Diffusion-based Episodes Augmentation for Offline Multi-Agent Reinforcement Learning

Jihwan Oh [1] **Sungnyun Kim** [1] **Gahee Kim** [1] **Sunghwan Kim** [1] **Se-Young Yun** [1]

Abstract

Offline multi-agent reinforcement learning (MARL) is increasingly recognized as crucial for effectively deploying RL algorithms in environments where real-time interaction is impractical, risky, or costly. In the offline setting, learning from a static dataset of past interactions allows for the development of robust and safe policies without the need for live data collection, which can be fraught with challenges. Building on this foundational importance, we present **EAQ**, **E**pisodes **A**ugmentation guided by **Q**-total loss, a novel approach for offline MARL framework utilizing diffusion models. **EAQ** integrates the Q-total function directly into the diffusion model as a guidance to maximize the global returns in an episode, eliminating the need for separate training. Our focus primarily lies on cooperative scenarios, where agents are required to act collectively towards achieving a shared goal—essentially, maximizing global returns. Consequently, we demonstrate that our episodes augmentation in a collaborative manner significantly boosts offline MARL algorithm compared to the original dataset, improving the normalized return by +17.3% and +12.9% for *medium* and *poor* behavioral policies in SMAC simulator, respectively.

1. Introduction

Offline multi-agent reinforcement learning (MARL) (Levine et al., 2020) tackles the difficulties of implementing learning algorithms in situations where immediate interaction is infeasible or carries substantial dangers, such as operating in safety-critical systems like autonomous driving and robotic surgery. This approach, which emphasizes the utilization of a collection of past interaction data, presents a promising alternative to conventional MARL methods that rely on real-time data. By relying on pre-existing datasets, offline MARL enables the development of robust decision-making strategies without the need to continuously gather new data in potentially hazardous situations.

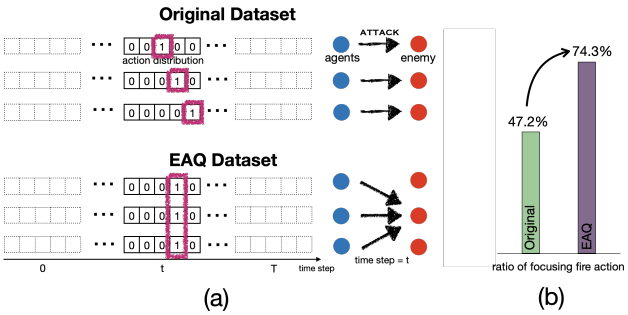

Figure 1. (a) **EAQ** enhances the dataset to involve cooperative behaviors among agents. For instance, in a scenario like *3marines vs 3marines* in StarCraft2, it encourages agents to sequentially target and attack a single enemy, which is a good strategy to win. (b) illustrates the proportion of focusing fire actions on a single enemy when all agents choose to attack. Compared to the original dataset, **EAQ** has successfully increased the frequency of cooperative behaviors, towards achieving a shared goal.

However, collecting static data for offline MARL poses significant challenges, often requiring substantial time, expertise, and financial resources. This process typically involves human experts to check the relevance and accuracy of data, which not only makes it costly but also time-consuming. Hence, such static datasets usually lack the variability and complexity of real-world scenarios, resulting in a limited range of experiences for training MARL algorithms. This may lead to overfitting to the training data and poor generalization in actual environments, ultimately stifling the learning process and restricting the potential of MARL algorithms to adapt effectively to real-world conditions.

In order to tackle these problems, Laskin et al. (2020) and Sinha et al. (2022) have proposed the use of data augmentation techniques in RL, which include introducing random variables into the original state space. The denoising diffu-

[1]Kim Jaechul Graduate School of Artificial Intelligence, KAIST, Seoul, South Korea. Correspondence to: Se-Young Yun <yunseyoung@kaist.ac.kr>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

sion probablistic model, proposed by Ho et al. (2020), recommends the use of data augmentation in single-agent RL and has been supported by the studies of Lu et al. (2024); He et al. (2024). Yet, data augmentation strategies for MARL remain underexplored compared to the field of single-agent RL, although addressing data scarcity and efficiency is particularly crucial in offline MARL given its complex environments and interactions between multiple agents. In this paper, we introduce a novel episodes augmentation method, named as **EAQ**, simplifying the training architecture while significantly enhancing performance through three primary contributions:

Pioneering episodes augmentation for MARL: We compile an offline dataset by incorporating multi-agent trajectories, including observations, actions, global rewards, and terminal states, formatted for a `Conv1D`-based diffusion model. As detailed in Section 3, we describe our approach for effectively synthesizing episodes. To the best of our knowledge, our work is the first to propose a method for augmenting episodes specifically within the context of offline MARL. By identifying and addressing the need for enriched training scenarios, this study opens new avenues for data-centric research and applications in MARL.

Cooperative trajectories augmentation: Based on the prepared data, we guide our diffusion models to augment trajectories to be cooperative, *i.e.*, exhibiting higher state-action values, with Q-total loss. This targeted approach ensures that the augmented trajectories are not only diverse but also strategically cooperative, enhancing the learning efficacy and performance of MARL algorithms in complex scenarios. Our findings indicate that our augmentation method, **EAQ**, promotes increased cooperative behavior among agents. Figure 1 depicts the SMAC (Samvelyan et al., 2019) scenario (*3m vs 3m*), where we observe that **EAQ** exhibits a significant rise in the cooperative actions among agents—represented as all agents attacking a single enemy to achieve maximum global returns.

Simplification of model architecture: Q-total loss guidance (Wang et al., 2023; Ada et al., 2024; Kang et al., 2024; Chen et al., 2024; Li et al., 2023a) directly optimizes the diffusion model to maximize the expected cumulative rewards. This ensures that the diffusion model aligns closely with the objective function of RL, which is to learn from the reward function effectively. However, guiding with Q-total loss also requires a Q-function estimator $Q_\theta^{\text{tot}}(\mathbf{o}_t, \mathbf{a}_t)$ to be trained separately (Wang et al., 2023; Ada et al., 2024; Kang et al., 2024; Chen et al., 2024; Li et al., 2023a). In contrast, our method utilizes only a single diffusion model for episodes augmentation, eliminating the need for estimating Q-function. This not only simplifies the overall model architecture but also reduces the computational overhead.

2. Preliminaries

2.1. Offline multi-agent reinforcement learning

Offline multi-agent reinforcement learning (offline MARL) represents an integration of the unique dynamics and complexities of MARL with distinctive methods and constraints inherent in the offline RL. This integration entails a range of challenges and leverages combined techniques from both domains to address the intricacies of learning and decision-making in multi-agent environments without the requirement of real-time data collection.

Offline MARL is a Decentralized-Partially Observable Markov Decision Process (Dec-POMDP). The framework of Dec-POMDP is described by a tuple $G := < S, A, P, r, \Omega, O, N, \gamma >$. In this context, $s \in S$ denotes all possible environmental states and configurations for the system. $a \in A$ represents the agents' actions, which may change the environment with specific probability. The transition function, $P(s' \mid s, a)$: $S \times A \times S' \rightarrow [0, 1]$, determines the probability distribution of which state follows the current state after an agent's action. The reward function r measures the immediate payoff after an action in a specific state, assessing action's effectiveness. Furthermore, Ω is the observation function which satisfy $\Omega(s, a) : S \times A \rightarrow O$ where O is the observation space. N represents the number of environmental agents. The discount factor γ determines the importance of future rewards over immediate ones, affecting the agents' long-term strategies. Each agent acts based on its own observations and the shared goal of maximizing cumulative sum of rewards $\sum_{t=0}^{T} \gamma^t r_t$. To express the collective observations and actions from all agents at time step t, we utilize bold notation $\mathbf{o}_t$ and $\mathbf{a}_t$.

In our setup, agents are trained with a fixed dataset $\mathcal{D} := \{\tau^1, \tau^2, ..., \tau^E\}$, where the trajectory τ is composed of sequential transitions $(\mathbf{o}_t, \mathbf{a}_t, r_t, done)$ where each refers to observation, action, and reward at timestep *t* from multiple agents and E is the number of trajectories. Generally, the policy is trained by maximizing the expectation of cumulative discounted global reward $\mathbb{E}\big[\sum_{t=0}^{T} \gamma^t r(\mathbf{o}_t, \mathbf{a}_t)\big]$ (Sutton & Barto, 2018). Challenges specific to offline MARL dataset include (1) handling data from a scarce dataset, which cannot cover all the state-action space that increases exponentially as the number of agents grows, (2) mitigating non-stationarity in the environment due to the evolving strategies of other agents, and (3) deriving insights from possibly sub-optimal and exploratory past actions of these agents.

2.2. Diffusion models

Diffusion models (Ho et al., 2020; Song et al., 2020) are generative models known for producing high-quality samples especially in vision domain. Recently, several works

have applied diffusion models to RL tasks such as policy learning (Wang et al., 2023; Kang et al., 2024; Ada et al., 2024; Chen et al., 2023; Li et al., 2023b), planning (Janner et al., 2022; Liang et al., 2023), and data synthesizing (Lu et al., 2024; He et al., 2024; Li, 2023; Ni et al., 2023) utilizing their capabilities to capture the data distribution and powerful expressiveness.

Diffusion models operate through two main phases: the forward noising process and the reverse denoising process. In the forward phase, data samples $\mathbf{x}_0$ from the original distribution $q(\mathbf{x}_0)$ are progressively transformed into Gaussian noise $\mathcal{N}(\mathbf{0}, \mathbf{I})$ over a series of diffusion steps k by adding i.i.d. Gaussian noise with a standard deviation σ. The transformation at each step according to a variance schedule $\beta_1, ..., \beta_K$ is given by:

$$q(\mathbf{x}_{1:K}|\mathbf{x}_0) := \prod_{k=1}^{K} q(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{1}$$

In the reverse phase, the model learns to reconstruct the original data from the noised starting point $p(\mathbf{x}_K) = \mathcal{N}(\mathbf{x}_K; \mathbf{0}, \mathbf{I})$. This process is modeled by:

$$p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k) := \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_k, k), \boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k)), \tag{2}$$

where $\boldsymbol{\mu}_\theta(\mathbf{x}_k, k)$ and $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k)$ are functions parameterized by neural networks, representing the mean and covariance of the Gaussian distribution at each reverse step. Finally, training is performed via optimizing the variational lower bound on negative log likelihood:

$$\mathbb{E}[\log p_\theta(\mathbf{x}_o)] \geq \mathbb{E}_q[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}] \tag{3}$$

In this paper, we denote trajectory τ_k as $\sqrt{\bar{\alpha}_k}\tau_0 + \sqrt{1-\bar{\alpha}_k}\epsilon$ where $\alpha_k := 1 - \beta_k$ and $\bar{\alpha}_k := \prod_{s=1}^{k} \alpha_s$, where τ_0 refers the trajectory which is not corrupted with gaussian noise. By leveraging diffusion models to planning domain, Zhu et al. (2023); Li et al. (2023a) could significantly improve their performance with MARL algorithms.

2.3. Data synthesis in RL

Conventional data augmentation methods in RL typically introduce minor perturbations to states to maintain consistency with the environment dynamics (Laskin et al., 2020; Sinha et al., 2022). However, diffusion models, commonly used in computer vision to create synthetic data, are well-suited for addressing data scarcity in the RL datasets and offer a more robust approach by learning the entire distribution from the actual dataset, $\mathcal{D}_{\text{real}}$. Lu et al. (2024) first parameterize and learn the data distribution $\rho_\theta(\tau)$ from $\mathcal{D}_{\text{real}}$, and then generate the desired synthetic data, represented as $\mathcal{D}_{\text{syn}} = \{\tau \sim \rho_\theta(\tau)\}$. The resulting composite dataset, $\mathcal{D} = \mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{syn}}$, is then utilized for policy learning.

In online settings, ongoing interaction between agent and environment facilitates a cyclical updating mechanism where both the diffusion model and the policy itself are iteratively refined. Such a dynamic process promotes continual enhancements in the policy's decision-making capabilities and the diffusion model's data generation accuracy. The ability to adapt and improve continuously is crucial for maintaining relevance and effectiveness in changing environments.

Data augmentation strategies for MARL are not as advanced as those developed for single-agent RL. Addressing this discrepancy is particularly essential given the complex and multifaceted dynamics inherent in MARL systems, where multiple agents interact within a shared space. These interactions can significantly complicate the training process, making the development of robust data augmentation techniques vital. In offline settings, where real-time data collection is not feasible, the scarcity of diverse and comprehensive datasets becomes a critical issue. Effective data augmentation for MARL is necessitated to simulate a wider range of scenarios and interactions, enhancing the model's ability to generalize from limited data.

3. Synthesizing Multi-Agent Offline Data

In the realm of synthesizing data for RL, methodologies can generally be categorized into two primary classes: (1) Purturbation on state space (Laskin et al., 2020; Sinha et al., 2022) and (2) Generative models (Lu et al., 2024; He et al., 2024). Our approach falls within the generative class, which focuses on creating new data points through models that learn the underlying distribution of the dataset. For that reason, generative models are more advantageous to highlight the diversity of the data rather than merely increasing the quantity by making the data similar.

One of the key challenges in MARL data augmentation is the complexity associated with handling data from multiple agents. The difficulties primarily arise from two main areas: (1) integration of multi-agent data which is often unclear due to the intricate relationships and dependencies between agents' actions and states; and (2) development of methods to enhance the cooperativeness of episodes, which is crucial for scenarios where agents need to work together towards a common goal. To address these challenges, we devise specific strategies as outlined below.

Integration of agents' features. We have established a method to efficiently merge attributes from numerous agents, guaranteeing that the data accurately represents the interactions between the agents. This entails employing sophisticated aggregation techniques that preserve the integrity and

context of every agent's actions and observations, enabling a comprehensive perspective of the multi-agent environment.

Enhancing episode cooperativeness. We focus on generating the episodes to foster more cooperative interactions among agents. This is achieved through the use of Q-total loss directly into the diffusion loss. By adpoting the cumulative sum of reward given all agents' current observation and action, generally denoted as $Q^{\text{tot}}(\mathbf{o}_t, \mathbf{a}_t)$ in RL domain, we promote the generative model to synthesize the episodes to be more cooperative direction, which is key to improving the overall effectiveness of the training episodes.

In the rest of this section, we present how we devised novel methods for synthesizing pre-existing MARL data, with the goal of improving the quality of offline datasets. Our method use a `Conv1D`-based diffusion model to create the fundamental elements of MARL episodes τ: observations $\mathbf{o}_t$, actions $\mathbf{a}_t$, rewards r_t, terminal states $done_t$, and total Q-values $Q^{\text{tot}}(\mathbf{o}_t, \mathbf{a}_t)$ (we will denote this as Q_t^{tot} for simplicity) which refers the sum of cumulative rewards given the current state and action until the episode ends, named as *rewards-to-go* in RL domain. We augment the episodes to be more cooperative by guiding with Q_t^{tot}.

We clarify our analytical approach and its implementation in section 3.1 and 3.2, illustrating how these strategies are used with MARL data. Furthermore, we present empirical evidence of substantial performance improvement when employing these enhanced datasets with offline MARL methods, as demonstrated in Table 1.

3.1. Training data preparation

In this section, we describe how we prepare training data, which involves reformatting the original data into a specific structure for diffusion model with `Conv1D`. The data structure is reformatted as the size of (**B**, **F**, **T**), serving as the input format for our model: **B**atch size refers to the number of episodes included in the given dataset $\mathcal{D}$, **F**eature length represents the combined length of the feature vectors for all agents, and **T**ime steps indicate the maximum duration of any episode within the dataset, $\mathcal{D}$. If the episode ends before the maximum time step in this dataset, then its feature values are padded with 0. Our dataset exhibits high multi-modality, consisting of diverse range of features from multiple agents that include both continuous and discrete values, as shown in Figure 2.

To address the discrete action space, we convert discrete actions into a one-hot encoding format, as depicted in Figure 2. In the last few dimensions of the feature space, we append several critical pieces of information:

- **Global reward** r_t^{global} given to the agents at time t.
- **Total state-action value** $Q^{\text{tot}}(\mathbf{o}_t, \mathbf{a}_t)$ evaluating the chosen actions' value given the states, particularly used for guiding generated episodes to be cooperative.
- **Episode termination state** indicating whether the episode terminates at the current step. We mark it with 1 if the episodes ends, or 0 otherwise.

Then, our episode data reformatted for training is shaped as:

$$\tau_0 := \begin{bmatrix} o_0^1 & o_1^1 & \cdots & o_T^1 \\ a_0^1 & a_1^1 & \cdots & a_T^1 \\ \vdots & \ddots & \ddots & \vdots \\ o_0^N & o_1^N & \cdots & o_T^N \\ a_0^N & a_1^N & \cdots & a_T^N \\ r_0^{global} & r_1^{global} & \cdots & r_T^{global} \\ Q_0^{\text{tot}} & Q_1^{\text{tot}} & \cdots & Q_T^{\text{tot}} \\ done_0 & done_1 & \cdots & done_T \end{bmatrix}, \tag{4}$$

where the superscript and subscript of observation o_t^n and action a_t^n denote the agent's index and the time step of the episode. The details on the each value is deployed in the Figure 2. This approach allows our model to perform observation learning, policy learning, reward learning, and even state-action value learning. We use only a diffusion model for these tasks, which significantly reduces the complexity of training and sampling processes. This streamlined approach ensures our model to efficiently learn the pre-collected trajectories distribution, while guiding its episodes synthesis in a cooperative manner. The detailed specifications for the dataset $\mathcal{D}$ is presented in Section 4.3.

3.2. Implicit Q^{tot} loss guidance augmentation

In training session, we train $Q^{\text{tot}}(\mathbf{o}_t, \mathbf{a}_t)$ whose learning target is sum of reward given the current state and action until the episode ends, named as *reward-to-go* in RL domain, which can be calculated given the dataset. The reason why we utilize the *reward to go* is that it indicates the unbiased estimator of the state-action value from the current timestep, defined by $Q(o_t, a_t) = \mathbb{E}[\Sigma_{t'=t}^{T} \gamma^{t'-t} r(o, a) | o = o_{t'}, a = a_{t'}]$. From this, we can implicitly train our diffusion model to make more cooperative scenarios by maximizing $Q_{f_\theta(\tau_k, k)}(\mathbf{o}, \mathbf{a})$, which can be represented as: $\arg\max_{f_\theta} Q_{f_\theta(\tau_k, k)}(\mathbf{o}, \mathbf{a})$, where f_θ is diffusion model and $Q_{f_\theta(\tau_k, k)}(\mathbf{o}, \mathbf{a})$ is regarded as the estimator of Q_t^{tot} generated from the diffusion model as shown in Figure 2 (b).

In this context, we optimize the diffusion model with the loss $\mathbb{E}_{k \in U[1,T], \tau_0 \in \mathcal{B} \sim \mathcal{D}}[\|\tau_0 - f_\theta(\tau_k, k)\|^2]$ to predict the start trajectory τ_0, not just like DDPM (Ho et al., 2020) which predict the noise ϵ with the loss $\mathbb{E}_{k \in U[1,T], \tau_0 \in \mathcal{B} \sim \mathcal{D}}[\|\epsilon - \epsilon_\theta(\tau_k, k)\|^2]$. These two formulations are interchangeable and are both widely used in diffusion models (Ramesh et al.,

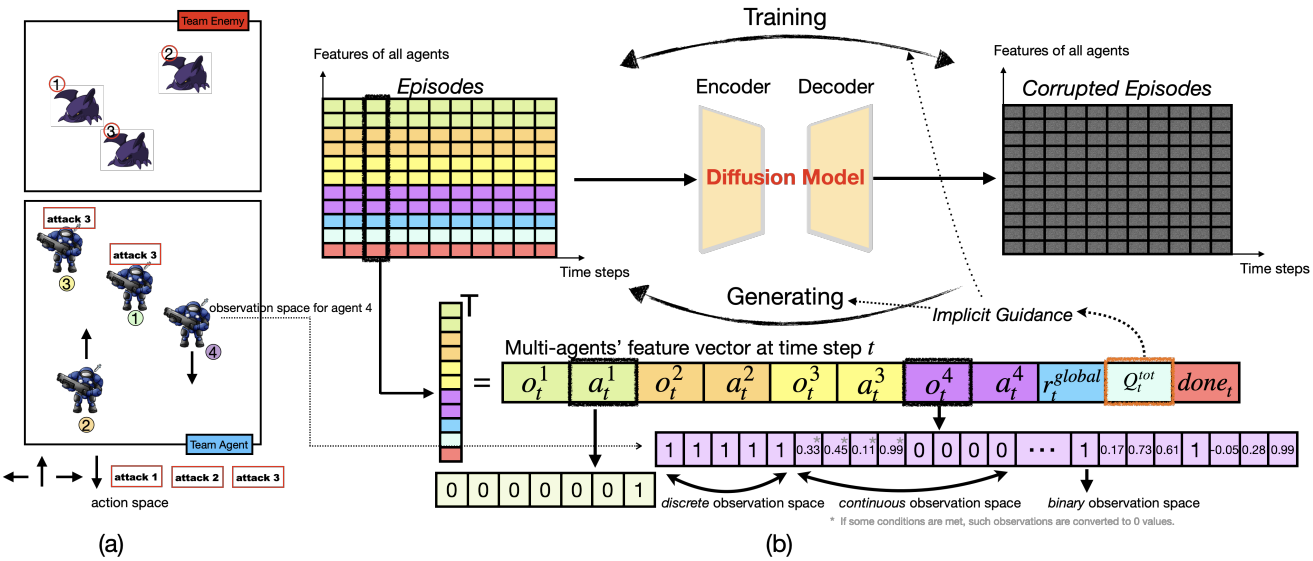

Figure 2. Our proposed algorithm architecture for MARL episodes augmentation. (a) represents the example of MARL offline dataset based on SMAC (Samvelyan et al., 2019) environments at time step = t. There are two teams that have to kill each other to win the battle. Here, the actions are in the discrete space such as digit which is converted into one-hot encoding format. (b) represents (1) how our training datasets are prepared and look like, (2) how **EAQ** is trained under the guidance of Q_t^{tot} and (3) how highly the values in the datasets is multi-modal.

2022). Our diffusion loss function to predicting the start trajectory is for utilizing the Q_t^{tot} as target values with the estimator of $Q_{f_\theta(\tau_k,k)}(\mathbf{o},\mathbf{a})$ in the training phase, simultaneously maximizing the estimator. If we predict the noise ϵ_θ, then we cannot optimize the $Q_{f_\theta(\tau_k,k)}(\mathbf{o},\mathbf{a})$ to be maximized in the training steps otherwise, it will require additional model to predict the Q^{tot}. Then, the final loss function with diffusion model is given by:

$$\begin{aligned}\mathcal{L}(\theta) &= \mathcal{L}_{\text{diffusion}} + \lambda \mathcal{L}_{Q_{\text{tot}}} \\ &= \mathbb{E}_{k\in U[1,K],\tau_0\in\mathcal{B}\sim\mathcal{D}}[\|\tau_0 - f_\theta(\tau_k,k)\|^2 \\ &\quad + \lambda\left[\max_{\tau_0\in\mathcal{B}} Q_{\tau_0}(\mathbf{o},\mathbf{a}) - Q_{f_\theta(\tau_k,k)}(\mathbf{o},\mathbf{a})\right]_{\geq 0}]\end{aligned} \tag{5}$$

where $\max_{\tau_0\in\mathcal{B}} Q_{\tau_0}(\mathbf{o},\mathbf{a})$ is the maximum Q-value empirically obtained from the minibatch $\mathcal{B}$. λ is a balancing hyperparameter, and $Q_{\tau_0}(\mathbf{o},\mathbf{a})$ is an expectation of Q_t^{tot} in an episodes represented as $\frac{1}{T}\Sigma_{t=0}^{t=T} Q_t^{\text{tot}}$. We define the upper bound of $Q_{f_\theta(\tau_k,k)}(\mathbf{o},\mathbf{a})$ to be $\max_{\tau_0\in\mathcal{B}} Q_{\tau_0}(\mathbf{o},\mathbf{a})$ in the minibatch $\mathcal{B}$ for the stable training, otherwise the value generated by $Q_{f_\theta(\tau_k,k)}(\mathbf{o},\mathbf{a})$ can explode to the infinity value by the reason that we maximize the $Q_{f_\theta(\tau_k,k)}(\mathbf{o},\mathbf{a})$. The logic to train **EAQ** is explained in Algorithm 1.

3.3. Visualization of augmented dataset

We wonder if **EAQ** is indeed beneficial to augment the data in the perspective of variability and cooperativeness. Thus, we raise a question: *"Does **EAQ**-generated dataset encompass a broader range of states than the original dataset?"* The cooperativeness of augmented data by **EAQ** has been demonstrated in Section 1 with Figure 1. To validate the variability of the augmented data, we map the observations of the agents into 2-D space using t-SNE (van der Maaten & Hinton, 2008) as shown in Figure 3. We find that **EAQ** not only follows the true data distribution but also covers broader range of observations, which shows the fidelity and variability of **EAQ**.

Algorithm 1 EAQ

Given: dataset $episodes_N \in \mathcal{D}_{\text{real}}$
Set: hyperparameter λ
Calculate the *reward-to-go* in the episodes.
Transform $\mathcal{D}$ into (B, F, T) shaped tensor.
/ Training Process */*
repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $k \sim \text{Uniform}(\{1, ..., K\})$
  Compute the loss
  $\mathcal{L}_{\text{diffusion}} = \|\tau_0 - f_\theta(\tau_k,k)\|^2$
  $\mathcal{L}_{Q_{\text{tot}}} = \left[\max_{\tau_0\in\mathcal{B}} Q_{\tau_0}(\mathbf{o},\mathbf{a}) - Q_{f_\theta(\tau_k,k)}(\mathbf{o},\mathbf{a})\right]_{\geq 0}$
  $\mathcal{L}(\theta) = \mathbb{E}_{k\in U[1,K],\tau_0\in\mathcal{B}\sim\mathcal{D}}[\mathcal{L}_{\text{diffusion}} + \lambda\mathcal{L}_{Q_{\text{tot}}}]$
until converged
/ Sampling Process */*
Set: Upsampling scale S
Generate $episodes_{N\times S} \in \mathcal{D}_{\text{syn}}$
$\mathcal{D}_{\text{aug}} = \mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{syn}}$

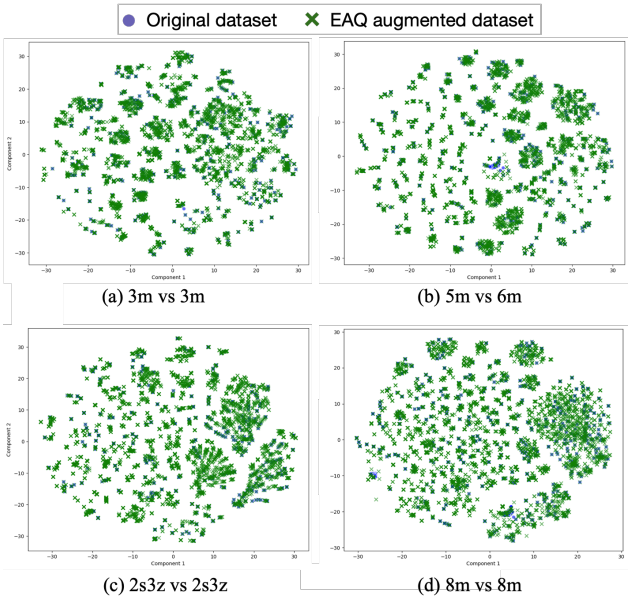

Figure 3. We visualize a t-SNE (van der Maaten & Hinton, 2008) projection on the observation space of original dataset and **EAQ** augmented dataset on several scenarios in SMAC environment.

4. Experiments

We validate **EAQ** practically with other RL data augmentation methods based on the offline MARL algorithms. For the experiments, we consider two variants of offline MARL algorithms adopted to QMIX (Rashid et al., 2018) with representatives of the offline RL baseline: QMIX-CQL (Kumar et al., 2020) and QMIX-BCQ (Fujimoto et al., 2019). We upsample the dataset five times more than the original size and aggregate it as $\mathcal{D}_{\text{aug}} = \mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{syn}}$.

4.1. Baseline RL algorithms

As aforementioned, we utilize the combined version of QMIX with CQL and BCQ algorithms implemented by Formanek et al. (2023). We briefly explain about each algorithm below.

QMIX (Rashid et al., 2018) addresses the challenge of learning decentralized policies in a centralized end-to-end fashion, leveraging global state information during training. It employs a mixing network that estimates the joint action-value function Q_t^{tot} as a monotonic combination of per-agent utility functions Q_t^{agent}.

CQL (Kumar et al., 2020) aims to penalize the Q-values of out-of-distribution actions in the offline data, ensuring that the Q-function underestimates rather than overestimates. It learns a Q-function network by minimizing the standard Bellman error, with an additional regularization term that penalizes Q-values for actions with low probability under the data distribution from the offline dataset.

BCQ (Fujimoto et al., 2019) is an offline reinforcement learning algorithm that mitigates distributional shift by constraining the policy to the behavior distribution. It starts by training a generative model, such as a VAE, on the offline dataset to capture the action distribution of the behavior policy conditioned on the states. For a given observation o_t, BCQ samples multiple actions from the generative model's, denoted as G_θ, output distribution $G_\theta(\cdot|o_t)$. It then selects the action a_t^* that maximizes the Q-value $Q(o_t, a_t)$ while remaining close to the behavior distribution $G_\theta(a_t|o_t)$.

4.2. Compared augmentation algorithms

Augmentation in RL remains largely unexplored, particularly in proprioceptive observation-based single-agent RL, and even more so in the multi-agent RL domain. For the purposes of comparison with existing augmentation methods, we utilize RAD (Laskin et al., 2020) as our benchmark model. Furthermore, to assess the effectiveness of the Q^{tot} optimization, we evaluate our algorithm without Q-loss optimization, which we will refer to as **EAQ**$^{-Q}$ for simplicity.

RAD-s and RAD-m (Rashid et al., 2018) RAD is a data augmentation method that multiplies the uniform random variable $z \sim \mathcal{U}[\alpha, \beta]$ to the given state s, *i.e.*, $s' = s * z$. Here, we set α and β to be 0.8 and 1.2 respectively. If the random variable z is single-variate variable, we denote it as RAD-s, otherwise (multi-variate variable), we denote it as RAD-m.

4.3. Datasets and environments

We select the simulator StarCraft Multi-Agent Challenges (SMAC) (Samvelyan et al., 2019) for evaluation of our proposed augmentation methods which is the most popular environment for the reason of complexity in cooperativeness and environments and the pre-collected dataset is from the open-source website (Formanek et al., 2023). For the assumption that the pre-collected dataset is small, we deliberately downsample the original dataset to be 3% amount of the pre-collected dataset. We denote the downsampled dataset as *original* dataset.

SMAC consists of allies and enemies, where each team must kill others to win the battle. Allies receive the episode's total reward of 20 when they win a battle, as well as small rewards of 0.05 for killing an enemy and a payout equal to the amount of damage they dealt to adversaries. To win a battle, agents must cooperate among themselves to manage their group behavior, like *focusing fire* while not overkilling the enemies, or *kiting* to lure the enemies and kill them one by one. We report the results of scenarios *3m vs 3m, 5m vs 6m, 8m vs 8m, 2s3z vs 2s3z and 3s5z vs 3s6z* with sup-optimal policies with return values in an episode.

Table 1. Returns on each SMAC scenario using CQL and BCQ-based algorithms, reported with mean and standard deviation across 5 runs for 15,000 training iterations. Each scenario consists of {*team*} - {*quality*}, where m, s, and z in *team* refer to marine, stalker, and zealot, respectively. The original data *quality* has medium and poor policy. The best performance is marked with **bold**.

	QMIX-CQL					QMIX-BCQ				
Scenario	original	RAD-s	RAD-m	**EAQ**$^{-Q}$	**EAQ** (ours)	original	RAD-s	RAD-m	**EAQ**$^{-Q}$	**EAQ** (ours)
3m-medium	7.47 ± 6.35	**16.90** ± 2.75	15.05 ± 3.11	8.22 ± 5.77	13.70 ± 7.43	5.17 ± 2.76	2.56 ± 1.45	4.56 ± 2.00	4.16 ± 3.27	**16.90** ± 2.21
3m-poor	3.81 ± 1.59	2.83 ± 0.85	7.87 ± 6.36	6.95 ± 3.46	**10.60** ± 6.25	0.79 ± 0.83	1.35 ± 1.31	0.77 ± 0.32	1.38 ± 0.97	**1.59** ± 1.59
5m vs 6m-medium	0.29 ± 0.45	0.05 ± 0.07	0.18 ± 0.31	0.77 ± 0.42	**3.75** ± 2.04	3.38 ± 0.43	3.00 ± 0.81	**3.52** ± 0.52	2.62 ± 1.87	1.72 ± 1.66
5m vs 6m-poor	0.99 ± 1.21	1.84 ± 1.83	1.73 ± 1.61	**6.48** ± 1.39	3.64 ± 2.24	2.01 ± 1.05	1.96 ± 1.27	2.25 ± 1.38	4.00 ± 1.53	**4.23** ± 0.70
8m-medium	1.87 ± 0.86	1.24 ± 0.67	1.22 ± 0.70	2.95 ± 1.17	**3.59** ± 0.60	1.64 ± 1.10	2.32 ± 0.75	1.60 ± 1.83	4.91 ± 1.34	**5.92** ± 5.58
8m-poor	0.35 ± 0.25	0.66 ± 0.62	0.88 ± 0.92	1.41 ± 0.93	**1.69** ± 1.69	2.56 ± 0.48	2.75 ± 0.42	2.44 ± 0.41	**9.23** ± 2.61	6.06 ± 1.79
2s3z-medium	3.77 ± 0.86	5.98 ± 2.92	5.26 ± 1.48	**9.74** ± 2.15	9.05 ± 1.80	6.86 ± 0.84	5.41 ± 1.78	6.57 ± 1.49	5.65 ± 1.68	**7.46** ± 2.61
2s3z-poor	5.16 ± 2.71	5.13 ± 1.68	3.30 ± 1.43	**6.13** ± 1.26	5.40 ± 3.37	5.20 ± 1.89	5.39 ± 1.33	4.72 ± 0.87	6.61 ± 1.38	**7.34** ± 3.90
3s5z vs 3s6z-medium	2.69 ± 1.49	2.81 ± 1.00	3.34 ± 1.37	6.14 ± 2.60	**7.17** ± 1.37	6.14 ± 1.86	5.37 ± 1.21	5.47 ± 0.79	6.07 ± 1.22	**6.58** ± 2.90
3s5z vs 3s6z-poor	2.87 ± 0.80	2.69 ± 0.42	2.00 ± 0.88	5.16 ± 1.19	**5.35** ± 0.74	4.69 ± 1.32	5.48 ± 1.23	5.75 ± 0.40	5.51 ± 0.37	**6.37** ± 0.46
AVG-medium	3.22	5.39	5.01	5.56	**7.45**	4.64	3.73	4.34	4.68	**7.72**
AVG-poor	2.64	2.63	3.16	5.23	**5.34**	3.05	3.38	3.19	**5.35**	5.12

4.4. Diffusion model for EAQ

In our experiments, we selected the Denoising Diffusion Probabilistic Model (DDPM) detailed by Ho et al. (2020), which incorporates a `Conv1D`-layer, to implement **EAQ** approach. We adhere to the established DDPM hyperparameters for training and sampling, ensuring consistency across our processes.

The choice of DDPM is particularly suitable for our purposes due to its robustness and proven effectiveness in generating high-quality samples. However, it is important to note that **EAQ** is versatile and not exclusively limited to DDPM or its variant DDIM. Our method is compatible with any diffusion model that can integrate a `Conv1D`-layer. This flexibility is advantageous because it allows us the possibility to experiment with different types of diffusion models, potentially enhancing our ability to fine-tune the algorithms according to specific requirements.

4.5. Results

In our comprehensive experiments, we assess the performance of two variants of RL algorithms across a spectrum of datasets that have been enhanced by different augmentation methods. We report **EAQ** augmented dataset performance tuned on the hyperparameter λ among [0.5, 0.1, 0.01]. Our evaluation spans 10 pre-collected datasets, utilized in conjunction with 2 distinct RL algorithms, resulting in a total of 20 unique scenarios being tested. Among these, the **EAQ** method demonstrates best performance in 14 out of the 20 tasks, marking a significant improvement in efficacy. Additionally, **EAQ**$^{-Q}$, which does not utilize the maximization of the Q-function, achieves the best performance in 4 tasks, illustrating the impact of sophisticated augmentation techniques even without direct Q-function optimization. The RAD-s method excelled only in the *3m vs 3m* scenario.

Overall, the results presented in Table 1 underscore the significant advantages offered by our **EAQ** method. Particularly, **EAQ** has outperformed the baseline dataset by a substantial margin—17.3% for datasets crafted under a medium policy datasets, and 12.9% for those under a poor policy datasets, where the margin is calculated based on the normalized returns (normalized by maximum return 20). These results not only emphasize the effectiveness of **EAQ** combined with original datasets but also highlight its superiority over prior augmentation methods.

Furthermore, it is intriguing to note that employing **EAQ**$^{-Q}$ also results in an enhanced performance compared to both the original dataset and other state-based augmented datasets such as RAD-s and RAD-m. This observation is critical as it suggests that even without explicit optimization of the Q-values, the inherent capabilities of diffusion models can significantly elevate the quality of data augmentation, thereby boosting algorithm performance across various tasks. This is because the diffusion model generates new samples with high fidelity by accurately learning the true data distribution.

Dataset where EAQ does not offer benefits. In our examination, we observed that when utilizing datasets characterized by poor behavioral policies, our algorithm **EAQ** exhibits performance levels that are similar to those achieved by the **EAQ**$^{-Q}$ method. This is in stark contrast to the results obtained from datasets with medium behavioral policies, where **EAQ** distinctly outperforms other approaches. This discrepancy in performance can be attributed to the inherent quality of the datasets being used.

EAQ is designed to maximize the function $Q_{f_\theta(\tau_k,k)}(\mathbf{o},\mathbf{a})$ during each epoch, with the upper bound of the maximum Q-value observed in the minibatch $\max_{\tau_0\in\mathcal{B}} Q_{\tau_0}(\mathbf{o},\mathbf{a})$. This strategy is predicated on the assumption that there is potential for improvement in the dataset's quality through

the application of our algorithm. However, if the initial quality of the dataset is exceedingly poor, the ceiling for enhancement is significantly lowered. Under such conditions, even with a considerable increase in the number of generated episodes, the improvement in dataset quality remains marginal compared to $\texttt{EAQ}^{-Q}$.

5. Conclusion and Future Work

In this study, we use a diffusion model as an episodes augmentation module for offline MARL that mitigates the data scarcity problem. Specifically, we guide the diffusion model by the implicit Q-loss to generate multi-agent's cooperative actions, which is found to be effectively deploying a practical, and cooperativeness-oriented model.

However, there is still room for improvement in **EAQ**. First, we need to validate our algorithms across a broader range of observation and action spaces, such as those involving continuous actions. Second, we must explore ways to enhance dataset quality, especially when dealing with extremely poor behavioral policies. Nevertheless, our approach will work with any generative models, which we believe could be a potential future work by applying to the next generation models like Flow Matching.

Acknowledgements

This work was supported by Center for Applied Research in Artificial Intelligence (CARAI) grant funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD) (UD230017TD).

References

Ada, S. E., Oztop, E., and Ugur, E. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4): 3116–3123, April 2024. ISSN 2377-3774. doi: 10.1109/lra.2024.3363530. URL http://dx.doi.org/10.1109/LRA.2024.3363530.

Chen, H., Lu, C., Ying, C., Su, H., and Zhu, J. Offline reinforcement learning via high-fidelity generative behavior modeling, 2023.

Chen, Y., Li, H., and Zhao, D. Boosting continuous control with consistency policy, 2024.

Formanek, C., Jeewa, A., Shock, J., and Pretorius, A. Off-the-grid marl: Datasets with baselines for offline multi-agent reinforcement learning, 2023.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration, 2019.

He, H., Bai, C., Xu, K., Yang, Z., Zhang, W., Wang, D., Zhao, B., and Li, X. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36, 2024.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

Kang, B., Ma, X., Du, C., Pang, T., and Yan, S. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning, 2020.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Li, W. Efficient planning with latent diffusion, 2023.

Li, Z., Pan, L., and Huang, L. Beyond conservatism: Diffusion policies in offline multi-agent reinforcement learning. *arXiv preprint arXiv:2307.01472*, 2023a.

Li, Z., Pan, L., and Huang, L. Beyond conservatism: Diffusion policies in offline multi-agent reinforcement learning, 2023b.

Liang, Z., Mu, Y., Ding, M., Ni, F., Tomizuka, M., and Luo, P. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*, 2023.

Lu, C., Ball, P., Teh, Y. W., and Parker-Holder, J. Synthetic experience replay. *Advances in Neural Information Processing Systems*, 36, 2024.

Ni, F., Hao, J., Mu, Y., Yuan, Y., Zheng, Y., Wang, B., and Liang, Z. Metadiffuser: Diffusion model as conditional planner for offline meta-rl, 2023.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents, 2022.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, 2018.

Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge, 2019.

Sinha, S., Mandlekar, A., and Garg, A. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pp. 907–917. PMLR, 2022.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning, 2023.

Zhu, Z., Liu, M., Mao, L., Kang, B., Xu, M., Yu, Y., Ermon, S., and Zhang, W. Madiff: Offline multi-agent learning with diffusion models. *arXiv preprint arXiv:2305.17330*, 2023.