MADIFF: Offline Multi-agent Learning with Diffusion Models

Zhengbang Zhu[1] **Minghuan Liu**[1] **Liyuan Mao**[1] **Bingyi Kang**[2] **Minkai Xu**[3]
Yong Yu[1] **Stefano Ermon**[3] **Weinan Zhang**[1][†]
[1] Shanghai Jiao Tong University, [2] ByteDance, [3] Stanford University
{zhengbangzhu, minghuanliu, maoliyuan, yyu, wnzhang}@sjtu.edu.cn,
bingykang@gmail.com, {minkai, ermon}@cs.stanford.edu

Abstract

Offline reinforcement learning (RL) aims to learn policies from pre-existing datasets without further interactions, making it a challenging task. Q-learning algorithms struggle with extrapolation errors in offline settings, while supervised learning methods are constrained by model expressiveness. Recently, diffusion models (DMs) have shown promise in overcoming these limitations in single-agent learning, but their application in multi-agent scenarios remains unclear. Generating trajectories for each agent with independent DMs may impede coordination, while concatenating all agents' information can lead to low sample efficiency. Accordingly, we propose MADIFF, which is realized with an attention-based diffusion model to model the complex coordination among behaviors of multiple agents. To our knowledge, MADIFF is the first diffusion-based multi-agent learning framework, functioning as both a decentralized policy and a centralized controller. During decentralized executions, MADIFF simultaneously performs teammate modeling, and the centralized controller can also be applied in multi-agent trajectory predictions. Our experiments demonstrate that MADIFF outperforms baseline algorithms across various multi-agent learning tasks, highlighting its effectiveness in modeling complex multi-agent interactions.

1 Introduction

Offline reinforcement learning (RL) [Fujimoto et al., 2019, Kumar et al., 2020] learns exclusively from static datasets without online interactions, enabling the effective use of pre-collected large-scale data. However, applying temporal difference (TD) learning in offline settings causes extrapolation errors [Fujimoto et al., 2019], where target value functions are evaluated on out-of-distribution actions. Sequence modeling algorithms bypass TD-learning by directly fitting the dataset distribution [Chen et al., 2021, Janner et al., 2021]. Nevertheless, these methods are limited by the model's expressiveness, making it difficult to handle diverse datasets. They also suffer from compounding errors [Xiao et al., 2019] due to autoregressive generation. Recently, diffusion models (DMs) have achieved remarkable success in various generative modeling tasks [Song and Ermon, 2019, Ho et al., 2020, Xu et al., 2022], owing to their exceptional abilities at capturing complex, high-dimensional data distributions. Their successes have also been introduced into offline RL, offering a superior modeling choice for sequence modeling algorithms [Janner et al., 2022, Ajay et al., 2023].

Compared to single-agent learning, offline multi-agent learning (MAL) has been less studied and is more challenging. Since the behaviors of all agents are interrelated, each agent is required to model interactions and coordination among agents, while making decisions in a decentralized manner to achieve the goal. Current MAL approaches typically train a centralized value function to update

---

[†]Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

individual agents' policies [Rashid et al., 2020] or use an autoregressive transformer to determine each agent's actions [Meng et al., 2021, Wen et al., 2022]. However, without online interactions, an incorrect centralized value can lead to significant extrapolation errors, and the transformer can only serve as an independent model for each agent.

In this paper, we aim to study the potential of employing DMs to solve the above challenges in offline MAL problems. Merely adopting existing diffusion RL methods by using independent DMs to model each agent can result in serious inconsistencies due to a lack of proper credit assignment among agents. Another possible solution is to concatenate all agents' information as the input and output of the DM. However, treating the agents as a single unified agent neglects the important nature of multi-agent systems. One agent may have strong correlations with only a few other agents, which makes a full feature interaction redundant. In many multi-agent systems, agents exhibit certain symmetry and can share model parameters for efficient learning [Arel et al., 2010]. However, concatenating them in a fixed order breaks this symmetry, forcing the model to treat each agent differently.

To address the aforementioned coordination challenges, we propose the first centralized-training-decentralized-execution (CTDE) diffusion framework for MA problems, named MADIFF. MADIFF adopts a novel attention-based DM to learn a return-conditional trajectory generation model on a reward-labeled multi-agent interaction dataset. In particular, the designed attention is computed in several latent layers of the model of each agent to fully interchange the information and integrate the global information of all agents. To model the coordination among agents, MADIFF applies the attention mechanism on latent embedding for information interaction across agents. The attention mechanism enables the dynamic modeling of agent interactions through learned weights, while also enabling the use of a shared backbone to model each agent's trajectory, significantly reducing the number of parameters. During training, MADIFF performs centralized training on the joint trajectory distributions of all agents from offline datasets, including different levels of expected returns. During inference, MADIFF adopts classifier-free guidance with low-temperature sampling to generate behaviors given the conditioned high expected returns, allowing for decentralized execution by predicting the behavior of other agents and generating its own behavior. Therefore, MADIFF can be regarded as a principled offline MAL solution that not only serves as a decentralized policy for each agent or a centralized controller for all agents, but also includes teammate modeling without additional cost. Comprehensive experiments demonstrated superior performances of MADIFF on various multi-agent learning tasks, including offline MARL and trajectory prediction.

In summary, our contributions are (1) the first diffusion-based multi-agent learning framework that unifies decentralized policy, centralized controller, teammate modeling, and trajectory prediction; (2) a novel attention-based DM structure that is designed explicitly for MAL and enables coordination among agents in each denoising step; (3) achieving superior performances for various offline multi-agent problems.

2 Preliminaries

2.1 Multi-agent Offline Reinforcement Learning

We consider a partially observable and fully cooperative multi-agent learning (MAL) problem, where agents with local observations cooperate to finish the task. Formally, it is defined as a Dec-POMDP [Oliehoek and Amato, 2016]: $G = \langle \mathcal{S}, \mathcal{A}, P, r, \Omega, O, N, U, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote state and action space separately, and γ is the discounted factor. The system includes N agents $\{1, 2, \ldots, N\}$ act in discrete time steps, and starts with an initial global state $s_0 \in \mathcal{S}$ sampled from the distribution U. At each time step t, every agent i only observes a local observation $o^i \in \Omega$ produced by the function $O(s, a) : \mathcal{S} \times \mathcal{A} \to \Omega$ and decides $a \in \mathcal{A}$, which forms the joint action $\mathbf{a} \in \boldsymbol{\mathcal{A}} \equiv \mathcal{A}^N$, leading the system transits to the next state s' according to the dynamics function $P(s'|s, \mathbf{a}) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathcal{S}$. Normally, agents receive a shared reward $r(s, \mathbf{a})$ at each step, and the optimization objective is to learn a policy π^i for each agent that maximizes the discounted cumulative reward $\mathbb{E}_{s_t, \mathbf{a}_t}[\sum_t \gamma^t r(s_t, \mathbf{a}_t)]$. In offline settings, instead of collecting online data in environments, we only have access to a static dataset $\mathcal{D}$ to learn the policies. The dataset $\mathcal{D}$ is generally composed of trajectories $\boldsymbol{\tau}$, i.e., observation-action sequences $[\boldsymbol{o}_0, \boldsymbol{a}_0, \boldsymbol{o}_1, \boldsymbol{a}_1, \cdots, \boldsymbol{o}_T, \boldsymbol{a}_T]$ or observation sequences $[\boldsymbol{o}_0, \boldsymbol{o}_1, \cdots, \boldsymbol{o}_T]$. We use bold symbols to denote the joint vectors of all agents.

2

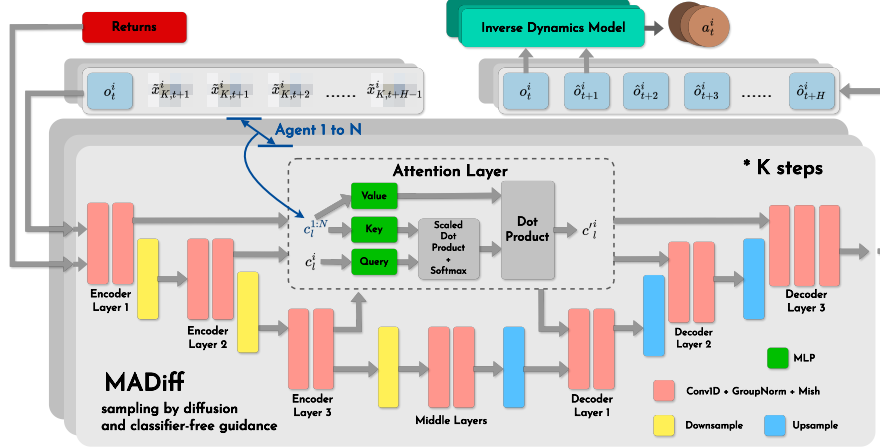



Figure 1: The architecture of MADIFF, which is an attention-based diffusion network framework that performs attention across all agents at every decoder layer of each agent.

2.2 Diffusion Probabilistic Models

Diffusion models (DMs) [Sohl-Dickstein et al., 2015, Song and Ermon, 2019, Ho et al., 2020], as a powerful class of generative models, implement the data generation process as reversing a forward noising process (denoising process). For each data point $x_0 \sim p_{\text{data}}(x)$ from the dataset $\mathcal{D}$, the noising process is a discrete Markov chain $x_{0:K}$ such that $p(x_k|x_{k-1}) = \mathcal{N}(x_k|\sqrt{\alpha_k}x_{k-1}, (1-\alpha_k)I)$, where $\mathcal{N}(\mu, \Sigma)$ denotes a Gaussian distribution with mean μ and variance Σ, and $\alpha_{0:K} \in \mathbb{R}$ are hyperparameters which control the variance schedule. The variational reverse Markov chain is parameterized with $q_\theta(x_{k-1}|x_k) = \mathcal{N}(x_{k-1}|\mu_\theta(x_k, k), (1-\alpha_k)I)$. The data sampling process begins by sampling an initial noise $x_K \sim \mathcal{N}(0, I)$, and follows the reverse process until x_0. The reverse process can be estimated by optimizing a simplified surrogate loss as in Ho et al. [2020]:

$$\mathcal{L}(\theta) = \mathbb{E}_{k \sim [1,K], x_0 \sim q, \epsilon \sim \mathcal{N}(0,I)} \left[\left\| \epsilon - \epsilon_\theta \left(x_k, k \right) \right\|^2 \right] . \tag{1}$$

The estimated Gaussian mean can be written as $\mu_\theta(x_k, k) = \frac{1}{\sqrt{\alpha_k}} \left(x_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}} \epsilon_\theta(x_k, k) \right)$, where $\bar{\alpha}_k = \Pi_{s=1}^k \alpha_s$.

2.3 Diffusing Decision Making

Diffusing over state trajectories and acting with inverse dynamics model. Among existing works in single-agent learning, Janner et al. [2022] chose to diffuse over state-action sequences, so that the generated actions for the current step can be directly used for executing. Another choice is diffusing over state trajectories only [Ajay et al., 2023], which is claimed to be easier to model and can obtain better performance due to the less smooth nature of action sequences:

$$\hat{\tau} := [s_t, \hat{s}_{t+1}, \cdots, \hat{s}_{t+H-1}], \tag{2}$$

where t is the sampled time step and H denotes the trajectory length (horizon) modeled by DMs. But the generated state sequences can not provide actions to be executed during online evaluation. Therefore, an inverse dynamics model is trained to predict the action $\hat{a}_t$ that makes the state transit from s_t to the generated next state $\hat{s}_{t+1}$:

$$\hat{a}_t = I_\phi(s_t, \hat{s}_{t+1}) . \tag{3}$$

Therefore, at every environment step t, the agent first plans the state trajectories using an offline-trained DM, and infers the action with the inverse dynamics model.

Classifier-free guided generation. For targeted behavior synthesis, DMs should be able to generate future trajectories by conditioning the diffusion process on an observed state s_t and information y. We use classifier-free guidance [Ho and Salimans, 2022] which requires taking $y(\tau)$ as additional inputs for the diffusion model. Formally, the sampling procedure starts with Gaussian noise $\hat{\tau}_K \sim \mathcal{N}(0, \alpha I)$, and diffuse $\hat{\tau}_k$ into $\hat{\tau}_{k-1}$ at each diffusion step k. Here $\alpha \in [0, 1)$ is the scaling factor used in

3

low-temperature sampling to scale down the variance of initial samples [Ajay et al., 2023]. We use $\tilde{x}_{k,t}$ to denote the denoised state s_t at k's diffusion step. $\hat{\tau}_k$ denotes the denoised trajectory at k's diffusion step for a single agent: $\hat{\tau}_k := [s_t, \tilde{x}_{k,t+1}, \cdots, \tilde{x}_{k,t+H-1}]$. Note that for sampling during evaluations, the first state of the trajectory is always set to the current observed state at all diffusion steps for conditioning, and every diffusion step proceeds with the perturbed noise:

$$\hat{\epsilon} := \epsilon_\theta(\hat{\tau}_k, \emptyset, k) + \omega(\epsilon_\theta(\hat{\tau}_k, y(\tau), k) - \epsilon_\theta(\hat{\tau}_k, \emptyset, k)) \,, \tag{4}$$

where ω is a scalar for extracting the distinct portions of data with characteristic $y(\tau)$. By iterative diffusing the noisy samples, we can obtain a clean state trajectory: $\hat{\tau}_0(\tau) := [s_t, \hat{s}_{t+1}, \cdots, \hat{s}_{t+H-1}]$.

3 Methodology

We formulate the problem of MAL as conditional generative modeling:

$$\max_\theta \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}}[\log p_\theta(\boldsymbol{\tau}|\boldsymbol{y}(\cdot))] \,, \tag{5}$$

where p_θ is learned for estimating the conditional data distribution of joint trajectory $\boldsymbol{\tau}$, given information $\boldsymbol{y}(\cdot)$, such as observations, rewards, and constraints. When all agents are managed by a centralized controller, *i.e.*, the decisions of all agents are made jointly, we can learn the generative model by conditioning the global information aggregated from all agents $\boldsymbol{y}(\boldsymbol{\tau})$; otherwise, if we consider each agent i separately and require each agent to make decisions in a decentralized manner, we can only utilize the local information $y^i(\tau^i)$ of each agent i, including the private information and the common information shared by all (*e.g.*, team rewards).

3.1 Multi-Agent Diffusion with Attention

In order to handle MAL problems, agents must learn to coordinate. To solve the challenge of modeling the complex inter-agent coordination in the dataset, we propose a novel attention-based diffusion architecture designed to interchange information among agents.

In Figure 1, we illustrate the architecture of MADIFF model. In detail, we adopt U-Net as the base structure for modeling agents' individual trajectories, which consists of repeated one-dimensional convolutional residual blocks. The convolution is performed over the time step dimension, and the observation feature dimension is treated as the channel dimension. To encourage information interchange and improve coordination ability, a critical change is made by adopting attention [Vaswani et al., 2017] layers before all decoder blocks in the U-Nets of all agents. Since embedding vectors from different agents are aggregated by the attention operation rather than concatenations, MADIFF is index-free such that the input order of agents can be arbitrary and does not affect the results.

Formally, the input to l-th decoder layer in the U-Net of each agent i is composed of two components: the skip-connected feature c_l^i from the symmetric l-th encoder layer and the embedding e_l^i from the previous decoder layer. The computation of attention in MADIFF is conducted on c_l^i rather than e_l^i since in the U-Net structure the encoder layers are supposed to extract informative features from the input data. We use $c_l'^i$ to denote the skip-connected feature after attention operations which aggregate information across agents. We adopt the multi-head attention mechanism to fuse the encoded feature $c_l'^i$ with other agents' information, which is important in effective multi-agent coordination.

3.2 Centralized Training Objectives

Given a multi-agent offline dataset $\mathcal{D}$, we train MADIFF which is parameterized through the unified noise model ϵ_θ for all agents and the inverse dynamics model I_ϕ^i of each agent i with the reverse diffusion loss and the inverse dynamics loss:

$$\begin{aligned} \mathcal{L}(\theta, \phi) := &\sum_i \mathbb{E}_{(o^i, a^i, o'^i) \in \mathcal{D}}[\|a^i - I_\phi^i(o^i, o'^i)\|^2] \\ &+ \mathbb{E}_{k, \boldsymbol{\tau}_0 \in \mathcal{D}, \beta}[\|\epsilon - \epsilon_\theta(\hat{\boldsymbol{\tau}}_k, (1-\beta)\boldsymbol{y}(\boldsymbol{\tau}_0) + \beta\emptyset, k)\|^2] \,, \end{aligned} \tag{6}$$

where β is sampled from a Bernoulli distribution to balance the training effort on unconditioned and conditioned models. For training the DM, we sample noise $\epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and a time step

4

$k \sim \mathcal{U}\{1, \cdots, K\}$, construct a noise corrupted joint state sequence $\boldsymbol{\tau}_k$ from $\boldsymbol{\tau}$ and predict the noise $\hat{\epsilon}_\theta := \epsilon_\theta(\hat{\boldsymbol{\tau}}_k, \boldsymbol{y}(\boldsymbol{\tau}_0), k)$. Note that the noisy array $\hat{\boldsymbol{\tau}}_k$ is applied with the same condition required by the sampling process, as we will discuss in Section 3.3 in detail. As for the inverse dynamics training, we sample the observation transitions of each agent to predict the action.

It is worth noting that the choice of whether agents should share their parameters of ϵ_θ^i and I_{ϕ^i} depends on the homogeneous nature and requirements of tasks. If agents choose to share their parameters, only one shared DM and inverse dynamics model are used for generating all agents' trajectories; otherwise, each agent i has extra parameters (*i.e.*, the U-Net and inverse dynamic models) to generate their states and predict their actions. The attention modules are always shared to incorporate global information into generating each agent's trajectory.

3.3 Centralized Control or Decentralized Execution

Centralized control. A direct and straightforward way to utilize MADIFF in online decision-making tasks is to have a centralized controller for all agents. The centralized controller has access to all agents' current local observations and generates all agents' trajectories along with predicting their actions, which are sent to every single agent for acting in the environment. This is applicable for multi-agent trajectory prediction problems and when interactive agents are permitted to be centralized controlled, such as in team games. During the generation process, we sample an initial noise trajectory $\hat{\boldsymbol{\tau}}_K$, condition the current joint states of all agents and the global information to utilize $\boldsymbol{y}(\boldsymbol{\tau}_0)$; following the diffusion step described in Equation (4) with ϵ_θ, we finally sample the joint observation sequence $\hat{\boldsymbol{\tau}}_0$ as below:

$$\underbrace{[\boldsymbol{o}_t, \cdots, \tilde{\boldsymbol{x}}_{K,t+H-1}]}_{\hat{\boldsymbol{\tau}}_K} \xrightarrow{K \text{ steps}} \underbrace{[\boldsymbol{o}_t, \cdots, \hat{\boldsymbol{o}}_{t+H-1}]}_{\hat{\boldsymbol{\tau}}_0} , \tag{7}$$

where every $\tilde{\boldsymbol{x}}_{K,t} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is a noise vector sampled from the normal Gaussian. After generation, each agent obtains the action through its own inverse dynamics model following Equation (3) using the current observation o_t^i and the predicted next observation $\hat{o}_{t+1}^i$, and takes a step in the environment. We highlight that MADIFF provides an efficient way to generate joint actions and the attention module guarantees sufficient feature interactions and information interchange among agents.

Decentralized execution with teammate modeling. Compared with centralized control, a more popular and widely-adopted setting is that each agent only makes its own decision without any communication with other agents, which is what most current works [Lowe et al., 2017, Rashid et al., 2020, Wang et al., 2023] dealt with. In this case, we can only utilize the current local observation of each agent i to plan its own trajectory. To this end, the initial noisy trajectory is conditioned on the current observation of the agent i. Similar to the centralized case, by iterative diffusion steps, we finally sample the joint state sequence based on the local observation of agent i as:

$$\underbrace{\begin{bmatrix} \tilde{x}_{K,t}^0, \cdots, \tilde{x}_{K,t+H-1}^0 \\ \cdots, \\ o_t^i, \cdots, \tilde{x}_{K,t+H-1}^i \\ \cdots, \\ \tilde{x}_{K,t}^N, \cdots, \tilde{x}_{K,t+H-1}^N \end{bmatrix}}_{\hat{\boldsymbol{\tau}}_K^i} \xrightarrow{K \text{ steps}} \underbrace{\begin{bmatrix} \hat{o}_t^0, \cdots, \hat{o}_{t+H-1}^0 \\ \cdots, \\ o_t^i, \cdots, \hat{o}_{t+H-1}^i \\ \cdots, \\ \hat{o}_t^N, \cdots, \hat{o}_{t+H-1}^N \end{bmatrix}}_{\hat{\boldsymbol{\tau}}_0^i} , \tag{8}$$

and we can also obtain the action through the agent i's inverse dynamics model as mentioned above. An important observation is that, the decentralized execution of MADIFF includes teammate modeling such that the agent i infers all others' observation sequences based on its own local observation. We show in experiments that this achieves great performances in various tasks, indicating the effectiveness of teammate modeling and the great ability in coordination.

History-based generation. We find DMs are good at modeling the long-term joint distributions, and as a result MADIFF perform better in some cases when we choose to condition on the trajectory of the past history instead of only the current observation. This implies that we replace the joint observation $\boldsymbol{o}_t$ in Equation (7) as the C-length joint history sequence $\boldsymbol{h}_t := [\boldsymbol{o}_{t-C}, \cdots, \boldsymbol{o}_{t-1}, \boldsymbol{o}_t]$, and replace the independent observation o_t^i in Equation (8) as the history sequence $h_t^i := [o_{t-C}^i, \cdots, o_{t-1}^i, o_t^i]$ of each agent i. Appendix Section D illustrates how agents' history and future trajectories are modeled by MADIFF in both centralized control and decentralized execution.

5

4 Related Work

Multi-agent Offline RL. While offline RL has become an active research topic, only a limited number of works studied offline MARL due to the challenge of offline coordination. Jiang and Lu [2021] extended BCQ [Fujimoto et al., 2019], a single-agent offline RL algorithm with policy regularization to multi-agent; Yang et al. [2021] developed an implicit constraint approach for offline Q learning, which was found to perform particularly well in MAL tasks; Pan et al. [2022] argued the actor update tends to be trapped in local minima when the number of agents increases, and correspondingly proposed an actor regularization method named OMAR. All of these Q-learning-based methods naturally have extrapolation error problem [Fujimoto et al., 2019] in offline settings, and their solution cannot get rid of it but only mitigate some. As an alternative, MADT [Meng et al., 2021] formulated offline MARL as return-conditioned supervised learning, and use a similar structure to a previous transformer-based offline RL work [Chen et al., 2021]. However, offline MADT learns an independent model for each agent without modeling agent interactions; it relies on the gradient from centralized critics during online fine-tuning to integrate global information into each agent's decentralized policy. MADIFF not only avoids the problem of extrapolation error, but also achieves the modeling of collaborative information while allowing CTDE in a completely offline training manner.

Diffusion Models for Decision-Making. There is a recent line of work applying diffusion models (DMs) to decision-making problems such as RL and imitation learning. Janner et al. [2022] design a diffusion-based trajectory generation model and train a value function to sample high-rewarded trajectories. A consequent work [Ajay et al., 2023] takes conditions as inputs to the DM, thus bringing more flexibility that generates behaviors that satisfy combinations of diverse conditions. Another line of work [Wang et al., 2022, Hansen-Estruch et al., 2023, Kang et al., 2024] uses the DM as a form of policy, *i.e.*, generating actions conditioned on states, and the training objective behaves as a regularization under the framework of TD-based offline RL algorithms. Different from the above, SynthER [Lu et al., 2024] adopts the DM to upsample the rollout data to facilitate learning of any RL algorithms. All of these existing methods focus on solving single-agent tasks. The proposed MADIFF is structurally similar to Ajay et al. [2023], but includes effective modules to model agent coordination in MAL tasks.

Opponent Modeling in MARL. Our modeling of teammates can be placed under the larger framework of opponent modeling, which refers to the process by which an agent tries to infer the behaviors or intentions of other agents using its local information. There is a rich literature on utilizing opponent modeling in online MARL. Rabinowitz et al. [2018] used meta-learning to build three models, and can adapt to new agents after observing their behavior. SOM [Raileanu et al., 2018] uses the agent's own goal-conditioned policy to infer other agents' goals from a maximum likelihood perspective. LIAM [Papoudakis et al., 2021] extracts representations of other agents with variational auto-encoders conditioned on the controlled agent's local observations. Considering the impact of the ego agent's policy on other agents' policies, LOLA [Foerster et al., 2017] and following works [Willi et al., 2022, Zhao et al., 2022] instead model the parameter update of the opponents. Different from these methods, MADIFF can use the same generative model to jointly output plans of its own trajectory and predictions of other agents' trajectories and is shown to be effective in offline settings.

5 Experiments

In experiments, we are aiming at excavating the ability of MADIFF in modeling the complex interactions among cooperative agents, particularly, whether MADIFF is able to (i) generate high-quality multi-agent trajectories; (ii) appropriately infer teammates' behavior; (iii) learn effective, coordinated policies from offline data.

5.1 Task Descriptions

We conduct experiments on multiple commonly used multi-agent testbeds.

- **Multi-agent particle environments (MPE)** [Lowe et al., 2017]: multiple 2D particles cooperate to achieve a common goal. *Spread*, three agents start at some random locations and have to cover three landmarks without collisions; *Tag*, three predators try to catch a pre-trained prey opponent

6

Table 1: The average score on offline MARL tasks. Shaded columns represent our methods. The mean and standard error are computed over 5 different seeds.

Testbed	Task	Dataset	BC	MA-ICQ	MA-TD3+BC	MA-CQL	OMAR	MADT	MADIFF-D	MADIFF-C
MPE	Spread	Expert	35.0 ± 2.6	104.0 ± 3.4	108.3 ± 3.3	98.2 ± 5.2	**114.9 ± 2.6**	-	95.0 ± 5.3	**116.7 ± 3.0**
		Md-Replay	10.0 ± 3.8	13.6 ± 5.7	15.4 ± 5.6	20.0 ± 8.4	**37.9 ± 12.3**	-	30.3 ± 2.5	**42.2 ± 8.1**
		Medium	31.6 ± 4.8	29.3 ± 5.5	29.3 ± 4.8	34.1 ± 7.2	47.9 ± 18.9	-	**64.9 ± 7.7**	58.2 ± 1.7
		Random	-0.5 ± 3.2	6.3 ± 3.5	9.8 ± 4.9	24.0 ± 9.8	**34.4 ± 5.3**	-	6.9 ± 3.1	4.3 ± 2.6
	Tag	Expert	40.0 ± 9.6	113.0 ± 14.4	115.2 ± 12.5	93.9 ± 14.0	116.2 ± 19.8	-	120.9 ± 14.6	**167.6 ± 18.6**
		Md-Replay	0.9 ± 1.4	34.5 ± 27.8	28.7 ± 20.9	24.8 ± 17.3	47.1 ± 15.3	-	62.3 ± 9.2	**95.0 ± 9.7**
		Medium	22.5 ± 1.8	63.3 ± 20.0	65.1 ± 29.5	61.7 ± 23.1	66.7 ± 23.2	-	77.2 ± 10.4	**132.9 ± 15.0**
		Random	1.2 ± 0.8	2.2 ± 2.6	5.7 ± 3.5	5.0 ± 8.2	**11.1 ± 2.8**	-	3.2 ± 4.0	10.7 ± 4.0
	World	Expert	33.0 ± 9.9	109.5 ± 22.8	110.3 ± 21.3	71.9 ± 28.1	110.4 ± 25.7	-	122.6 ± 14.4	**174.0 ± 16.8**
		Md-Replay	2.3 ± 1.5	12.0 ± 9.1	17.4 ± 8.1	29.6 ± 13.8	42.9 ± 19.5	-	57.1 ± 10.7	**83.0 ± 4.4**
		Medium	25.3 ± 2.0	71.9 ± 20.0	73.4 ± 9.3	58.6 ± 11.2	74.6 ± 11.5	-	123.5 ± 4.5	**158.2 ± 6.3**
		Random	-2.4 ± 0.5	1.0 ± 3.2	2.8 ± 5.5	0.6 ± 2.0	**5.9 ± 5.2**	-	2.0 ± 3.0	**8.1 ± 3.5**
MA Mujoco	2halfcheetah	Good	6846 ± 574	-	7025 ± 439	-	1434 ± 1903	-	8246 ± 342	**8514 ± 336**
		Medium	1627 ± 187	-	**2561 ± 82**	-	1892 ± 220	-	2207 ± 23	2203 ± 65
		Poor	465 ± 59	-	736 ± 72	-	384 ± 420	-	**759 ± 18**	760 ± 15
	2ant	Good	2697 ± 267	-	2922 ± 194	-	464 ± 469	-	2946 ± 77	**3069 ± 60**
		Medium	1145 ± 126	-	744 ± 283	-	799 ± 186	-	1211 ± 69	**1243 ± 37**
		Poor	954 ± 80	-	**1256 ± 122**	-	857 ± 73	-	946 ± 66	1038 ± 26
	4ant	Good	2802 ± 133	-	2628 ± 971	-	344 ± 631	-	**3080 ± 38**	3068 ± 44
		Medium	1617 ± 153	-	**1843 ± 494**	-	929 ± 349	-	1649 ± 100	1871 ± 52
		Poor	1033 ± 122	-	1075 ± 96	-	518 ± 112	-	**1295 ± 57**	1353 ± 44
SMAC	3m	Good	16.0 ± 1.0	18.8 ± 0.6	-	**19.6 ± 0.3**	-	19.1 ± 0.5	19.3 ± 0.6	**19.9 ± 0.1**
		Medium	8.2 ± 0.8	18.1 ± 0.7	-	**18.9 ± 0.7**	-	15.8 ± 0.4	17.3 ± 0.5	**18.1 ± 0.6**
		Poor	4.4 ± 0.1	**14.4 ± 1.2**	-	5.8 ± 0.4	-	4.4 ± 0.3	9.6 ± 1.7	9.5 ± 0.5
	2s3z	Good	18.2 ± 0.4	**19.6 ± 0.3**	-	19.0 ± 0.8	-	19.3 ± 0.2	**19.6 ± 0.3**	**19.7 ± 0.3**
		Medium	12.3 ± 0.7	17.2 ± 0.6	-	14.3 ± 2.0	-	15.0 ± 0.6	**17.4 ± 0.2**	**17.6 ± 0.3**
		Poor	6.7 ± 0.3	**12.1 ± 0.4**	-	10.1 ± 0.7	-	7.0 ± 0.3	9.8 ± 0.2	10.4 ± 0.7
	5m6m	Good	16.6 ± 0.6	16.3 ± 0.9	-	13.8 ± 3.1	-	16.7 ± 0.1	**17.8 ± 0.8**	**18.0 ± 0.8**
		Medium	12.4 ± 0.9	15.3 ± 0.7	-	17.0 ± 1.2	-	16.6 ± 0.2	**17.3 ± 0.5**	**18.0 ± 0.8**
		Poor	7.5 ± 0.2	9.4 ± 0.4	-	**10.4 ± 1.0**	-	7.8 ± 0.4	8.9 ± 0.2	10.3 ± 1.3
	8m	Good	16.7 ± 0.4	**19.6 ± 0.3**	-	11.3 ± 6.1	-	18.4 ± 0.3	19.2 ± 0.1	**19.8 ± 0.4**
		Medium	10.7 ± 0.5	18.6 ± 0.5	-	16.8 ± 3.1	-	18.5 ± 0.3	18.9 ± 0.9	**19.4 ± 0.9**
		Poor	5.3 ± 0.1	**10.8 ± 0.8**	-	4.6 ± 2.4	-	4.7 ± 0.1	5.1 ± 0.1	5.1 ± 0.1

that moves faster and needs cooperative containment; *World*, also requires three predators to catch a pre-trained prey, whose goal is to eat the food on the map while not getting caught, and the map has forests that agents can hide and invisible from the outside.

- **Datasets**: we use the offline datasets constructed by Pan et al. [2022], including four datasets collected by policies of different qualities trained by MATD3 [Ackermann et al., 2019], namely, Expert, Medium-Replay (Md-Replay), Medium and Random.

- **Multi-Agent Mujoco (MA Mujoco)** [Peng et al., 2021]: independent agents control different subsets of a robot's joints to run forward as fast as possible. We use three configurations: *2-agent halfcheetah (2halfcheetah)*, *2-agent ant (2ant)*, and *4-agent ant (4ant)*.

- **Datasets**: we use the off-the-grid offline dataset [Formanek et al., 2023], including three datasets with different qualities for each robot control task, *e.g.*, Good, Medium, and Poor.

- **StarCraft Multi-Agent Challenge (SMAC)** [Samvelyan et al., 2019]: a team of either homogeneous or heterogeneous units collaborates to fight against the enemy team that is controlled by the hand-coded built-in StarCraft II AI. We cover four maps: *3m*, both teams control three Marines; *2s3z*, both teams control two Stalkers and 3 Zealots; *5m_vs_6m (5m6m)*, requires controlling five Marines and the enemy team has six Marines; *8m*, both teams control eight Marines.

- **Datasets**: we use the off-the-grid offline dataset [Formanek et al., 2023], including three datasets with different qualities for each map, *e.g.*, Good, Medium, and Poor.

- **Multi-Agent Trajectory Prediction (MATP)**: different from the former offline MARL challenges which should learn the policy for each agent, the MATP problem only requires predicting the future behaviors of all agents, and no decentralized model is needed.

- **NBA dataset**: the dataset consists of various basketball players' recorded trajectories from 631 games in the 2015-16 season. Following Alcorn and Nguyen [2021], we split 569/30/32 training/validation/test games, with downsampling from 25 Hz to 5Hz. Different from MARL tasks, other information apart from agents' historical trajectories is available for making predictions, including the ball's historical trajectories, player ids, and a binary variable indicating the side of each player's frontcourt. Each term is encoded and concatenated with diffusion time embeddings as side inputs to each U-Net block.

7

5.2 Compared Baselines and Metrics

For offline MARL experiments, we use the episodic return obtained in online rollout as the performance measure. We include MA-ICQ [Yang et al., 2021] and MA-CQL [Kumar et al., 2020] as baselines on all offline RL tasks. On MPE, we also include OMAR and MA-TD3+BC [Fujimoto and Gu, 2021] in baseline algorithms and use the results reported by Pan et al. [2022]. On MA Mujoco, baseline results are adopted from Formanek et al. [2023]. On SMAC, we include MADT [Meng et al., 2021] as a sequence modeling baseline, while other baseline results are reported by Formanek et al. [2023]. We implement independent behavior cloning (BC) as a naive supervised learning baseline.

We use distance-based metrics including average displacement error (ADE) $\frac{1}{L \cdot N} \sum_{t=1}^{L} \sum_{i=1}^{N} \|\hat{o}_t^i - o_t^i\|$ and final displacement error (FDE) $\frac{1}{N} \sum_{i=1}^{N} \|\hat{o}_L^i - o_L^i\|$, where L is the prediction length [Li et al., 2020]. We also report minADE$_{20}$ and minFDE$_{20}$ as additional metrics to balance the stochasticity in sampling, which are the minimum ADE and FDE among 20 predicted trajectories, respectively. We compare MADIFF with Baller2Vec++ [Alcorn and Nguyen, 2021], an autoregressive MATP algorithm based on the transformer structure and specifically designed for the NBA dataset.

5.3 Numerical Results

We reported the numerical results both for the CTDE version of MADIFF (denoted as MADIFF-D) and the centralized version MADIFF (MADIFF-C). For offline MARL, since baselines are tested in a decentralized style, *i.e.*, all agents independently decide their actions with only local observations, MADIFF-C is not meant to be a fair comparison but to show if MADIFF-D fills the gap for coordination without global information. For MATP, due to its centralized prediction nature, MADIFF-C is the only variant involved.

Offline MARL. As listed in Table 1, MADIFF-D achieves the best result on most of the datasets. Similar to the single-agent case, direct supervised learning (BC) on the dataset behaves poorly when datasets are mixed quality. Offline RL algorithms such as MA-CQL that compute conservative values have a relatively large drop in performance when the dataset quality is low. Part of the reason may come from the fact that those algorithms are more likely to fall into local optima in multi-agent scenarios [Pan et al., 2022]. Thanks to the distributional modeling ability of the DM, MADIFF-D generally obtains better or competitive performance compared with OMAR [Pan et al., 2022] without any design for avoiding bad local optima similar to Pan et al. [2022]. On SMAC tasks, MADIFF-D achieves comparable performances, although it is slightly degraded compared with MADIFF-C.

MATP on the NBA dataset. In Table 2, when comparing ADE and FDE, MADIFF-C significantly outperforms the baseline; however, our algorithm only slightly beats baseline for minADE$_{20}$, and has higher minFDE$_{20}$. We suspect the reason is that Baller2Vec++ has a large prediction variance. When Baller2Vec++ only predicts one trajectory, a few players' trajectories deviate from the truth so far that deteriorate the overall ADE and FDE. When allowing to sample 20 times and calculating the minimum ADE/FDE according to the ground truth, Baller2Vec++ can choose the best trajectory for every single agent, which makes minADE$_{20}$ and minFDE$_{20}$ significantly smaller than one-shot metrics. However, considering it may be not practical to select the best trajectories without access to the ground truth, MADIFF-C is much more stable than Baller2Vec++. Predicted trajectories of MADIFF-C and Baller2Vec++ are provided in the Appendix Section H.4.

Table 2: Multi-agent trajectory prediction results on NBA dataset across 3 seeds, given the first step of all agents' positions.

Traj. Len.	Metric	Baller2Vec++	MADIFF-C
20	ADE	15.15 ± 0.38	$\mathbf{7.92 \pm 0.86}$
	FDE	24.91 ± 0.68	$\mathbf{14.06 \pm 1.16}$
	minADE$_{20}$	5.62 ± 0.05	$\mathbf{5.20 \pm 0.04}$
	minFDE$_{20}$	$\mathbf{5.60 \pm 0.12}$	7.61 ± 0.19
64	ADE	32.07 ± 1.93	$\mathbf{17.24 \pm 0.80}$
	FDE	44.93 ± 3.02	$\mathbf{26.69 \pm 0.13}$
	minADE$_{20}$	14.72 ± 0.53	$\mathbf{11.40 \pm 0.06}$
	minFDE$_{20}$	$\mathbf{10.41 \pm 0.36}$	11.26 ± 0.26

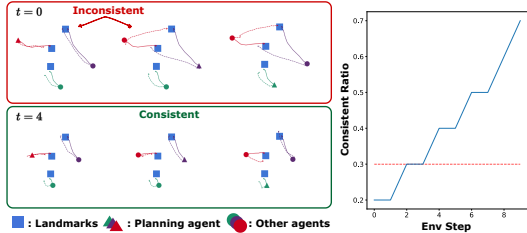



Figure 2: Visualization of an episode in the *Spread* task. Solid lines are real rollouts, and dashed lines are DM-planned trajectories.

8

5.4 Qualitative Analysis on Teammate modeling

We discuss the quality of teammate modeling as mentioned in Section 3.3 and how it is related to the decentralized execution scenario. In Figure 2 left, we visualize an episode generated by MADIFF-D trained on the Expert dataset of *Spread* task. The top and bottom rows are snapshots of entities' positions on the initial and intermediate time steps. The three rows from left to right in each column represent the perspectives of the three agents, red, purple, and green, respectively. Dashed lines are the planned trajectories for the controlled agent and other agents output by DMs, and solid lines are the real rollout trajectories. We observe that at the start, the red agent and the purple agent generate *inconsistent* plans, where both agents decide to move towards the middle landmark and assume the other agent is going to the upper landmark. At the intermediate time step, when the red agent is close to the middle landmark while far from the uppermost ones, the purple agent altered the planned trajectories of both itself and the red teammate, which makes all agents' plans *consistent* with each other. This particular case indicates that MADIFF is able to correct the prediction of teammates' behaviors during rollout and modify each agent's own desired goal correspondingly.

In Figure 2 right, we demonstrate that such corrections of teammate modeling are common and can help agents make globally coherent behaviors. We sample 100 episodes with different initial states and define *Consistent Ratio* at some time step t as the proportion of episodes in which the three agents make consistent planned trajectories. We plot the curve up to step $t = 9$, which is approximately halfway through the episode length limit in MPE. The horizontal red line represents how many portions of the real rollout trajectories are consistent at step $t = 9$. The interesting part is that the increasing curve reaches the red line before $t = 9$, and ends up even higher. This indicates that the planned teammates' trajectories are guiding the multi-agent interactions beforehand, which is a strong exemplar of the benefits of MADIFF's teammate modeling abilities. We also include visualizations of imagined teammate observation sequences in SMAC *3m* task in the Appendix Section H.3.

5.5 Ablation Study

Our key argument is that the great coordination ability of MADIFF is brought by the attention modules among individual agents' diffusion networks. We validate this insight through a set of ablation experiments on MPE. We compare MADIFF-D with independent DMs, *i.e.*, each agent learns from corresponding offline data using independent U-Nets without attention. We denote this variant as MADIFF-D-Ind. In addition, we also ablate the choice of whether each agent should share parameters of their basic U-Net, noted as Share or NoShare. Without causing ambiguity, we omit the name of MADIFF, and notate the different variants as *D-Share*, *D-NoShare*, *Ind-Share*, *Ind-NoShare*.

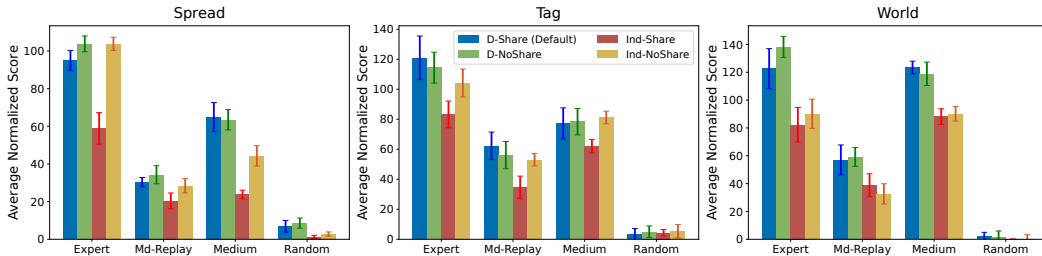



Figure 3: The average normalized score of MADIFF ablation variants in MPE tasks. The mean and standard error are computed over 5 different seeds.

As is obviously observed in Figure 3, with attention modules, MADIFF-D significantly exceeds that of the independent version on most tasks, justifying the importance of inter-agent attentions. The advantage of MADIFF-D is more evident when the task becomes more challenging and the data becomes more confounded, e.g., results on World, where the gap between centralized and independent models is larger, indicating the difficulty of solving offline coordination with independently trained models. As for the parameter sharing choice, the performance of MADIFF-D-Share and MADIFF-D-NoShare is similar overall. Since MADIFF-D-Share has fewer parameters, we prefer MADIFF-D-Share, and use it as the default variant to be reported in Table 1. Another advantage of sharing U-Net parameters is that the trajectories of various agents can be batched together and fed through the network. This not only decreases sampling time but also renders it insensitive to an increasing number of agents. We provide a specific example in Appendix Section G.4.

9

5.6 Limitations

Scalability to many agents. MADIFF-D requires each agent to infer all teammates' future trajectories, which is difficult and unnecessary in environments with a large number of agents. Although we have done experiments on a maximum number of 8 agents (SMAC *8m*), MADIFF-D is in general not suitable for scenarios with tens or hundreds of agents. A potential solution is to infer a latent representation of teammates' trajectories.

Applicability in highly stochastic environments. Several theoretical and empirical studies [Paster et al., 2022, Brandfonbrener et al., 2022, Chen et al., 2021] have demonstrated that in offline RL, sequence modeling algorithms tend to underperform Q-learning-based algorithms in environments with high stochasticity. This is primarily because sequence modeling algorithms are more susceptible to high-reward offline trajectories that are achieved by chance. Since MADIFF is a sequence modeling algorithm, it shares this weakness. To assess how much MADIFF is affected by environmental stochasticity, we conducted experiments on the *terran_5_vs_5* map in SMACv2 [Ellis et al., 2022]. The design principle of SMACv2 is to add stochasticity to the original SMAC environment, including randomized initial positions and unit types. We conducted experiments under four settings: the original version, without position randomness, without unit type randomness, and without both kinds of randomness. MADIFF performs worse than the Q-learning-based method only when both kinds of stochasticity are present. In all settings, MADIFF outperforms the sequence modeling baseline. Detailed experimental settings and results can be found in Appendix Section H.1.

6 Conclusion

In this paper, we propose MADIFF, a novel generative multi-agent learning framework, which is realized with an attention-based diffusion model designed to model the complex coordination among multiple agents. To our knowledge, MADIFF is the first diffusion-based offline multi-agent learning algorithm, which behaves as both a decentralized policy and a centralized controller including teammate modeling, and can be used for multi-agent trajectory prediction. Our experiments indicate strong performance compared with a set of recent offline MARL baselines on a variety of tasks.

Acknowledgements

The SJTU team is partially supported by National Key R&D Program of China (2022ZD0114804), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and National Natural Science Foundation of China (62322603, 62076161).

References

Johannes Ackermann, Volker Gabler, Takayuki Osa, and Masashi Sugiyama. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465*, 2019.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *International Conference on Learning Representations*, 2023.

Michael A Alcorn and Anh Nguyen. baller2vec++: A look-ahead multi-entity transformer for modeling coordinated agents. *arXiv preprint arXiv:2104.11980*, 2021.

Itamar Arel, Cong Liu, Tom Urbanik, and Airton G Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135, 2010.

David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? *arXiv preprint arXiv:2206.01079*, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

10

Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489*, 2022.

Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

Claude Formanek, Asad Jeewa, Jonathan Shock, and Arnu Pretorius. Off-the-grid marl: a framework for dataset generation with baselines for cooperative 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 and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies, 2023.

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

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

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

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

Jiechuan Jiang and Zongqing Lu. Offline decentralized multi-agent reinforcement learning. *arXiv preprint arXiv:2108.01832*, 2021.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

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.

Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. *Advances in neural information processing systems*, 33:19783–19794, 2020.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

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

Linghui Meng, Muning Wen, Yaodong Yang, Chenyang Le, Xiyun Li, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, and Bo Xu. Offline pre-trained multi-agent decision transformer: One big sequence model tackles all smac tasks. *arXiv e-prints*, pages arXiv–2112, 2021.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

11

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*, pages 17221–17237. PMLR, 2022.

Georgios Papoudakis, Filippos Christianos, and Stefano Albrecht. Agent modelling under partial observability for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:19210–19222, 2021.

Keiran Paster, Sheila McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers and rvs fail in stochastic environments. *Advances in neural information processing systems*, 35: 38966–38979, 2022.

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

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR, 2018.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pages 4257–4266. PMLR, 2018.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020.

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

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xihuai Wang, Zheng Tian, Ziyu Wan, Ying Wen, Jun Wang, and Weinan Zhang. Order matters: Agent-by-agent policy optimization. In *The Eleventh International Conference on Learning Representations*, 2023.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

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

Timon Willi, Alistair Hp Letcher, Johannes Treutlein, and Jakob Foerster. Cola: consistent learning with opponent-learning awareness. In *International Conference on Machine Learning*, pages 23804–23831. PMLR, 2022.

Chenjun Xiao, Yifan Wu, Chen Ma, Dale Schuurmans, and Martin Müller. Learning to combat compounding-error in model-based reinforcement learning. *arXiv preprint arXiv:1912.11206*, 2019.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

12

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.

Stephen Zhao, Chris Lu, Roger B Grosse, and Jakob Foerster. Proximal learning with opponent-learning awareness. *Advances in Neural Information Processing Systems*, 35:26324–26336, 2022.

13

A Outline

In this appendix, we provide a table to explain the main notations we used in Section B. In Section C, we give the pseudocode of multi-agent planning and multi-agent trajectory prediction with MADIFF model. In Section D, we demonstrate how multiple agents' trajectories are modeled by MADIFF during centralized control and decentralized execution in an example three-agent environment. In Section E, we give additional information on offline datasets, including how they are collected, violin plots of return distributions, and a minor issue of MPE dataset. In Section F, we briefly describe the implementation of baseline algorithms and links to related resources. In Section G, we provide details of the experiments, including the normalization used to compute the average score, the detailed network illustration unrolling each agent's U-Net, crucial hyperparameters, and examples of wall-clock time and resources required for training and sampling from MADIFF. In Section H, we demonstrate and analyze additional experimental results. Specifically, we provide experiment results on SMACv2 to demonstrate how much MADIFF is affected by environmental stochasticity. We also provide ablation results to support the effectiveness of teammate modeling in MADIFF-D, show the quality of teammate modeling by MADIFF-D on SMAC tasks, and visualize predicted multi-player trajectories by MADIFF and the baseline algorithm on the NBA dataset.

B Notations

Table 3: List of main notations used in the paper.

Notation	Description
$\mathcal{S}, \mathcal{A}, \Omega$	state, action, and local observation spaces
γ	the discounted factor
N	number of controlled agents
s_t	state at step t
a_t^i, o_t^i	action and local observation of agent i at environment step t
$\boldsymbol{a}_t, \boldsymbol{o}_t$	joint action and observation of all agents at environment step t
$r(s, \boldsymbol{a})$	shared reward function
$\boldsymbol{\tau}$	joint trajectory of all agents
$\boldsymbol{y}(\boldsymbol{\tau})$	additional conditioning information
ϕ	parameters of the inverse dynamics model
θ	parameters of the diffusion model
$\tilde{x}_{k,t}^i$	noised observation of agent i at diffusion step k and environment step t
$\tilde{\boldsymbol{x}}_{k,t}$	noised joint observation at diffusion step k and environment step t
$\hat{o}_t^i$	predicted observation of agent i at environment step t
$\hat{\boldsymbol{\tau}}_k$	noised joint trajectory of all agents at diffusion step k
h_t^i	historical trajectory of agent i up to environment step t
$\boldsymbol{h}_t$	historical joint trajectory of all agents up to environment step t

C Algorithm

D Illustration of Multi-agent Trajectory Modeling

To provide a better understanding of how multiple agents' observations are modeled by MADIFF in centralized control and decentralized execution scenarios, we show illustrative examples in a typical three-agent environment in Figure 4. If the environment allows for centralized control, we can condition MADIFF on all agents' historical and current observations, and let the model sample all agents' future trajectories as a single sample, as shown in Figure 4a. Then the current and next observations are sent to the inverse dynamics model for action prediction. If only decentralized execution is permitted, as shown in Figure 4b, agent 1 can only condition the model on its own information. The historical and current observations of other agents are masked when performing conditioning. MADIFF now not only generates agent 1's own future trajectories but also predicts the

14

Algorithm 1 Multi-Agent Planning with MADIFF

1: **Input:** Noise model ϵ_θ, inverse dynamics I_ϕ, guidance scale ω, history length C, condition $\boldsymbol{y}$
2: Initialize $h \leftarrow \texttt{Queue}(\texttt{length} = C)$; $t \leftarrow 0$ // Maintain a history of length C
3: **while** not done **do**
4:     Observe joint observation $\boldsymbol{o}$; $h.\texttt{insert}(\boldsymbol{o})$; Initialize $\boldsymbol{\tau}_K \sim \mathcal{N}(\boldsymbol{0}, \alpha\boldsymbol{I})$
5:     **for** $k = K \dots 1$ **do**
6:        $\boldsymbol{\tau}_k[:\texttt{length}(h)] \leftarrow h$ // Constrain plan to be consistent with history
7:        **if** Centralized control **then**
8:           $\hat{\boldsymbol{\epsilon}} \leftarrow \epsilon_\theta(\boldsymbol{\tau}_k, k) + \omega(\epsilon_\theta(\boldsymbol{\tau}_k, \boldsymbol{y}, k) - \epsilon_\theta(\boldsymbol{\tau}_k, k))$ // Classifier-free guidance
9:           $(\boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1}) \leftarrow \texttt{Denoise}(\boldsymbol{\tau}_k, \hat{\boldsymbol{\epsilon}})$
10:        **else if** Decentralized execution **then**
11:           **for** agent $i \in \{1, 2, \dots, N\}$ **do**
12:              $\hat{\epsilon}^i \leftarrow \epsilon_\theta^i(\tau_k^i, k) + \omega(\epsilon_\theta^i(\tau_k^i, y^i, k) - \epsilon_\theta^i(\tau_k^i, k))$ // Classifier-free guidance
13:              $(\mu_{k-1}^i, \Sigma_{k-1}^i) \leftarrow \texttt{Denoise}(\tau_k^i, \hat{\epsilon}^i)$
14:           **end for**
15:        **end if**
16:        $\boldsymbol{\tau}_{k-1} \sim \mathcal{N}(\boldsymbol{\mu}_{k-1}, \alpha\boldsymbol{\Sigma}_{k-1})$
17:     **end for**
18:     Extract $(\boldsymbol{o}_t, \boldsymbol{o}_{t+1})$ from $\boldsymbol{\tau}_0$
19:     **for** agent $i \in \{1, 2, \dots, N\}$ **do**
20:        $a_t^i \leftarrow f_{\phi^i}(o_t^i, o_{t+1}^i)$
21:     **end for**
22:     Execute $\boldsymbol{a}_t$ in the environment; $t \leftarrow t + 1$
23: **end while**

Algorithm 2 Multi-Agent Trajectory Prediction with MADIFF

1: **Input:** Noise model ϵ_θ, guidance scale ω, condition $\boldsymbol{y}$, historical joint observations h with length C, predict horizon H
2: Initialize $\boldsymbol{\tau}_K \sim \mathcal{N}(\boldsymbol{0}, \alpha\boldsymbol{I})$
3: **for** $k = K \dots 1$ **do**
4:     $\boldsymbol{\tau}_k[:C] \leftarrow h$ // Constrain prediction to be consistent with history
5:     $\hat{\boldsymbol{\epsilon}} \leftarrow \epsilon_\theta(\boldsymbol{\tau}_k, k) + \omega(\epsilon_\theta(\boldsymbol{\tau}_k, \boldsymbol{y}, k) - \epsilon_\theta(\boldsymbol{\tau}_k, k))$ // Classifier-free guidance
6:     $(\boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1}) \leftarrow \texttt{Denoise}(\boldsymbol{\tau}_k, \hat{\boldsymbol{\epsilon}})$
7:     $\boldsymbol{\tau}_{k-1} \sim \mathcal{N}(\boldsymbol{\mu}_{k-1}, \alpha\boldsymbol{\Sigma}_{k-1})$
8: **end for**
9: Extract prediction $(\boldsymbol{o}_C, \boldsymbol{o}_{C+1}, \dots, \boldsymbol{o}_{C+H-1})$ from $\boldsymbol{\tau}_0$

---

current and future observations of the other two agents. Due to the joint modeling of all agents during training, such predictions are also reasonable and can be considered as a form of teammate modeling from agent 1's perspective. Although teammate modeling is not directly used in generating agent 1's ego actions, it can help agent 1 refine its planned trajectories to be consistent with the predictions of others.

E   Additional Information on offline datasets

E.1   MPE Datasets

For MPE experiments, we use datasets and a fork of environment[2] provided by OMAR [Pan et al., 2022]. They seem to be using an earlier version of MPE where agents can receive different rewards. For example, in the *Spread* task, team reward is defined using the distance of each landmark to its closest agent, which is the same for all agents. But when an agent collides with others, it will receive the team reward minus a penalty term. The collision reward has been brought into the team reward in the official repository since this commit[3]. However, the fork provided by OMAR still uses a

---

[2]https://github.com/ling-pan/OMAR
[3]https://github.com/openai/multiagent-particle-envs/commit/
6ed7cac026f0eb345d4c20232bafa1dc951c68e7

15

(a) MADIFF in centralized control.   (b) MADIFF in decentralized execution.

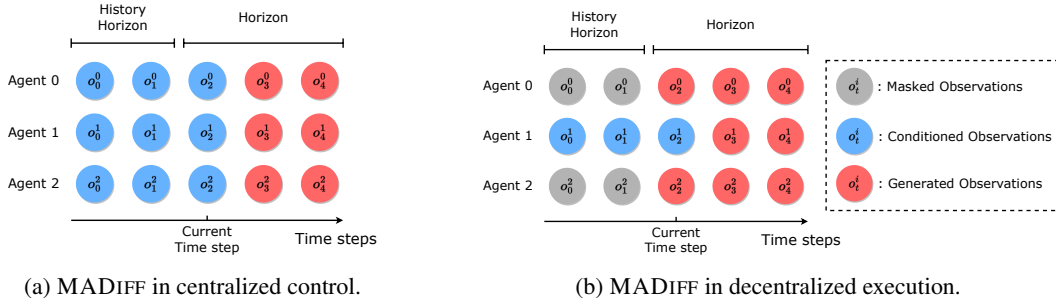



Figure 4: Illustration of how agents' observations are modelled by MADIFF in a three-agent environment. Note that figure (b) shows the situation when Agent 1 is taking action during decentralized execution.

legacy version. For fair and proper comparisons, we use OMAR's dataset and environment where all baseline models are trained and evaluated.

We have to note that different rewards for agents only happen at very few steps, which might not contradict the fully cooperative setting much. For example, OMAR's expert split of the *Spread* dataset consists of 1M steps, and different rewards are recorded only at less than 1.5% (14929) steps.

E.2 MA Mujoco Datasets

For MA Mujoco experiments, we adopt the off-the-grid dataset Formanek et al. [2023] and use *Good*, *Medium* and *Poor* datasets for each task. Each dataset is collected by three independently trained MA-TD3 policies, and a small amount of exploration noise is added to the policies for enhanced behavioral diversity.

For visualizations of the distribution of episode returns in each dataset, we provide violin plots of all datasets we used in Figure 5.

E.3 SMAC Datasets

For SMAC experiments, we adopt the off-the-grid dataset [Formanek et al., 2023] and use *Good*, *Medium* and *Poor* datasets for each map. Each dataset is collected by three independently trained QMIX policies, and a small amount of exploration noise is added to the policies for enhanced behavioral diversity.

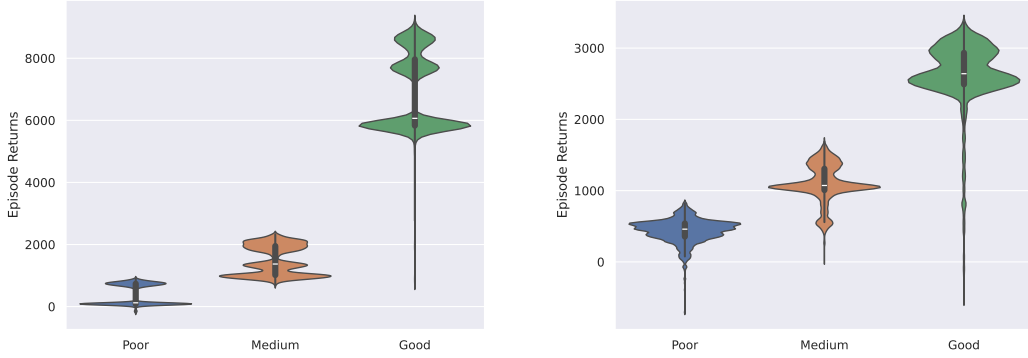
For visualizations of the distribution of episode returns in each dataset, we provide violin plots of all datasets we used in Figure 6.

F Baseline Implementations

Here we briefly describe how the baseline algorithms are implemented. For MATP experiments, we use the implementation from the official repository of Baller2Vec++[4]. Baseline results on MPE datasets are borrowed from Pan et al. [2022]. According to their paper, they build all algorithms upon a modified version of MADDPG[5], which uses decentralized critics for all methods. Baselines on SMAC datasets are implemented by Formanek et al. [2023], and the performances are adopted from their reported benchmark results. The open-sourced implementation and hyperparameter settings can be found in the official repository[6].

---

[4] https://github.com/airalcorn2/baller2vecplusplus
[5] https://github.com/shariqiqbal2810/maddpg-pytorch
[6] https://github.com/instadeepai/og-marl

16

(a) MA Mujoco 2halfcheetah.

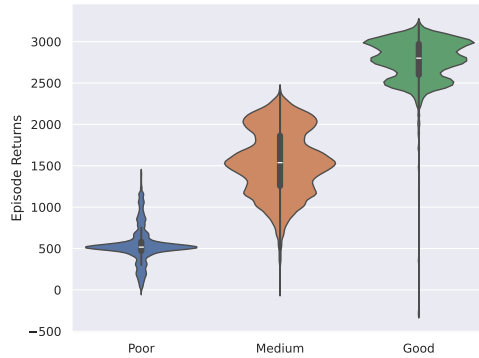
(b) MA Mujoco 2ant.

(c) MA Mujoco 4ant.

Figure 5: Violin plots of returns in MA Mujoco datasets.

G   Implementation Details

G.1   Score Normalization

The average scores of MPE tasks 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

$$S_{\text{norm}} = 100 \times (S - S_{\text{random}})/(S_{\text{expert}} - S_{\text{random}}) \,,$$

which follows Pan et al. [2022] and Fu et al. [2020]. The expert and random scores on Spread, Tag, and World are {516.8, 159.8}, {185.6, -4.1}, and {79.5, -6.8}, respectively.

G.2   Detailed Network Architecture

In Figure 7, we unroll the U-Net structure of different agents.

We describe the computation steps of attention among agents in formal. Each agent's local embedding c^i is passed through the key, value, and query network to form q^i, k^i, and v^i, respectively. Then the dot product with scaling is performed between all agents' q^i and k^i, which is followed by a Softmax operation to obtain the attention weight α^{ij}. Each α^{ij} can be viewed as the importance of j-th agent to the i-th agent at the current time step. The second dot product is carried out between the weight matrix and the value embedding v^i to get $\hat{c}^i$ after multi-agent feature interactions. Then $\hat{c}^i$ is skip-connected to the corresponding decoder block. The step-by-step computation of multi-agent

17

(a) SMAC 3m.　　　　　　　　　　　　　　(b) SMAC 5m_vs_6m.

(c) SMAC 2s3z.　　　　　　　　　　　　　　(d) SMAC 8m.

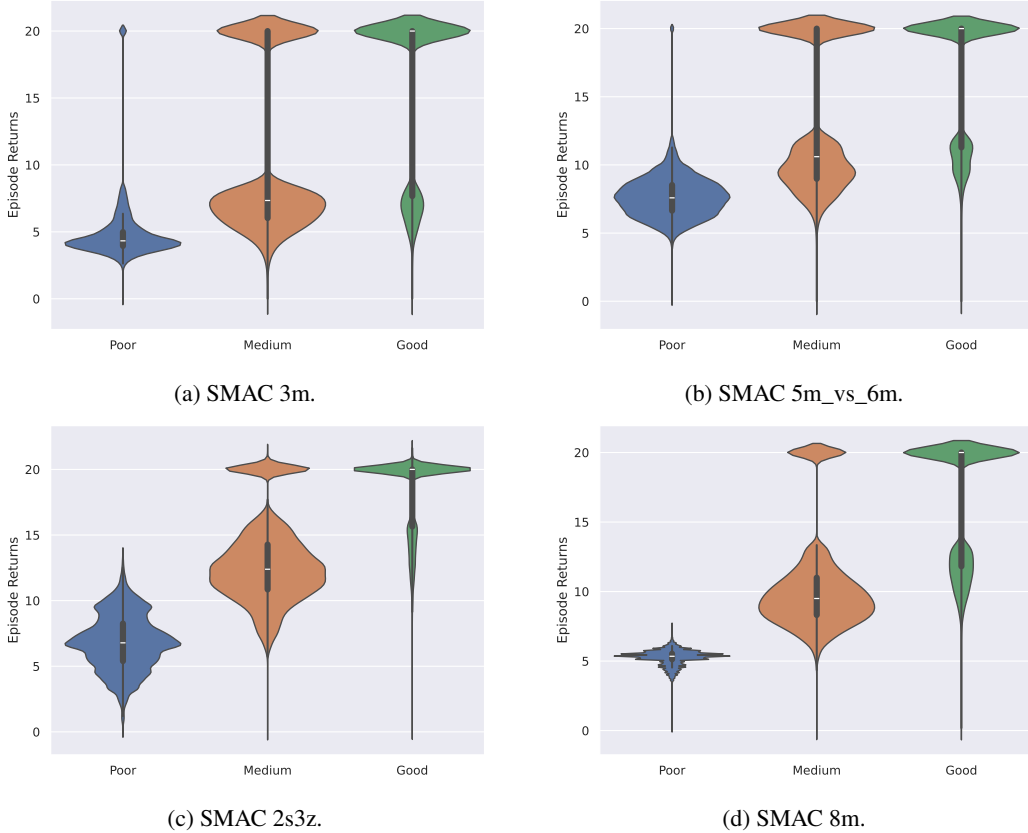



Figure 6: Violin plots of returns in SMAC datasets.

attention in MADIFF can be written as

$$q^i = f_{\text{query}}(c^i),\ k^i = f_{\text{key}}(c^i),\ v^i = f_{\text{value}}(c^i)\ ;$$

$$\alpha^{ij} = \frac{\exp(q^i k^j / \sqrt{d_k})}{\sum_{p=1}^{N} \exp(q^i k^p / \sqrt{d_k})}\ ;$$

$$\hat{c}^i = \sum_{j=1}^{N} \alpha^{ij} v^j\ ,$$

where d_k is the dimension of k^i.

G.3　Hyperparameters

We list the key hyperparameters of MADIFF we used in Table 4, Table 5, and Table 6. In all of our experiments, we use a scaling factor of 0.5 and β of 0.25. Return scale is the normalization factor used to divide the conditioned return before input to the diffusion model. The rough range of the return scale can be determined by the return distributions of the training dataset. We only tune the guidance weight ω, return scale, planning horizon H, and history horizon. We tried the guidance weight of $\{1.0, 1.2, 1.4, 1.6\}$, and found that different choices do not significantly affect final performances, we chose 1.2 for all experiments. For MPE tasks, we find it unnecessary to condition on history observation sequence; thus, we set all history horizons to zero.

G.4　Computing Resources and Wall Time

The training of MADIFF does not involve an iterative process, and thus, the training time is not related to the total number of diffusion steps. Thanks to the property that the sum of two independent

18

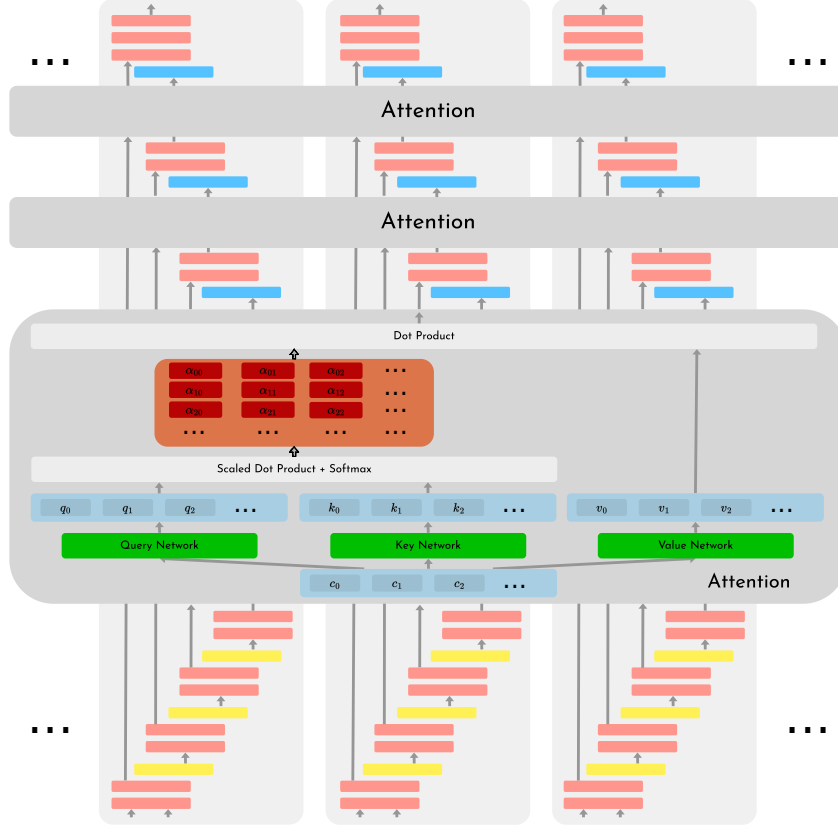



Figure 7: The detailed architecture of MADIFF. Each agent's U-Net is unrolled and lined up in the horizontal direction.

Table 4: Hyperparameters of MADIFF on MPE datasets.

TestBed	Spread				Tag				World			
Dataset	Expert	Md-Replay	Medium	Random	Expert	Md-Replay	Medium	Random	Expert	Md-Replay	Medium	Random
Return scale	350	200		50	350	200		50	200	100		10
Learning rate	2e-4											
Guidance scale ω	1.2											
Planning horizon H	24											
History horizon	0											
Batch size	32											
Diffusion steps K	200											
Reward discount γ	0.99											
Optimizer	Adam Optimizer											

Table 5: Hyperparameters of MADIFF on MA Mujoco datasets.

TestBed	2halfcheetah			4ant			2ant		
Dataset	Good	Medium	Poor	Good	Medium	Poor	Good	Medium	Poor
Return scale	1000	300	100	380	320	150	380	320	150
Learning rate	2e-4								
Guidance scale ω	1.2								
Planning horizon H	10								
History horizon	18								
Batch size	32								
Diffusion steps K	200								
Reward discount γ	0.99								
Optimizer	Adam Optimizer								

Gaussian random variables remains a Gaussian, the multistep forward process can be written in a closed form [Ho et al., 2020]:

$$q(\hat{\boldsymbol{\tau}}_k | \boldsymbol{\tau}_0) = \mathcal{N}(\hat{\boldsymbol{\tau}}_k; \sqrt{\bar{\alpha}_t} \boldsymbol{\tau}_0, (1 - \bar{\alpha}_t) \boldsymbol{I}) . \tag{9}$$

19

Table 6: Hyperparameters of MADIFF on SMAC datasets.

TestBed	3m			2s3z			5m6m			8m		
Dataset	Good	Medium	Poor	Good	Medium	Poor	Good	Medium	Poor	Good	Medium	Poor
Return scale	20		8	20		12	20		10	20		8
Learning rate	2e-4											
Guidance scale ω	1.2											
Planning horizon H	4											
History horizon	20											
Batch size	32											
Diffusion steps K	200											
Reward discount γ	1.0											
Optimizer	Adam Optimizer											

Therefore, the k-th step noisy trajectory in Equation (6) can be easily sampled from the Gaussian distribution above without an iterative process. We provide a concrete example to illustrate the time and resources required for training MADIFF. On a server equipped with an AMD Ryzen 9 5900X (12 cores) CPU and an RTX 3090 GPU, we trained the MADIFF-C model on the Expert dataset from the MPE Spread task, achieving convergence in approximately one hour. The curve depicting Wall-clock time spent on training and the corresponding model performance is shown in Figure 8.

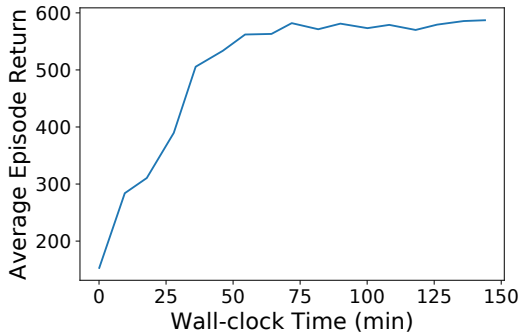



Figure 8: Wall-clock time and corresponding average episode return (average over 10 episodes) during training MADIFF-C for MPE Spread task.

Table 7: The wall-clock time spent when generating multi-agent trajectories. We fix the dimension of observation space to 88 and use DDIM of 15 steps during sampling. The history horizon is set to 20, and the planning horizon is 8. The results are obtained on a server with an AMD Ryzen 9 5900X (12 cores) CPU and an RTX 3090 GPU, and are averaged over 1000 trials. The computation time does not increase much with the number of agents thanks to GPU-accelerated computing.

Num. Agents (Incl. Ego)	8	16	32	64
Wall clock Time	124.25 ms	126.90 ms	127.65 ms	127.35 ms

In Table 7, we showcase the time required for sampling multi-agent trajectories with MADIFF as the number of agents increases. We can see that the sampling time does not differ much when generating different number of trajectories. Since we use shared U-Net models for all agents in our experiments, different agents' trajectories can be batched together and passed through the network. Therefore, using GPU-accelerated computing, the second part does not cost much more time than predicting each agent's trajectory during inference.

H   Additional Experimental Results

H.1   SMACv2 Experiments

To understand how much MADIFF is affected by environmental stochasticity, we conducted experiments on the *terran_5_vs_5* map in SMACv2 [Ellis et al., 2022]. SMACv2 is built upon SMAC with

20

a focus on higher stochasticity. Specifically, in SMACv2, the unit types and agent start positions are randomized at the beginning of each episode. As each agent can only observe a nearby area, such randomness results in increased stochasticity in environment transitions. There are two different types of starting positions, reflect and surround. In reflect settings, the map is splitted into two sides. Allied units and enemy units are randomly and uniformly spawned on different sides. In surround settings, allied units are spawned at the center of the map, and enemy units are randomly stationed along the four diagonals. In *terran_5_vs_5*, there are three different unit types: marine, marauder, and medivac. The default sampling probabilities of these three types are 0.45, 0.45 and 0.1.

We design four settings with different degree of stochasicity: the original version, without position randomness (w/o PR), without unit type randomness (w/o TR), and without both kinds of randomness (w/o PR&TR). To reduce position randomness, we only use surrounding settings. Note that this does not mean the staring positions of all units are fixed, since enemy units are still randomized along the four diagonals. To remove unit type randomness, we set all units to be marines. The dataset for the original version is the *terran_5_vs_5* Replay dataset from Formanek et al. [2023]. Datasets for other three stochasicity settings were collected by ourselves. We partially trained three MAPPO[7] models in each setting. Each model was then used to collect 500 episodes, resulting in a dataset comprising 1500 episodes for each setting.

Three algorithms are benchmarked under these four settings: MAICQ, which represents the state-of-the-art in Q-learning-based algorithms; MADT, a representative multi-agent sequence modeling baseline; and MADIFF-D. Results are presented in Table 8. We can see that MADIFF-D performs worse than MAICQ only when both kinds of stochasticity are present. As the environmental randomness diminishes, MADIFF-D's performance gradually catches up with and surpasses MAICQ. In all settings, MADIFF-D outperforms MADT.

Table 8: The average score on different settings of SMACv2 *terran_5_vs_5*. Shaded columns represent our method. The mean and standard error are computed over 3 different seeds.

Setting	MAICQ	MADT	MADIFF-D
Original	13.7 ± 1.7	8.2 ± 0.2	10.1 ± 0.8
w/o PR	16.0 ± 1.6	14.3 ± 0.8	16.1 ± 0.3
w/o TR	18.4 ± 0.5	14.6 ± 0.3	18.6 ± 0.2
w/o PR&TR	17.3 ± 0.3	16.8 ± 0.3	18.5 ± 0.2

H.2 Effectiveness of Teammate Modeling

To investigate whether teammate modeling can lead to performance improvements during decentralized execution, we conduct ablation experiments on MPE Spread datasets. We compare MADIFF-D with its variant that adopts the same network architecture but masks the diffusion loss on other agents' trajectories during training. We denote the variant as MADIFF-D w/o TM. The results are presented in Table 9, which show that teammate modeling results in notable performance improvements on all four levels of datasets.

Table 9: Ablation results of teammate modeling on MPE Spread datasets across 3 seeds.

Dataset	MADIFF-D w/o TM	MADIFF-D
Expert	93.4 ± 3.6	98.4 ± 12.7
Medium	35.4 ± 6.6	53.2 ± 2.3
Md-Replay	17.7 ± 4.3	42.9 ± 11.6
Random	5.7 ± 3.1	19.4 ± 2.9

---

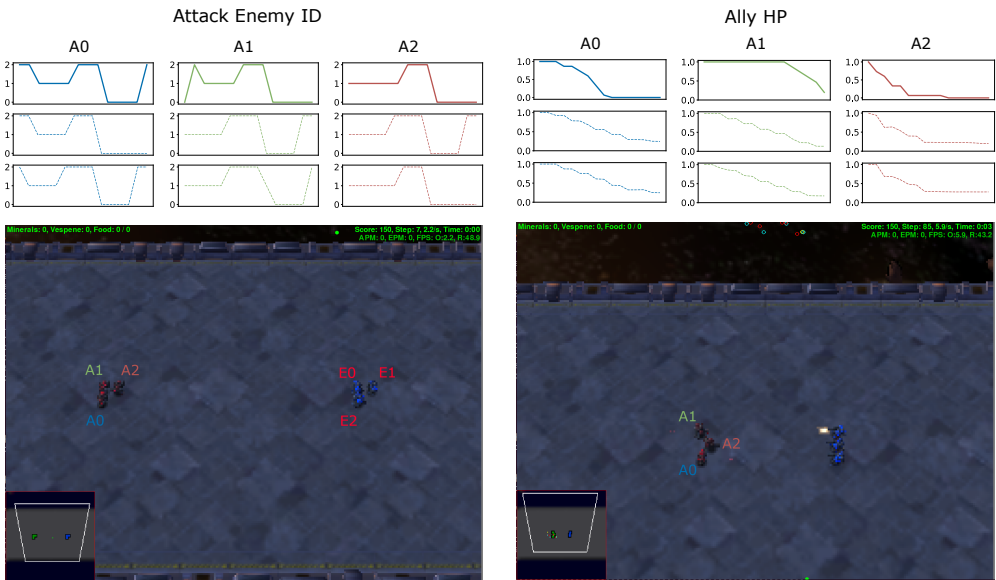
[7]https://github.com/marlbenchmark/on-policy

21

H.3 Teammate Modeling on SMAC Tasks

We show and analyze the quality of teammate modeling by MADIFF-D on SMAC. Specifically, we choose two time steps from an episode on 3m map to analyze predictions on allies' attack targets and health points (HP), respectively.

On top of Figure 9a is attacked enemy agent ID (0, 1, 2 stands for E0, E1, E2) of ally agents A0, A1, and A2. The first row is the ground-truth ID, and the second and the third rows are the predictions made by MADIFF-D from the other two allies' views. We can see that the predictions are in general consistent with the ground-truth ID. As can be seen from the true values of the attack enemy ID, agents tend to focus their firepower on the same enemies at the same time. And the accurate prediction of allies' attack enemy IDs intuitively can help to execute such a strategy.

In Figure 9b, we visualize the HP change curve of ally agents starting from another time step. From the environment state visualization below, agent A2 is the closest to enemies, so its HP drops the fastest. Such a pattern is successfully predicted by the other two agents.

(a) Ground-truth and predicted enemy's ID to attack by each ally agent.

(b) Ground-truth and predicted health points (HP) of each ally agent.

Figure 9: The ground-truth and predicted information of different MADIFF agents at two-time step. On the top of each figure, each column describes a different agent. The first row shows the change curve of the real value, and the last two rows below are the information predicted by other agents.

H.4 Predicted Trajectory Visualization on NBA Dataset

We visualize the players' moving trajectories predicted by MADIFF-C and Baller2Vec++ on the NBA dataset in Figure 10. In each image, the solid lines are real trajectories and the dashed lines are trajectories predicted by the model. The trajectories predicted by MADiff-C are closer to the real trajectories and are overall smoother compared to the Baller2Vec++ predictions.

22

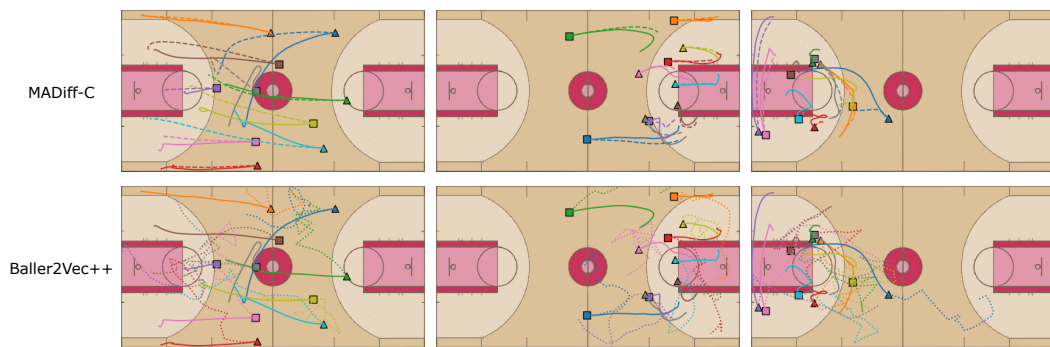



Figure 10: Real and Predicted multi-player trajectories by MADIFF-C and Baller2Vec++.

23

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction accurately reflect our contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We point out the limitations of our method in Section 5.6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

24

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We illustrate our model architecture in Figure 1 and list important hyperparameters in Appendix Section G.3. We also provide the source code in supplementary materials.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

25

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide code, anonymous data download link, and necessary instructions in supplementary materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide the experimental details in Section 5.1, Section 5.2, and Appendix Section E, Section F, Section G.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We provide error bars in Table 1, Table 2, and Figure 3. Reported error bar is standard deviation calculated over trials with different random seeds.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

26

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

    Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

    Answer: [Yes]

    Justification: We provide concrete instances of both training and sampling wall time and resources of our algorithm in Appendix Section G.4.

    Guidelines:

    - The answer NA means that the paper does not include experiments.
    - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
    - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
    - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

    Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

    Answer: [Yes]

    Justification: Our paper conform with the NeurIPS Code of Ethics.

    Guidelines:

    - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
    - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
    - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The proposed algorithm is a general solution for a wide range of offline multi-agent learning problems. In our opinion, there is no specific societal impact that should be stated explicitly.

    Guidelines:

27

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited and mentioned open-sourced implementation we used in Appendix Section F.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

28

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: Our paper does not publicly release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

29

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

30