particle Environment (MPE) (Lowe et al., 2017) is a multi-agent particle environment where particles can execute continuous observations and actions, as illustrated in Figure A5(a). Our study employed three distinct MPE scenarios. The Spread environment consists of three agents and three landmarks, where the agents' objective is to learn collision avoidance while ensuring complete landmark coverage. The Tag scenario involves one pre-trained prey, three predators, and two obstacles, requiring the predators to collaborate in capturing the prey. The World environment, similar to Tag, includes one pre-trained prey and three predators; however, the prey agent must locate food within the map and can utilize forest areas for hiddening.

For our MPE experiments, we used dataset[1] provided by OMAR (Pan et al., 2022). This comprehensive dataset comprises multiple subsets of varying quality. The random datasets are generated using a randomly initialized policy for one million steps. The medium replay (md-replay) dataset is obtained by recording all samples in the buffer when training achieves a medium performance level. The medium dataset was derived from a partially trained policy with medium performance, while the expert dataset was generated using a fully trained policy. It is worth noting that OMAR utilizes an earlier version of MPE, which allows agents to potentially receive different rewards. However, we find that the different rewards for agents only occur infrequently and do not significantly impact the predominantly cooperative nature of the environments.

[1] https://github.com/ling-pan/OMAR

H.2 SMAC DATASETS

The StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) is a widely recognized benchmark in the MARL field, as shown in Figure A5(b). The SMAC environment simulates battle scenarios between two competing teams. One team is controlled by built-in artificial intelligence, while the other is managed by policies learned through MARL algorithms. Our experiments utilized three distinct maps: the 3m map, where both sides command three marines; the 8m map, featuring eight marines on each side; and the 5m_vs_6m map, where our team controls five marines against an enemy force of six.

For our SMAC experiments, we employ the off-the-grid dataset[1] (Formanek et al., 2023), which provides good, medium, and poor quality datasets for each map. These datasets were generated using three independently trained QMIX policies. To enhance behavioral diversity, a modest amount of exploration noise was introduced to the policies during data collection.

I OFFLINE MARL ALGORITHM IMPLEMENTATION

This section details the implementation of various offline MARL algorithms.

- **MA-ICQ**[2] (Yang et al., 2021) implicit constraint Q-learning to multi-agent settings, introducing a decomposed multi-agent joint-policy under implicit constraints. By relying solely on state-action pairs provided in the dataset, this approach effectively mitigates extrapolation errors.
- **MA-CQL**[3] (Kumar et al., 2020) adapts the conservative Q-learning framework to multi-agent environments. This algorithm addresses the overestimation problem by adjusting Q-values for both policy samples and dataset state-action pairs, resulting in more conservative and reliable policy updates.
- **MA-BCQ**[4] (Wu et al., 2019), the multi-agent version of BCQ, implements decentralized multi-agent reinforcement learning using value deviation and transition normalization to achieve coordinated policies.
- **OMAR**[5] (Pan et al., 2022) employs zeroth-order optimization in conjunction with multi-agent CQL, enhancing coordination among agents' policies. In implementing these offline MARL algorithms, we adhere to the hyperparameters specified in their respective original papers to ensure consistency and reproducibility of results.

J MA-SYNTHER IMPLEMENTATION

We implement MA-SynthER, a multi-agent implementation of SynthER (Lu et al., 2024), which extends the synthetic data generation approach to multi-agent settings. MA-SynthER employs a diffusion model based on the MLP-Mixer (Tolstikhin et al., 2021) architecture to generate transition-level data (observation, action, reward, and next observation) for multiple agents. This model synthesizes transitions for each agent independently, without considering interactions between multiple agents. We built MA-SynthER upon the code[6] provided by authors, adapting it to the multi-agent context. Our implementation maintains consistency with the original SynthER by adopting the same hyperparameters and training settings.

[1] https://huggingface.co/datasets/InstaDeepAI/og-marl
[2] https://github.com/YiqinYang/ICQ
[3] https://github.com/aviralkumar2907/CQL
[4] https://github.com/sfujim/BCQ
[5] https://github.com/ling-pan/OMAR
[6] https://github.com/conglu1997/SynthER

K IMPLEMENTATION DETAILS OF INS

K.1 DETAILED NETWORK ARCHITECTURE

In this subsection, we detail the diffusion model architecture of INS, as shown in Figure A6. The architecture begins with a shared embedding layer that processes each agent's transition, denoted as $x_i = (o_i, a_i, r, o_i')$. Subsequently, the embedded transition is projected into query, key, and value vectors, which serve as inputs to a sparse attention mechanism. The output of the attention mechanism is combined with the input through a residual connection and normalized, after which it is transformed by a feedforward neural network with ReLU activation. The block's final representation is obtained by applying layer normalization to the feedforward network's output. The transformer encoder typically stacks this block N times to enhance model depth. Our implementation utilizes a stack of 4 such blocks. To enhance the model's expressive capacity, we implement multi-head attention with 4 heads, extending the sparse attention mechanism. The diffusion model incorporates temporal information by integrating a step embedding vector k. This embedding is concatenated with the agent's transition before being fed into the network.

Extending to Synthesize Global States. For certain MARL algorithms that require a global state s, we can extend the input of the diffusion model to generate the global state s. Specifically, instead of modeling joint local transitions with diffusion tuples of the form $(\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}', r)$, we could model expanded tuples of the form $(\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}', s, r)$, and so on. This allows the method to be applied to value decomposition models that require a global state.

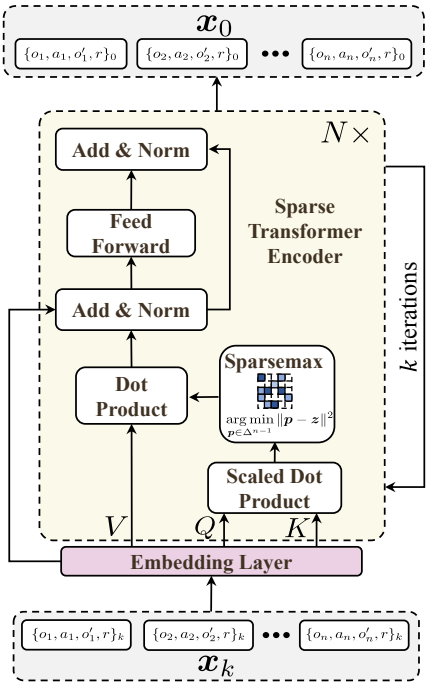


Figure A6: **The detailed diffusion model architecture of INS.**

K.2 SCORE NORMALIZATION

Following Pan et al. (2022) and Fu et al. (2020), the average scores of MPE datasets in Table 1 are normalized by the expert and random scores on each task. Denote the original episodic return as S, then the normalized score S_{norm} is computed as

$$S_{\text{norm}} = 100 \times \frac{S - S_{\text{random}}}{S_{\text{expert}} - S_{\text{random}}}, \tag{A6}$$

where S_{random} and S_{expert} are the average scores of the random and expert datasets, respectively. Besides, the score 0 corresponds to a random policy performance and 100 corresponds to an expert policy performance.

K.3 HYPERPARAMETERS

In this subsection, we first list the key hyperparameters of INS in Table A4. The noise level of the diffusion model is encoded by a Random Fourier Feature (RFF) (Rahimi & Recht, 2007) embedding.

Table A4: INS Hyperparameters.

Hyperparameter	Value
Selection proportion	0.8
Embedding dimension	64
Number of attention heads	4
Number of blocks	2
Dropout	0.1
Batch size	1024
Optimizer	Adam
Learning rate	2×10^{-4}
Weight decay	10^{-4}
Learning rate schedule	Cosine annealing warmup
RFF dimension	16
Training steps	1e6

For the diffusion sampling process, we use the SDE sampler of Karras et al. (2022) with the default hyperparameters used for SynthER (Lu et al., 2024), given in Table A5. In the implementation of downstream offline MARL algorithms, we use the hyperparameters specified in their respective original papers to ensure consistency and reproducibility of results.

Table A5: Diffusion Sampling Hyperparameters following EDM (Karras et al., 2022).

Hyperparameter	Value
Diffusion steps	128
σ_{min}	0.002
σ_{max}	80
S_{churn}	80
S_{min}	0.05
S_{max}	50
S_{noise}	1.003

K.4 COMPUTATIONAL RESOURCES

Most experiments are conducted on a server equipped with an Intel(R) Xeon(R) Gold 6442Y and two NVIDIA RTX A6000 GPUs. The training time of INS is about $7 \sim 9$ hours for each task, the synthesizing time is about $1.5 \sim 3$ hours, and the training time of offline MARL policies varies depending on the specific algorithm.